Malicious Security Comes for Free in Consensus with Leaders

MARK ABSPOEL, CWI, Cryptology Group, the Netherlands
THOMAS ATTEMA, CWI, Cryptology Group, the Netherlands and Leiden University, Mathematical Institute, the Netherlands and TNO, Cyber Security and Robustness, the Netherlands
MATTHIEU RAMBAUD, Telecom Paris and Institut polytechnique de Paris, France

We consider Byzantine Agreement in the fully asynchronous network model with $t < n/3$ corruptions out of n players. An important challenge in this field is to develop efficient protocols, measured in the total number of bits communicated. We present the first consensus protocol with quasilinear complexity that achieves the following three desirable properties: responsiveness with optimal latency, optimistic fast track, and strong unanimity. To date, *none of these* three properties has been achieved with subquadratic complexity.

Our protocol uses the leader-based framework, which proceeds in phases in which a single player is designated as the leader for that phase. Our key insight is that previous constructions that achieve some of the three properties implicitly require the leader to prove knowledge of certain signed messages from sufficiently many $(2t + 1)$ distinct players. We observe that, up to now, proving this knowledge was done by naively forwarding these messages to all players, resulting in quadratic communication. We formalize these proofs for the first time, with two novel abstractions that we refer to as "proof of recency" (PoR) and "proof of exclusivity" (PoE).

We are able to use recent results on communication-efficient zero-knowledge protocols to *directly* instantiate these proofs with constant size, and as a result, obtain a consensus protocol with quasilinear complexity. Due to the generality of our techniques, we are able to directly improve existing constructions. In particular, we can improve the latency of HotStuff (PODC'19), which is implemented in Facebook's Libra, from 7 to 5 messages before output, and even to 3 in optimistic executions, while preserving a worst-case quasilinear complexity. Our protocol uses standard cryptographic assumptions and only requires a public-key infrastructure, in contrast to the trusted setup required in HotStuff.

Of independent interest, we show that both the PoR and PoE functionalities can also be obtained by interactive protocols, using any threshold signature scheme in a black box manner, while keeping optimal latency and the complexity *linear* in n, at the cost of one of several possible tradeoffs regarding complexity and responsiveness.

Finally we provide *halting in finite time* with both external validity and linear complexity, in the amortized regime over long input values.

1 INTRODUCTION

The problem of consensus is the oldest and most studied task in distributed systems [LSP82; Ben83; DLS88]. The challenge is for n players to agree on some common information, in the presence of an adversary that may cause some of the players to behave arbitrarily. Recently, consensus has gotten a surge of popularity due to the introduction of blockchain [Bad+17; GKR20].

We regard the problem of Byzantine Agreement, where $n = 3t + 1$ players want to agree on some value in the presence of a malicious adversary that adaptively corrupts up to t of them. Each of the players inputs a value, and the goal is for the honest (i.e., non-corrupted) players to output the same value (consistency), while also ensuring that *if* each player has the same input value, then this value will be outputted (weak unanimity). Byzantine Agreement is closely related to the problem of fault-tolerant state machine replication, which is used in decentralized cryptocurrencies and can be seen as an ordered sequence of consensus instances [Lam98; CL99].

We assume the players are connected through an asynchronous network controlled by the adversary, that may inspect all messages, and arbitrarily alter, reroute, drop, delay, or replay them. Our protocols require an established

public-key infrastructure (PKI), and by employing this we may without loss of generality assume the integrity and authenticity of messages, although they may still be dropped or delayed. The challenge of consensus in an asynchronous network is that an honest player may receive messages that cause it to output, but afterwards become isolated from the network. From the point of view of the other players, the player might have been corrupted, so they cannot wait to hear from it to output. The other players have no means of knowing which messages the isolated player received, thus they must find an indirect way to infer its output value.

A common framework to achieve consensus in an asynchronous network is to designate one player as leader, and have each player communicate only with the leader [Lam98; CL99; Gol+18; Yin+19]. If the leader is honest and the network is not blocked, this allows the honest players to efficiently obtain consensus. To deal with the possibility of a corrupt leader, the protocol operates in subsequent *phases* where a new leader is elected in each phase.

In this paper, we focus on three highly desirable properties of consensus protocols, while our objective is to minimize the total number of bits communicated by the players.

Responsiveness with optimal latency. Once the players enter and remain in a phase with an honest leader, the players should output a value as soon as possible. Concretely, each player outputs within the *actual network delay* of 6 consecutive messages exchanged between itself and the leader, once this situation occurs.

Optimistic fast track. Even though an adversary may delay output for an arbitrary amount of time, we do wish for the output to be produced as soon as possible in absence of any adversarial behavior. Concretely, each player outputs within 2 consecutive messages from the start of the protocol if all players are honest and have the same input.

Strong unanimity. Despite all *honest* players having the same input value, with weak unanimity it can happen that the players still output a different value. Therefore we want: if all *honest* players have the same input value, then this value will be outputted.

To date, no protocol has been presented that achieves *any* of these three properties with a subquadratic (in the number of players) number of bits communicated. We detail the state of the art of each of these open questions in Section 1.3.

1.1 Main result

We present the first leader-based consensus protocol that achieves responsiveness with optimal latency, optimistic fast track, and strong unanimity, with a *quasilinear* complexity in the number of players, thus proving the following theorem.

Main Theorem 1. *Consider $n = 3t + 1$ players of which t are malicious, in the leader-based model. Then there exists a consensus protocol that has* responsiveness with optimal latency*, optimistic fast track, and strong unanimity, with a worst-case $O(n \log n)$ total bits sent by all honest players per phase. The protocol requires no trusted setup (beyond PKI) nor distributed key generation.*

To achieve this result, we build upon previous protocols that realize these three properties albeit with quadratic communication complexity. Our key insight that allows us to bring down the required communication, is that these protocols *implicitly* require the leader to prove knowledge of signed messages with certain properties from sufficiently many $(2t + 1)$ distinct players. We observe that, up to now, proving this knowledge was done by naively forwarding these messages to all players, resulting in quadratic communication. We formalize these proofs for the first time, and introduce two novel abstractions.

The first is a "proof of recency" (PoR), where the leader of phase ϕ proves that a specific earlier phase ϕ_{max} is the most recent one in which at least $2t + 1$ players reported they saw a message of a certain form (called a "lock certificate", details will be explained in Section 3). In practice, this translates to providing a combined range proof for $2t + 1$ values, where each player sends a signed message of the most recent phase in which they saw a lock certificate. The second is a "proof of exclusivity" (PoE), in which the leader proves knowledge of $2t + 1$ authenticated values from distinct players, where only one designated value v may occur strictly more than t times.

With the formalization of these proofs in hand, we use recent results from [AC20; ACF20; ACR20] on communication-efficient zero-knowledge protocols to *directly* instantiate these proofs with logarithmic size. Although the short proofs could be instantiated from a trustless circuit-based zero knowledge protocol [Ben+19], this approach has two inefficiencies. First, the large circuit of a hash function leads to both large proofs and a painful implementation process. Second, [Ben+19] gives an unspecified $polylog(n)$ size. In contrast, [ACR20, §7] provides a much shorter and explicit tool for proving knowledge of signed messages whose values satisfy predicates.

There are some hurdles in applying [ACR20, §7] to our context: it allows providing predicates denoted as "R_F", which consist of: knowledge of $k = 2t + 1$ signed messages from distinct signers out of n, such that a public relation of the form: $F(w_1, \ldots, w_n) = 0$ is true, when applied on n values of messages, but of which the $n - k$ values w_i outside of the set of the k signed messages *can be chosen arbitrarily by the prover*. As observed in Section 5.1, this freedom of the prover can be easily get around, to obtain a PoR on the $2t + 1$ actual values signed. A slightly more delicate case is realizing the PoE. In Section 5.2 we achieve this using 3 proofs of the above form R_F, which the prover "binds" together by committing to 3 respective sets of indices of signers, which it proves disjunct.

Due to the generality of our techniques, we are able to directly improve existing constructions. In particular, our protocol improves the latency of HotStuff ([Yin+19], PODC'19), which is implemented in Facebook's Libra, from 7 to 5 messages before output, and even to 3 in optimistic executions, while preserving a worst-case quasilinear complexity. Furthermore, we only use standard cryptographic assumptions and a public-key infrastructure, in contrast to the trusted setup required in HotStuff.

1.2 Extensions and Elementary Alternatives

First, from our main result Main Theorem 1 we first derive several immediate consequences and extensions. (i) we note that the latency gain of our methods compared to [Yin+19] also directly allows us to reduce the latency of the state-of-the-art randomized consensus of [AMS19]. (ii) by amortization of our succinct proofs we are able to achieve $n \log(\log(n))$ amortized complexity over multiple instances. (iii) we provide optimizations in the regime of sequential executions, which is commonly referred to as *state machine replication*.

Second, our main advanced contribution is to provide an alternatives to our direct zero-knowledge construction. Namely, we buid weaker-but-sufficient variants of PoR and PoE from any threshold signature scheme (TSS) in an entirely black box manner, and show how to obtain an efficient consensus protocol still with complexity linear in n. This comes at the cost of either sacrificing responsiveness, or accepting an additional complexity overhead in the bit-length of the inputs.

At first, such alternative constructions seem hard to achieve, since the statements to be proven (PoE and PoR) apply to many messages ($2t + 1$), which are possibly all different, whereas a TSS applies to *identical* messages. We resolve this using interactive protocols. The broad idea is that, upon receiving $2t + 1$ possibly different declarations, the leader "cooks" ad-hoc predicates and *requests* players to send signed statements that their inputs satisfy those predicates. These

further interactions create a new problem of making the latency suboptimal, but we resolve this by making the key observation that leader can safely deliver his proofs *two messages later* than in the baseline protocols from Section 3.

Third, we deal with *halting*: the property that the players can stop sending messages in finite time. So far our protocols guarantee output in finite time, but *not* halting. Halting is achieved in [Nay+20] with linear bit complexity, when amortized over long input values, but at the cost of *sacrificing* the additional desirable "external validity" condition [Cac+01, Def 4.1], which is desirable in the use case of state machine replication. We solve this issue in Appendix E.2.

1.3 Related work

1.3.1 Reponsiveness with optimal latency, linear bit complexity in the worst case and optimal adversary bound. The protocol of [Yin+19] is responsive, but at the cost of a worst case latency of 8 messages delay for an honest leader, instead of the optimal 6.[1] In contrast, the protocol of [AGM18, p. v1] achieves this optimal latency, but it suffers from another limitation. Namely, as pointed out in [Yin+19], that protocol is not responsive: the leader of a new phase is instructed to *wait* for some fixed delay Δ, irrespective of the current network delays, before it can send new messages. Notice that the recent preprint [Jal+21] follow-up of [Yin+19] claims to present a consensus protocol, in the same asynchronous leader-based model, with both optimal latency and responsiveness. However, in their Algorithm 1 (described in their Section V), the leader is instructed to forward to every player the concatenation (denoted *AssQC*) of $n - t = 2t + 1$ signed messages, each message content being denoted "*QC*". These *QC* have a priori *different* values, so they are not amenable to being compactified with a threshold signature. Indeed a threshold signature applies only to *identical messages*. Thus, they have quadratic bit complexity.

1.3.2 Strong unanimity with overall linear complexity. the only known consensus protocol in our model that achieves strong unanimity is [Abr+20a, §6.1 long version]. It has quadratic complexity, which follows from the fact that all players in this protocol are notified of the inputs of $n - t$ players. We describe this issue in a self-contained manner in Section 3.2, where we denote by $\mathcal{D}$ this set of $n - t$ signed inputs.

1.3.3 Fast track with overall linear complexity. All consensus protocols to date with either a fast track, or strong unanimity, have quadratic bit complexity of communications per phase: [Lam02; MA05; Kur02; Kot+09; Cle+09; Aub+15; GV10; Abr+20b; KTZ21]. The question of achieving fast track with subquadratic bit complexity, *even without requiring responsiveness*, was still open up to now. It was raised by the authors of the (nonresponsive) unpublished protocol [AGM18][2]. Let us also point out the fact that enabling a fast track in consensus that is not necessarily of optimal length, also poses challenges: see [Abr+17; SK19a; SK19b] for safety or liveness violations in such published protocols.

Disambiguation: linearity of the fast track of [Gol+18], versus the overall quadratic bit complexity of [Gol+18]. The "SBFT protocol" [Gol+18] has a fast track (although in a suboptimal 3 rounds, instead of the optimal 2). During this fast track, the bit complexity of communications is linear ([Gol+18, §E]). However, from the beginning of the second phase onwards ([Gol+18, §G]), the protocol has communication which is quadratic in the number of players. This is due to the fact that a new leader in [Gol+18] forwards a large set of $n - t$ signed messages to every player, in order to justify safety of his proposed value. We actually observe in Section 3.3 that this bottleneck can be brought back to the issue of

[1] Optimality of this 6 message delay is shown in [Ram20, Theorem A'].

[2] 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; Kot+09; Cle+09; Aub+15; GV10], we can achieve a linear reduction, either by applying LVC [linear phase change], or by using threshold signature schemes (as demonstrated in [Gol+18]). However, combining the two to get linear-over-linear reduction is not obvious."

strong unanimity. A way to see this is: consider as "input bits" the votes (possibly the empty vote) that players cast in the fast track. Then, safety when adding a fast track is equivalent to strong unanimity with respect to these "input bits".

Disambiguation: most fast tracks considered have one more message delay, but still have overall quadratic bit complexity. Most of the protocols cited above are cast in the "state machine replication" setting. There, the inputs are distributed in an *additional* preliminary message, by the possibly malicious leader (or an external "proposer" or "client"). This is why the fast track considered in these papers is weakened since it has one more message delay. However, all of these protocols have the same quadratic complexity bottleneck as soon as the second phase starts.

1.3.4 Transparent setup. Transparent setup means that no prior interaction between players nor trusted dealer of keys (as in [KSM20; Sho00; Rob20]) is required before the protocol starts. Recall by contrast that modern efficient randomized consensus protocols with subquadratic communication either require correctly generated secret keys ([Abr+19; CPS19]) and/or a public random string (e.g., a "genesis block") appearing *after* the keys are published ([Dav+18; CM19; CPS19; CKS20; Blu+20]). The only exception is the recent [BCG20a] in a synchronous setting, which also uses proofs of knowledge of signatures, which are left unspecified (SNARKs with linear extractor), unlike ours which are fully explicit.

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

1.4 Roadmap

We start by defining the network, cryptographic and computational models and the desired properties in Section 2. To describe our protocol, we first need to introduce some existing techniques used by previous protocols. We do so in Section 3 by presenting simplified versions of previous protocols. This presentation leads up to showing how short proofs of knowledge of many different messages can be plugged *directly* in these simplified versions, as a drop-in replacement of the previous solution of naively forwarding these many messages. We carry out this idea in Section 4, where we prove correctness of the protocols obtained. We finish the proof of Main Theorem 1 by showing how to instantiate the proofs in Section 5. In Section 6 we give an overview of our alternative instantiations using merely elementary methods, which are found in more details in the appendices. We then give an overview of other implications and extensions of Main Theorem 1, especially halting in finite time, which are detailed in the appendices.

2 MODEL AND DEFINITIONS

Network. We assume the parties are pairwise connected through a fully asynchronous network controlled by the environment, that can arbitrarily alter, reroute, drop, delay or replay the messages.

Cryptographic model. All parties and the adversary are polynomially-time bounded interactive Turing machines. We assume an established public-key infrastructure, such that each player can register at most one public key to a public bulletin board before the start of the protocol, and where no assumption is made on how the players generate their keys. This plain model, that does not require any additional trusted setup, is referred to as the "bulletin board PKI model" in [BCG20a; TLP20], and it is closely related to the ideal functionality denoted as "$\mathcal{F}_{CA}$" in Canetti [Can04]. We rely on the standard cryptographic hardness assumptions of a bilinear group equipped with a pairing, such that the Decisional Diffie-Hellman assumption holds in both groups. The adversary may adaptively corrupt up to t parties, learning their entire internal state and obtaining full control.

Consensus. The functionality we aim to achieve is the "consensus form" of Byzantine Agreement, which is as follows. Each player is given an input value in some finite range $\mathcal{V} = \{0, \ldots, V\}$ by the environment, and they may at some point in the protocol output some value, also in $\mathcal{V}$. A consensus protocol requires the following two properties to hold:

Consistency. No two honest parties output different values.

Weak unanimity. If all n parties are honest *and* have the same input, then this is the only value that they can possibly output [Lam83].

Weak unanimity has the inherent limitation that, even if all honest parties have the same input value, they may still output another value if the adversary corrupts a player. Therefore, the following stronger property is often desired.

Strong unanimity. If all *honest* players have the same input value, then this is the only possible output.

Leader-based model. Our consensus protocol proceeds in *phases*, that are coordinated by an abstract entity called the *synchronizer* (as in [Nao+19; NK20; BCG20b], also known as the "pacemaker" in [Yin+19]), that is controlled by the environment. The synchronizer maintains a monotonically increasing *phase counter* ϕ, along with the identity of one player L_ϕ per phase that it designates as the leader for that phase. Every time ϕ increases, it sends (ϕ, L_ϕ) to each player using 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.

We now make the definitions of responsiveness and optimal latency from the introduction slightly more precise.

Optimal latency. A consensus protocol has optimal latency if, when the pacemaker remains blocked on the same phase ϕ forever with the leader L_ϕ of this phase being 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 protocol has neither fast track nor strong unanimity, then the latency is required to be 5 messages in the first phase $\phi = 1$ (but still 6 in the higher phases).

Responsive. A distributed protocol is responsive if players have no instruction to wait for some fixed delay at any point in the protocol.

For communication complexity, we count the total number of bits sent by all honest players *in the same phase*, and then take the worst case over all phases and all executions.[3]

3 BASIC PROTOCOLS

In this section, we show some existing consensus protocols in simplified form, to demonstrate the techniques we build upon to achieve our results. Meanwhile, we provide some insight on why no protocol to date was able to achieve our three desired properties with a quasilinear complexity.

3.1 Responsiveness with optimal latency

We start off with a basic consensus protocol that we denote "PBFT" [CL99], that only achieves weak unanimity. PBFT has a quadratic complexity, but it does achieve responsiveness with optimal latency. Modifications to the protocol allow for a linear complexity, at the cost of losing this property. Afterwards, we show how to first add strong unanimity, and then fast track, and see where the quadratic complexities for these properties come from. Looking ahead to the

[3]Under some weaker synchronizers than ours, many players may possibly believe that they are the leader of the same phase ϕ. For the sake of simplicity we do not consider such models.

next section, there we will take the blueprint from these protocols, and replace the parts with quadratic complexity by defining two novel proof abstractions.

PBFT is illustrated in Figures 1 and 2. The protocol distinguishes between the first phase $\phi = 1$ and the later phases $\phi \geq 2$.

We first describe the first phase in Figure 1. Note that we have modified the original PBFT by replacing all-to-all messages by a "star-shaped" communication pattern: each message is either leader-to-all or all-to-leader. This makes sense given that we ultimately want to achieve subquadratic complexity.

The protocol in Figure 1 is *responsive*: players in phase $\phi = 1$ perform these actions *as soon as they can*, and 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 that a threshold signature scheme (TSS), in the strong sense achieved by [ACR20, Theorem 4], provides for any integer k, an algorithm AGGREGATE_k which, 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_{2t+1} on messages of type lock vote, and decision certificate the data type output by AGGREGATE_{2t+1} on messages of type decision vote.

1 Propose Leader L_ϕ multicasts propose(v_L, ϕ) where v_L is his input.

2 Lock vote Every player, upon receiving propose(v, ϕ) from L_ϕ for the first time (for whatever value of v), replies with a signed "lock vote(v, ϕ)".

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

4 Decision vote Every player, upon receiving from L_ϕ a lock certificate(v, ϕ) for the first time (for whatever value of v), replies with a signed decision vote(v, ϕ).

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

6 Output Upon receiving a decision certificate for v, a player outputs v.

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

Let us examine the communication complexity of the first phase. Honest players send a total number of bits equal to $O(n)$ times the size of a threshold signature. The latter can be implemented in constant size assuming a trusted setup ([Sho00; Rob20]), or $\log(n)$ size without ([ACR20]). Therefore, we have quasilinear complexity overall. The star-shaped PBFT that we consider has the same latency of 5 in the first phase as [AGM18], and it is also considered in [Gol+18] under the name "linear PBFT". However, in [Gol+18] they describe only the first phase, and as we see now, the complexity issues instead arise in the higher phases $\phi \geq 2$.

The higher phases of PBFT are described in Figure 2. Informally, PBFT enforces that every player casting a lock vote(v, ϕ), must be "convinced" beforehand that no decision certificate(v_j) for any conflicting $v_j \neq v$ could have been formed in a lower phase $\phi_j < \phi$ (and thus possibly have been 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 exist at least $t + 1$ honest players who received a lock certificate(v_j, ϕ_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 lock certificates.

The simple but strict rule chosen by several previous works (e.g., [AGM18]) is that players lock vote(v, ϕ) only if the leader exhibits to them a lock certificate$(v, \phi_{\max})$, such that they did not receive any lock certificate(v_j, ϕ_j) for a conflicting value v_j in a higher phase $\phi_j > \phi_{\max}$. The drawback is that, mechanically, output is then only guaranteed in phase ϕ if the leader collected the highest seen lock certificate(v_j, ϕ_j) of *all* honest players.[4] This is why the leader must *wait* for the eventual upper bound on the network delay Δ before processing his mailbox. Notice that [Yin+19] has a related rule that achieves responsiveness but not optimal latency, since it requires two extra messages delay.

The rule of PBFT is more permissive, since players have the right to lock vote even if they saw a higher lock certificate for a conflicting value. This is the key to achieving responsiveness with optimal latency. Here, the mechanism is that the leader must append some additional data, that we refer to as *justifications*, to convince players to lock vote a value. The downside of this is that it has *quadratic* bit complexity[5], since these justifications include the set of propose messages from $2t + 1$ players, which are each of size $\Omega(n)$. The higher phases for PBFT are described in Figure 2, with the first phase from Figure 1 taken as a baseline.

0 Report 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 $\phi_i \geq 1$.

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

(i) If $0 < \phi_{\max}$, he received at least one $\mathsf{lc}_{\max} :=$ lock certificate$(v_{\max}, \phi_{\max})$. Then he multicasts propose$(v_{\max}, \phi)$, appended with the justification $\{\mathcal{R}, \mathsf{lc}_{\max}\}$.
(ii) Otherwise, he sets v_L equal to his own input and multicasts propose(v_L, ϕ) appended with the justification $\mathcal{R}$.

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

(i) *Either* the justification comes with some lock certificate(v, ϕ') for v, such that ϕ' is the highest ϕ_j reported in $\mathcal{R}$;
(ii) *Or* the justification contains no lock certificate, and *all* messages in $\mathcal{R}$ report $\phi_j = 0$;

then if the check passes, he replies with a signed lock vote(v, ϕ).

Steps 3-6 are as in Figure 1.

Fig. 2. Star-shaped PBFT: modified steps for the higher phases $\phi \geq 2$. Step **1 Propose** has quadratic bit complexity.

Consider any phase satisfying the definition of optimal latency from Section 2, or more generally, any phase with an honest leader and such that the network allows for 6 consecutive messages to be delivered before the synchronizer issues a new phase. In such a phase, all honest players will output. 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 3 and Proposition 2.

- 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 $\phi_{\max}$; is a proof that strictly less than $t + 1$ honest players did see a lock certificate in a phase ϕ' such that $\phi_{\max} < \phi' \leq \phi$, while being in a phase up to ϕ.

[4] See the discussion in [Yin+19] ("Livelessness with two-phases").
[5] We followed the optimisation (credited to [CL02] in the v2 18 Oct 2018 of [Yin+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.

- 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 $\phi_{\max}$ 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.

3.2 Adding strong unanimity

To obtain strong unanimity, we do not take as a baseline the protocol of [Abr+20a, §6.1 long version]. Although it does achieve strong unanimity, it is constructed on the top of [AGM18], which as we discussed above is incompatible with our goal of responsiveness with optimal latency.

Instead, we are going to achieve strong unanimity by constructing on the top of our baseline consensus protocol (with weak unanimity) of Figure 2. Following [Abr+20a, §6.1 long version], we add to our baseline a **Report** step even when $\phi = 1$, then an additional justification in **1 Propose**, and finally an additional check in **2 Lock vote**.

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

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

(i) If $0 < \phi_{\max}$, then proceed as in Figure 2 under **1 Propose** (i).
(ii) Otherwise, 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(v_L, ϕ) 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 phases ϕ.

Now, the leader also forwards to every player a set of messages $\mathcal{D}$, which itself is of bitsize $\Omega(n)$. This is the quadratic communication bottleneck of strong unanimity, which comes in addition to the previous one of Figures 1 and 2, which was due to the size of $\mathcal{R}$. The check performed in **2 Lock vote** guarantees that honest players do not cast a lock vote for some value v if there exists a conflicting value $v_{\text{un}} \neq v$ which is a unanimous input of all honest players, thus guaranteeing strong unanimity. We shall later formalize this in Lemma 21.

3.3 Adding optimistic fast track

Building on our consensus protocol with strong unanimity of Figure 3, we can obtain a fast track with a tight 2 rounds as follows. Upon receiving messages declare(v) for the same value v from *all* the $n = 3t + 1$ players (denoted "fast quorum" in the folklore), then the leader L_ϕ AGGREGATES$_{3t+1}$ them into a fast dec cert(v) that he multicasts to players. This enriched construction brings no overhead in bit complexity nor latency. To prove the safety of it, note that our baseline construction of Section 3.2 has strong unanimity, and so we only need to prove that: no (isolated) player can output some value v from a fast dec cert(v), while another player outputs a conflicting value v' from a (slow) decision certificate(v'). But a fast dec cert(v) exists only if all $2t + 1$ honest players have the same input v. Thus by strong unanimity of the previous protocol, no decision certificate(v') for a conflicting value can ever be formed. Therefore, we see that having

enforced the strong unanimity previously in Section 3.2, then brings linear complexity of a fast track for free. We shall formalize this later in Lemma 19.

For the sake of completeness, notice that enforcing strong unanimity brings an unavoidable *one extra* message of delay (6 messages total) in the first phase ([Ram20, B']). Despite this, enforcing a plain 2 messages fast track would also result in this extra delay, so this explains the definition of optimal latency in Section 2. However, this extra delay *can* be circumvented by instead directly enforcing an alternative fast track in 3 messages (Definition F.1), as described in Appendix F.2.3 (following [Gol+18]).

4 TWO NEW PROOFS

In this section, we modify the protocol from the previous section by replacing implicit proofs by a drop-in black box proof of knowledge. Concretely, we define two proofs of knowledge. One proves knowledge of the set $\mathcal{R}$, which we call a *proof of recency* (or "PoR"), since the leader wants to prove the report from $\phi_{\max}$ is the most recent. The other proves knowledge of the set $\mathcal{D}$, which we call a *proof of exclusivity* (or "PoE"), where the leader wants to show that v is the only possible value that was declared strictly more than t times.

In the next section, we will show how to instantiate these proofs, but for now we treat them as a black box. We show how to modify the protocols to incorporate these proofs, then prove consistency, in Proposition 2, and strong unanimity, in Proposition 4, of the protocols given these proofs.

4.1 Proof of Recency (PoR)

Let us recall the definition of a *k-threshold signature scheme (TSS)*, as introduced by [ACR20, §5], which implies the classical definition of a TSS. Considering a baseline digital signature scheme, a k-threshold signature is a proof of knowledge of signatures issued by k out of n distinct players, on the same message m. The TSS of [ACR20, §5] proves knowledge of k out of n signatures of the type [Ben04]. Their sizes are in $O(\log(n))$, and it has the property that k can actually be dynamically chosen by the prover. Notice that, in order to instantiate the black box proofs later in Section 4, then we will use the implementation of [ACR20, §7]. There, the baseline signatures are instead of the type known as "structure preserving", such as the ones of [Abe+11].

All the protocols of this section will use a $(2t+1)$-TSS (except the fast track, which also requires a $(3t+1) - TSS$. We use the following data types:

- Let $0 \leq \phi_i \leq \phi$. We have messages of type report(ϕ_i, ϕ). If $0 < \phi_i$, this message is concatenated by a $\mathsf{lc}_i = \mathsf{lock\ certificate}(v_i, \phi_i)$ for some $v_i \in \mathcal{V}$.
- For a value $v \in \mathcal{V}$ and phase number $\phi \in \{1, 2, \dots\}$, we have messages of type propose$(v, \phi, \mathsf{justifs})$, lock vote$(v, \phi)$ and decision vote(v, ϕ). The justifs are additional data, to be made precise below, that are needed when $\phi \geq 2$ to make honest players accept the propose message as valid.
- We take lock certificate(v, ϕ) to be a $(2t+1)$-threshold signature on lock vote(v, ϕ), while decision certificate(v) is a $(2t+1)$-threshold signature on decision vote(v, ϕ) for *any* fixed ϕ.
- Our specialized definition for a *proof of recency* in this context, is a proof of knowledge of the following:

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

In some cases we also refer to the corresponding existential statement: we say that $\phi_{\max}$ satisfies the *recency predicate up to* ϕ, if such a set $\mathcal{R}$ exists. So a $\mathsf{PoR}(\phi_{\max}, \phi)$ proves in particular that $\phi_{\max}$ satisfies the recency predicate up to ϕ. We denote the size of a PoR as $|\mathsf{PoR}|$.

- Finally, the justifs mentioned above are as follows. If $v_{\max} \neq \perp$, then it is a $\mathsf{PoR}(\phi_{\max}, \phi)$ together with $\mathsf{lc}_i = \text{lock certificate}(v_{\max}, \phi_{\max})$, where $\phi_{\max} > 0$. If $v_{\max} = \perp$, then it is a pair $\{\mathsf{PoR}(0, \phi), \perp\}$.

0 Report If $\phi = 1$, then skip directly to step **1 Propose**.
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 $\text{report}(\phi_i, \phi)$, appended with a $\text{lock certificate}(v_i, \phi_i)$ 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 $\text{propose}\big(v, 1, \perp\big)$. Otherwise, upon receiving valid report messages from $2t + 1$ distinct players, then, letting $\phi_{\max}$ be the highest ϕ_i received, builds a $\mathsf{PoR}(\phi_{\max}, \phi)$ out of them.

(i) either $\phi_{\max} > 0$, thus he received at least one $\mathsf{lc}_{\max} := \text{lock certificate}(v_{\max}, \phi_{\max})$. Then he multicasts $\text{propose}\big(v_{\max}, \phi, \{\mathsf{PoR}(\phi_{\max}, \phi), \mathsf{lc}_{\max}\}\big)$;
(ii) or if $\phi_{\max} = 0$, he sets v equal to his own input and multicasts $\text{propose}\big(v, \phi, \{\mathsf{PoR}(0, \phi), \perp\}\big)$.

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

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

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

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

6 Output Upon receiving a decision certificate for v, a player outputs v.

Fig. 4. Star-shaped PBFT with a PoR: any phase ϕ, with communication complexity $O(n|\mathsf{PoR}|)$.

Since an honest leader sends to each player messages containing a constant number of threshold signatures and PoRs, we have that the worst case bit communication complexity per phase is in $O(n)$ times the largest size of a PoR or a threshold signature. Optimal latency follows directly from the ability to build a PoR out of any set $\mathcal{R}$ of $2t + 1$ well-formed report messages.

We now show the protocol of Figure 4 has consistency. As a preliminary remark, note that no two distinct lock certificates relative to the same phase can be formed. Indeed, honest players send at most one lock vote per phase.

Proposition 2. *The protocol of Figure 4 has* consistency.

Proof. Consider an execution in which some value v is output by some player. Thus a $\text{decision certificate}(v)$ has been formed in this execution. Thus there exists a phase ϕ_v such that $2t + 1$ distinct players in this phase signed a $\text{decision vote}(v, \phi_v)$. Without loss of generality, take ϕ_v to be minimal with respect to this condition, and 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 $\text{decision vote}(v', \phi')$. 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 3. □

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

Lemma 3. *Suppose in some execution of the protocol there exists a phase ϕ_v in which $2t+1$ messages* decision vote(v, ϕ_v) *are sent for some value v. Then there does not exist any other value $v' \neq v$ and higher phase $\phi' > \phi_v$ in which some honest player sends a* lock vote(v', ϕ') *for v'.*

Proof. Suppose by contradiction that such a higher phase $\phi' > \phi_v$ exists. Let ϕ' be minimal, i.e. ϕ' is the smallest phase number with $\phi' > \phi_v$ in which there is some honest player that sends a lock vote for a conflicting value $v' \neq v$. This implies that in phase ϕ' this honest player must have received a PoR(ϕ_{lc}, ϕ') with respect to some phase $\phi_{lc} < \phi'$, along with a lock certificate(v', ϕ_{lc}).

Observe that we cannot have $\phi_v < \phi_{lc} (< \phi')$, since then the existence of a lock certificate(v', ϕ_{lc}) would imply that an honest player sent a lock vote for the value $v' \neq v$ in phase $\phi_{lc} > \phi_v$, which contradicts the minimality of ϕ'. We also cannot have $\phi_v = \phi_{lc}$, since the existence of decision vote(v, ϕ_v) in particular implies the existence of a lock certificate(v, ϕ_v), which in turn implies a lock vote from at least $t+1$ honest players for v; but we cannot in the same phase have at least $t+1$ honest players simultaneously voting for a conflicting value $v' \neq v$.

So $\phi_{lc} < \phi_v$. But this also cannot be true, since at least $t+1$ honest players sent a decision vote for v in phase ϕ_v, so they all received a lock certificate for v in phase ϕ_v, which means in a subsequent phase $\phi' > \phi_v$ there cannot exist a PoR(ϕ_{lc}, ϕ'). This contradiction finishes the proof. □

4.2 Adding the Fast Track and Strong Unanimity

We now solve the quadratic bottleneck for strong unanimity. Then, as sketched in Section 3.3 and formalized in Lemma 19, this will automatically give us an optimistic fast track. Our method is to introduce a second proof that we call a *proof of exclusivity* (PoE). Recall from Section 3.2 that each (honest) player is asked to issue signed declare(v) messages only for his own input value v. A proof of exclusivity for $v \in \mathcal{V}$ attests that v satisfies the following *exclusivity predicate*, with respect to some execution of our consensus protocol: there exists a set $\mathcal{D}$ of $2t+1$ declare(v_i) messages issued by distinct players, such that there is no single *other* value $v' \neq v$ which is identically present strictly more than t times in these $2t+1$ declared input values v_i.

If v satisfies the exclusivity predicate, this guarantees that if the honest players have a unanimous input, then this input equals v (see Lemma 21). Our generic definition for PoE(v) is the following proof of knowledge:

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

Therefore, a PoE(v) proves in particular that v satisfies the exclusivity predicate. Given this black box, we only need to add the modifications of Figure 3 to the protocol of Figure 4, where we use PoE(v_L) as a drop-in replacement for sending players the set $\mathcal{D}$. In Figure 5, we show the modifications to the protocol compared to Figure 4.

Proposition 4. *The consensus protocol of Figure 5 satisfies strong unanimity.*

Proof. Suppose by contradiction that all honest players have (unanimously) the same input v_{un}. Consider any value $v' \neq v_{\text{un}}$. We will show show that no lock certificate(v', ϕ') can be formed in the execution (and a fortiori no decision certificate for v').

Without loss of generality, let us consider ϕ' to be the smallest phase in the execution where a lock certificate(v', ϕ') is formed for a value v' different from v_{un}. Thus at least $t+1$ honest players must have sent a lock vote(v', ϕ') in **2 Lock vote** of phase ϕ'.

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

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

(i) If $\phi_{\max} > 0$, proceed as in Figure 4, **1 Propose** under (i).

(ii) Otherwise $\phi_{\max} = 0$, and 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 identically reported in any $t+1$ messages of $\mathcal{D}$. He constructs a PoE(v_L) out of $\mathcal{D}$. Then he multicasts propose(v_L, ϕ) appended with the justification PoE(v_L) (in addition to the justification PoR$(0, \phi)$, as in Figure 4).

2 Lock vote When a player validates the justifications in a received message propose$(v, \phi, \text{justifs})$, if the justifs include a PoR$(0, \phi)$ (so we are in case (ii) of **1 Propose**), then justifs should also include a valid PoE(v) for the value v proposed.
(Then if the checks pass, he replies with a signed lock vote(v, ϕ), as in Figure 4)

Fig. 5. Adding strong unanimity to the protocol of Figure 4, with $O(n|\text{PoE}|)$ complexity.

First, note none of these honest players could have received justifs that include a PoR$(0, \phi')$, since, by the condition in step **2 Lock vote**, it must be the case that they also received a valid PoE(v'). But, existence of such a PoE(v') for a conflicting value $v' \neq v_{\text{un}}$ violates unanimity of v_{un} [in case this fact is not already obvious for the reader, it is formalized in Lemma 21]. Therefore, all these $t+1$ honest players received a PoR(ϕ_{lc}, ϕ') for $0 < \phi_{lc} < \phi'$, together with a lock certificate(v', ϕ_{lc}). We conclude the proof by noting this contradicts the minimality of ϕ'. □

5 IMPLEMENTATION OF PoR AND PoE

From a high level view, we could instantiate the proofs of PoR and PoE by assuming any digital signature scheme, consisting of three algorithms (KEYGEN, SIGN, VERIFY), and assuming that the proofs needed are produced by the trustless general-purpose succinct proof system of [Ben+19] used as black box. But what we actually do in this section, is that we leverage the specific proof scheme of [ACR20, §7], whose comparative advantages are stressed in the introduction.

For convenience we consider that all messages are encoded as signed pairs $m_i = (w, \text{tag}) \in \mathbb{Z}_q \times \mathbb{Z}_q$ for q a prime number (of cryptographic size), which is the same message space as in [ACR20, §7]. In practice the values w encode the (variable) message content, while tag typically encodes the protocol instance number, the type of message etc. Players are meant to send messages to the prover, appended with their signature $\sigma_i := \text{SIGN}(sk_i, (w_i, \text{tag}_i))$ computed from their secret key sk_i. The proofs developed in this section prove knowledge of many (k) such correctly signed messages that have the same tag, such that these signed values $\{w_i\}_i$ (jointly) satisfy some predicate.

5.1 PoR

Let us first generalize Equation (1), that defines a PoR, into a proof of knowledge of k signed messages whose values belong to some publicly-known interval $[a, b]$. This generalization will allow us a similar proof to efficiently implement a PoE. For this latter purpose, we also need that the prover communicates a succinct commitment (of the type used in [ACR20]) $Q_{\mathcal{I}} = \text{COM}(\mathcal{I})$, of the set $\mathcal{I}$ of indices of players that sent the messages for which the prover claims knowledge.

Overall, this gives the following relation. Let $a \leq b \in \mathbb{Z}_q$ and $k \in \{0, 1 \ldots, n\}$. We define:

$$R_{k,[a,b],\text{tag}} := \Big\{ \Big(\textsc{Public Input}: \ k, \text{tag}, \ a \leq b, Q_{\mathcal{I}}; \ \textsc{I Know}: \mathcal{I} \subset \{1, \ldots, n\}, \ \{(w_i, \text{tag}), \sigma_i\}_{i \in \mathcal{I}} \\ \textsc{Such That}: w_i \in [a, b] \quad \text{and} \quad \textsc{verify}\left(\mathsf{pk}_i, (w_i, \text{tag}), \sigma_i\right) = \text{true} \ \text{ and } \ |\mathcal{I}| = k \ \text{ and } \ Q_{\mathcal{I}} = \textsc{Com}(\mathcal{I}) \Big\}. \quad (3)$$

The $\mathsf{PoR}(\phi_{\max}, \phi)$ in the protocol of Figure 4 can then be instantiated as a $R_{k,[a,b],\text{tag}}$ with $k := 2t+1$, where (w_i, tag) encodes a report message in phase ϕ as follows: the (variable) witness w_i is the highest phase ϕ_i in which player i saw a lock certificate; while the public parameters are: $a := 0 \leq b := \phi_{\max}$ and the tag which encodes the phase ϕ of the report message, and various other unspecified metadata that uniquely identify the message, such as an identifier of the protocol instance. Finally, implementation of a succinct proof for $R_{k,[a,b],\text{tag}}$ follows as a particular case of the relation R_F of [ACR20, §7]. This is not directly obvious since R_F deals only with proving relations of the form $F(w_1, \ldots, w_n) = 0$ on n values, of which the $n-k$ values outside of the indices of the k signed messages *can be chosen arbitrarily by the prover*. Nevertheless, we claim that, choosing the specific boolean function $F := \{w_i \in [a, b] \ \ \forall i \in [1, \ldots, n]\}$, induces $R_F = R_{k,[a,b],\text{tag}}$. The reason, tautological, is that R_F implies knowledge of a set $\mathcal{I} \subset [1, \ldots, n]$ of k indices and of messages (w_i, tag) signed by players in $\mathcal{I}$, such that $w_i \in [a, b]$ for all $i \in \mathcal{I}$.

5.2 PoE

We now define PoE in a slightly more specific way:

Definition 5. A proof of exclusivity $\mathsf{PoE}(v)$ of a value v, relative to a publicly-known fixed tag, proves knowledge of one of the two following:

(i) either of a set $\mathcal{I} \subset \{1, \ldots, n\}$ of $t+1$ distinct indices, together with a signature from each player $i \in \mathcal{I}$ for the message (v, tag);

(ii) or, of a set $\mathcal{S}$ of $2t+1$ messages (v_j, tag) for $j = 1, \ldots, 2t+1$, each signed by a distinct player, such that no value v_j repeats identically in strictly more than t messages.

It is straightforward to show that this definition is equivalent to an instantiated form of the previously defined PoE of Equation (2), and we do so in Lemma 20.

We now discuss the implementation. If the prover is in the easy case (i), where some value v repeats $t+1$ times, then he proves this with a $(t+1)$-threshold signature on v. This again can be seen as a proof of knowledge: see the one without trusted setup of [ACR20, §5], that $t+1$ values v_i are equal to the public v.

In what follows we assume that the prover is in the hard case (ii), in which no value repeats more than t times in $\mathcal{S}$. The prover orders the numbers v_i in $\mathcal{S}$ by increasing value, with repetitions, and separates them into three consecutive disjunct nonempty subsets of values (with repetitions): $\mathcal{S} = \mathcal{S}_{\text{low}} \cup \{v_{med}\} \cup \mathcal{S}_{\text{high}}$, all of cardinalities at most t. To construct $\mathcal{S}_{\text{low}}$, he starts from the lowest value (which repeats less than t times by definition), then adds progressively values (with repetitions) in $\mathcal{S}_{\text{low}}$. He stops just before adding the next value, called v_{med} (the median), such that adding v_{med} with repetitions would have led to have strictly more than t values in $\mathcal{S}_{\text{low}}$. Then he constructs the set $\mathcal{S}_{\text{high}}$ containing all remaining values. $\mathcal{S}_{\text{high}}$ cannot be empty, otherwise this would mean that v_{med} repeats at least $t+1$ times.

Let $\mathcal{I}_{\text{low}}$, $\mathcal{I}_{v_{med}}$ and $\mathcal{I}_{\text{high}}$ be the (disjunct) subsets of indices in $[1, \ldots, n]$ of players from which the values in $\mathcal{S}_{\text{low}} \cup \{v_{med}\} \cup \mathcal{S}_{\text{high}}$ originate. They are encoded as binary vectors of length n. Denote $|\mathcal{S}_{\text{low}}|$, $|\{v_{med}\}|$, $\left|\mathcal{S}_{\text{high}}\right|$ the

cardinalities of these sets of players. The prover finally outputs: the public inputs $|\mathcal{S}_{\text{low}}|$, $|\{v_{med}\}|$, $\left|\mathcal{S}_{\text{high}}\right|$, along with succinct commitments to the sets of indices: $Q_{\mathcal{I}_{\text{low}}}$, Q, $\mathcal{I}_{v_{med}}$ and $Q_{\mathcal{I}_{\text{low}}}$. He appends the follwing proofs of knowledge:

1) of binary vectors of length n: $\mathcal{I}_{\text{low}}$, $\mathcal{I}_{v_{med}}$ and $\mathcal{I}_{\text{high}}$, containing exactly $|\mathcal{S}_{\text{low}}|$, $|\{v_{med}\}|$, $\left|\mathcal{S}_{\text{high}}\right|$ coordinates equal to 1, such that these 1 coordinates do not overlap. In particular, this implies disjunctness of the sets encoded by $\mathcal{I}_{\text{low}}$, $\mathcal{I}_{v_{med}}$ and $\mathcal{I}_{\text{high}}$.
2) (i) a proof of $R_{|\mathcal{S}_{\text{low}}|,[0,v_{med}-1]}$ with public input equal to commitment $Q_{\mathcal{I}_{\text{low}}}$, (ii) a proof of $R_{|\mathcal{S}_{\text{high}}|,[v_{med}+1,q-1]}$ with public input equal to commitment $Q_{\mathcal{I}_{\text{high}}}$, (iii) and a $|\{v_{med}\}|$-threshold signature on v_{med}

Proposition 6. *This* output *is a* PoE(v).

Proof. We already addressed the easy case, thus we consider only the output of the hard case, as described just above. The data provided by the prover implies knowledge of some set $\mathcal{S}'$ of $2t+1$ pairs (v_i, tag) signed by distinct players, such that the values v_i are organized in three nonempty subsets which are disjunct, each of cardinality at most t. Thus no value v_i repeats identically more than t times in $\mathcal{S}'$. □

6 ELEMENTARY ALTERNATIVES AND ADVANCED CONTRIBUTIONS

We now show some of the direct implications of Main Theorem 1, and discuss some extensions to the theorem. This section is intended to give a high-level overview of our more advanced results. Most details can be found in the appendices.

6.1 Re-building our Program from Threshold Signatures Only

The use of succinct arguments of knowledge, that we used in the proof of Main Theorem 1, is new in the consensus literature. Thus, we now want to demonstrate that Main Theorem 1 can be alternatively obtained, although less efficiently, from solely threshold signatures (TSS), which are a mainstream tool in the area. We consider an arbitrary threshold signature scheme (TSS), instantiated twice (with threshold $t+1$ for one instantiation and threshold $2t+1$ for the other). Let us denote by |TSS| the size of a threshold signature, and by V the size of the domain of inputs for the TSS. For instance, the TSS without trusted setup of [ACR20, Thm 4] has $|\text{TSS}| = O(\log n)$ size.

In the theorem 7, apart from |TSS| which may be not constant, the other coefficients of the terms in n are constant and not specified. These coefficients are clear from the protocols detailed in the appendices. Typically, they are equal to the number of steps (6), times the size of a message content, e.g., such as the short strings of characters specified in the PoR protocol of §D. Instantiating with optimized elementary interactive protocols for PoR and PoE, of independent interest, introduced below in Sections 6.1.1 and 6.1.4, we have:

Theorem 7. *In the leader-based consensus model with $n = 3t+1$ players of which t are malicious, there exists a consensus protocol that has responsiveness with optimal latency, optimistic fast track, and strong unanimity, with a worst-case $O(n\phi + n \log V\,|\text{TSS}|)$ total bits sent by all honest players in any given phase ϕ. If responsiveness is not required, the bit complexity per phase reduces to $O(7n\,|\text{TSS}|)$.*

6.1.1 First ingredient: Preserving Optimal Latency Despite Delaying the Proofs. Recall in Figure 3 we showed how to add strong unanimity to our star-shaped PBFT protocol with quadratic complexity. Here, the leader is required to forward the set $\mathcal{D}$ of messages in **1 Propose**, immediately after receiving it. To obtain linear complexity and achieve Main Theorem 1, we replaced forwarding $\mathcal{D}$ by sending a short proof (PoE) for the value v_L that he proposes. Therefore, the leader does not have the time to perform further interaction with players before issuing this proof in step **1 Propose**.

Interestingly, a tweak on the baseline protocol of Figure 3, allows the leader to use 2 more messages delay to issue this PoE, now in step **3 Lock certificate**. We describe this tweak in Figure 6. Note that this tweak does require the leader to already in **1 Propose** "commit" to the same value v_L for which he is going to deliver a proof later in **3 Lock certificate**.

Likewise, for the case of Theorem 7 where responsiveness is not required, we start this time from the baseline consensus protocol of [AGM18], which we recall in Appendix B.3.3. Then we apply the same tweak on it, enabling the leader to issue a PoE two steps later: see Figure 8.

6.1.2 Second ingredient: weakening of Definition of PoE. To show the constructions that prove Theorem 7, we now leverage the extra delays obtained previously. Precisely, we want to construct interactive sub-protocols with low communication complexity, that enable the leader to output the required proofs (PoE and PoR) in a total maximum of 2 round trips. To enable efficient elementary sub-protocols, we will weaken the definition of a PoE(v) (from Equation (2)) into Definition 10). 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.

6.1.3 Third ingredient: interactive sub-protocols producing PoE *and* PoR *of short sizes.* For the case of Theorem 7 where responsiveness is not required, the sub-protocol for PoE is presented in Appendix C.2. The PoE output by the leader has size 5 |TSS|, which, combined with the 2 |TSS| bits of certificates sent by the leader to each player in a phase, results in the stated complexity of $7n|TSS|$. Interestingly, the baseline of [AGM18] that we chose in Section 6.1.1 does not require a PoR and still having Optimal Latency. This is what enables this sharp communication complexity.

For the case of Theorem 7 where responsiveness is required, the responsive interactive sub-protocol is presented in Appendix C.1. The PoE output has size $\log V$ |TSS| ($\mathcal{V}$: the range of possible input values). Likewise, for usage in the responsive consensus protocol, we provide a responsive elementary sub-protocol in Proposition 17 in Appendix D, that enables the leader to output a PoR in one message delay. This sub-protocol requires players to send $O(n\phi)$ bits in total, and the bitsize of the PoR output by the leader is in $O(|\text{TSS}|)$.

All those elementary constructions of PoE also handle External Validity, whose Definition is recalled in Section 6.1.4.

6.1.4 Extension: Handling External Validity. In Appendices B and C we discuss how to extend techniques to a consensus protocol with *external validity conditions* (Definition 8, following [Cac+01, Def 4.1]) on the input and output values. These conditions come into play for example when using consensus protocols to provide a functionality called "state machine replication" (cf. [Lam98; CL99]). Namely, it is required that the output value is, in addition, appended with some additional information, denoted as "certificate of validity" and noted π, such that π is returned as valid be some efficiently computable public algorithm, denoted as Q_{ID}. Concretely, these "certificates of validity" can be, e.g., a proof of work on the value, or the signature of some entity, denoted as "client", on the value. Certificates can be delivered to players by the Environment at any point in the protocol, and can be forwarded by players to each other. In the External Validity setting, some players may not start with a valid certificate on their input value, if not start with no input at all. Optimal Latency of leader-based consensus is then narrowed (Definition 9) to phases where: either a honest leader knows a valid value, or at least $t + 1$ honest players have an input with a valid certificate (so the leader is guaranteed to be declared at least one of them). We handle this extra difficulty in Appendix B and Appendix C.

As a slight variation on these specifications of External Validity, notice that, in some use-cases (e.g. [CL99]), one could want to guarantee output in finite time, even in situations where strictly less than $t + 1$ honest players would have

a valid certificate on their input. We formalize in Lemma 23 a possible add-on to protocols, without communication overhead, that guarantees the output of a default value in these cases, that does not come with a valid certificate (thus this violates External Validity).

6.2 Halting in Finite Time *and* External Validity, with Amortized Linear Complexity Over Long Values

Halting in Finite Time with Amortization over Long Values, by [Nay+20]. 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 compiler introduced [Nay+20, Figure 4] transforms any consensus with output in finite time, into one Halting in finite time and a communication complexity which is amortized when ℓ is large. Namely, if the baseline consensus has linear complexity, then the compiled one has $n^2O(1) + n\ell$ complexity. 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 in the situation where *not 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 [Nay+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* enforce the 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 [Cac+01, 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 9. 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 [Nay+20] that we provide in Appendix E.2.

6.3 Direct Consequences and Extensions of the Main Result

6.3.1 Reducing the Latency of the state of the art Leaderless Asynchronous *Validated Consensus ("VABA") of [AMS19].* The protocol of [AMS19] consists in running n instances of a variation of the consensus of [Yin+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 Main Theorem 1 can be used as a drop-in replacement for [Yin+19]. This is detailed in E.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.

6.3.2 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 Main Theorem 1 to this naive approach in M instances of PoR in parallel, would result in a total $O(Mn\log(n))$ bits sent. However, the proof system of [ACR20] that we use, being in logarithmic size in the circuit to be proven, thus 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.

6.3.3 Other Trivial Consequences. In Appendix F.2 we consider direct application of our short proofs to linearize complexity of most of the protocols considered in 1.3.3. These consensus, presented under the name"state machine replication", offer degraded properties. The degraded properties are: no strong unanimity and a weakened fast track (Definition F.1), of 3 messages delays instead of 2. This degradation, which thus escapes the impossibility [Ram20, B'], enables in return latency of 5 in the first phase instead of 6. We illustrate how short PoR/PoE also apply and make these protocols communication-linear. These applications also carry in a straightforward manner over the protocols with suboptimal adversary bounds considered in [DGV05; Gol+18], where $n = 3t + 2c + 1$ for $c \geq 1$. Notice the nice recent improvement $n = 3t + 2c - 1$ in [KTZ21], which however applies only to weakened fast tracks with one more message delay. Recall that they enable the benefit, in the above degraded context, to enable a fast track even if c players are corrupted (whereas $c = 0$, for the optimal adversary bound $n = 3t + 1$ considered elsewhere in this paper).

We also briefly review how the technique denoted as "pipelining", introduced in [Yin+19], applies. Namely, in the regime of multiplie ordered sequences (denoted as "state machine replication"), this consists in having the same leader manage simultaneously two instances of consensus: the last steps of the former, and the first steps of the latter. This multiplies throughput by 2, but at the cost of making both instances fail if the leader is corrupted.

REFERENCES

[Abe+11] Masayuki Abe et al. "Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups". In: *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 649–666.

[Abr+17] Ittai Abraham et al. "Revisiting Fast Practical Byzantine Fault Tolerance". In: *CoRR* abs/1712.01367 (2017).

[Abr+19] Ittai Abraham et al. "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*. ACM, 2019, pp. 317–326.

[Abr+20a] Ittai Abraham et al. *Communication Complexity of Byzantine Agreement, Revisited*. 2020. arXiv: 1805.03391 [cs.DC].

[Abr+20b] Ittai Abraham et al. "On the Optimality of Optimistic Responsiveness". In: 2020.

[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*. Vol. 12172. Lecture Notes in Computer Science. Springer, 2020, pp. 513–543.

[ACF20] Thomas Attema, Ronald Cramer, and Serge Fehr. "Compressing Proofs of k-Out-Of-n Partial Knowledge". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 753.

[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. https://eprint.iacr.org/2020/1447. 2020.

[AGM18] Ittai Abraham, Guy Gueta, and Dahlia Malkhi. "Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil". In: *CoRR* abs/1803.05069 (2018). https://eprint.iacr.org/2020/406.

[AMS19] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. "Asymptotically Optimal Validated Asynchronous Byzantine Agreement". In: *PODC*. ACM, 2019, pp. 337–346.

[Aub+15] Pierre-Louis Aublin et al. "The Next 700 BFT Protocols". In: *ACM Trans. Comput. Syst.* 32.4 (2015), 12:1–12:45.

[Bad+17] Christian Badertscher et al. "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*. Springer, 2017, pp. 324–356.

[BCG20a] Elette Boyle, Ran Cohen, and Aarushi Goel. "Breaking the $O(\sqrt{n})$-Bits Barrier: Balanced Byzantine Agreement with Polylog Bits Per-Party". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 130.

[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*. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 23:1–23:17.

[Ben+19] Eli Ben-Sasson et al. "Scalable Zero Knowledge with No Trusted Setup". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Springer, 2019, pp. 701–732. DOI: 10.1007/978-3-030-26954-8_23. URL: https://doi.org/10.1007/978-3-030-26954-8%5C_23.

[Ben04] Dan Boneh and Ben Lynn and Hovav Shacham. "Short Signatures from the Weil Pairing". In: *J. Cryptology* 17 (2004), pp. 297–319.

[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*. ACM, 1983, pp. 27–30.

[BG17] Vitalik Buterin and Virgil Griffith. "Casper the Friendly Finality Gadget". In: *CoRR* abs/1710.09437 (2017).

[Blu+20] Erica Blum et al. "Asynchronous Byzantine Agreement with Subquadratic Communication". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 851.

[Cac+01] Christian Cachin et al. "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*. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 524–541.

[Can04] Ran Canetti. "Universally Composable Signature, Certification, and Authentication". In: *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*. IEEE Computer Society, 2004, p. 219.

[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*. ACM, 2020, pp. 175–177.

[CL02] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance and Proactive Recovery". In: *ACM Trans. Comput. Syst.* 20 (Nov. 2002), pp. 398–461.

[CL99] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. OSDI '99. New Orleans, Louisiana, USA: USENIX Association, 1999.

[Cle+09] Allen Clement et al. "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*. ACM, 2009, pp. 277–290.

[CM19] Jing Chen and Silvio Micali. "Algorand: A secure and efficient distributed ledger". In: *Theor. Comput. Sci.* 777 (2019), pp. 155–183.

[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*. Vol. 11476. Lecture Notes in Computer Science. Springer, 2019, pp. 720–749.

[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*. ACM, 2020, pp. 1–11.

[Dav+18] Bernardo David et al. "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*. Vol. 10821. Lecture Notes in Computer Science. Springer, 2018, pp. 66–98.

[DGV05] Partha Dutta, Rachid Guerraoui, and Marko Vukolić. *Best-case complexity of asynchronous Byzantine consensus*. Tech. rep. 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". In: *J. ACM* (1988).

[GKR20] Peter Gazi, Aggelos Kiayias, and Alexander Russell. "Tight Consistency Bounds for Bitcoin". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020.

[Gol+18] Guy Golan-Gueta et al. "SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains". In: *CoRR* abs/1804.01626 (2018). arXiv: 1804.01626. URL: http://arxiv.org/abs/1804.01626.

[GV10] Rachid Guerraoui and Marko Vukolic. "Refined quorum systems". In: *Distributed Comput.* 23.1 (2010), pp. 1–42.

[Jal+21] Mohammad M. Jalalzai et al. *Fast-HotStuff: A Fast and Resilient HotStuff Protocol*. 2021. arXiv: 2010.11454 [cs.DC].

[Kot+09] Ramakrishna Kotla et al. "Zyzzyva: Speculative Byzantine fault tolerance". In: *ACM Trans. Comput. Syst.* 27.4 (2009), 7:1–7:39.

[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.

[KTZ21] Petr Kuznetsov, Andrei Tonkikh, and Yan X Zhang. *Revisiting Optimal Resilience of Fast Byzantine Consensus*. 2021. arXiv: 2102.12825 [cs.DC].

[Kur02] Klaus Kursawe. "Optimistic Byzantine Agreement". In: *21st Symposium on Reliable Distributed Systems (SRDS 2002), 13-16 October 2002, Osaka, Japan*. IEEE Computer Society, 2002, pp. 262–267.

[Lam02] Leslie Lamport. *Fast byzantine paxos*. US7620680B1 patent. 2002.

[Lam83] Leslie Lamport. “The Weak Byzantine Generals Problem”. In: *J. ACM* (1983).

[Lam98] Leslie Lamport. “The Part-time Parliament”. In: *ACM Trans. Comput. Syst.* 16.2 (May 1998), pp. 133–169. ISSN: 0734-2071.

[Lib19] Libra Team. *State Machine Replication in the LibraBlockchain*. https://developers.libra.org/docs/assets/papers/libra-consensus-state-machine-replication-in-the-libra-blockchain/2019-10-24.pdf. 2019.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. “The Byzantine Generals Problem”. In: *ACM Trans. Program. Lang. Syst.* (1982).

[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*. IEEE Computer Society, 2005, pp. 402–411.

[MR20] Atsuki Momose and Ling Ren. *Optimal Communication Complexity of Byzantine Consensus under Honest Majority*. 2020. eprint: 2007.13175.

[Nao+19] Oded Naor et al. “Cogsworth: Byzantine View Synchronization”. In: *CoRR* abs/1909.05204 (2019).

[Nay+20] Kartik Nayak et al. “Improved Extension Protocols for Byzantine Broadcast and Agreement”. In: *DISC*. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 28:1–28:17.

[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*. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 26:1–26:17.

[Ram20] Matthieu Rambaud. *The latency costs of Optimistically fast output and of Strong unanimity, in authenticated leader-based Byzantine consensus under partial synchrony*. Tech. rep. https://perso.telecom-paristech.fr/rambaud/articles/nofast.pdf. 2020.

[Rob20] Alin Tomescu and Robert Chen and Yiming Zheng and Ittai Abraham and Benny Pinkas and Guy Golan-Gueta and Srinivas Devadas. “Towards Scalable Threshold Cryptosystems”. In: *IEEE Symposium on Security and Privacy*. IEEE, 2020, pp. 877–893.

[Sho00] Victor Shoup. “Practical Threshold Signatures”. In: *EUROCRYPT*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 207–220.

[SK19a] Nibesh Shrestha and Mohan Kumar. “Revisiting EZBFT: A Decentralized Byzantine Fault Tolerant Protocol with Speculation”. In: *CoRR* abs/1909.03990 (2019).

[SK19b] Nibesh Shrestha and Mohan Kumar. “Revisiting hBFT: Speculative Byzantine Fault Tolerance with Minimum Cost”. In: *CoRR* abs/1902.08505 (2019).

[TLP20] Georgios Tsimos, Julian Loss, and Charalampos Papamanthou. “Nearly Quadratic Broadcast Without Trusted Setup Under Dishonest Majority”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020).

[Yin+19] Maofan Yin et al. “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*. ACM, 2019, pp. 347–356.

A MODEL: REMINDERS AND COMPLEMENTS

A.1 Messages lost or not, output vs Halting

For Simplicity: Messages are Always Delivered. Recall that the definiton of optimal latency from Section 2 assumes for simplicity an asynchronous network where messages *cannot* be lost. Main Theorem 1 and Theorem 7 (the proof of responsiveness) are stated with respect to this definition.

Adapting The Model To Account For Lost Messages. We could alternatively have supposed that messages sent can be lost, and restrict the definition of optimal latency to phases where no message is lost. The proofs of Main Theorem 1 and Theorem 7 hold unchanged in such an alternative model.

Halting. In the definition of optimal latency, we only consider latency 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 1 nor Theorem 7. However we *will* consider halting in Appendix E.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].

A.2 Partial Synchrony

For the specific case of the last nonresponsive claim of Theorem 7, on nonresponsive protocols (supported by the protocols Appendix B.3.3 and Appendix C.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 Appendix B.3.3 and Appendix C.2) hold even under the specific model [DLS88, §3.1] where messages can be lost before GST. Thus also our nonresponsive claim of Theorem 7.

Except if halting is required. However, to apply the amortized halting techniques of Appendix E.2, then, as mentionned in A.1, one needs to assume in addition that messages are never lost, such as in the model [DLS88, 2.3 3)].

B HANDLING EXTERNAL VALIDITY AND PoE WITH INTERACTIONS

B.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 [Cac+01, Def 4.1] or as in [AMS19, Lemma 23 of full paper]. 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 of Consensus, as provided in §2, in order to handle External Validity conditions on the output:

Definition 8 (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 10 how tolerating this loosening enables in return *better liveness in protocols*, i.e., that guarantees output even if no honest player has a valid input. By contrast, if one wanted to strictly enforce External Validity without this loosening, then Consensus protocols may be simply modified by requiring players to ignore any message containing a value not valid. This is formalized by the easy Lemma 22. 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 B.1. The behavior of Q_{ID} is arbitrary, it may possibly issue certificates only for a subset of input values. The Lemma 23 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 9 (Optimal Latency). The additional restriction is made to the definition of optimal latency in §2 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 ([Cac+01, 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 23 brings a mechanism for output in any case, which matches the optimal latency criterion specified in §2. This mechanism works *whatever the inputs*. In return, it comes at the cost of possibly violating External Validity in the situation where t honest players or less have a valid input.

B.2 Useful Weakening of Exclusivity Predicate. Extension with the Special Nonvalid Value

For our Elementary constructions in Appendix C, 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 21. 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 4. 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 10 (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 10, in link External Validity (Definition 8). Definition 10 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 10 is the key to achieving better liveness in our protocols (Definition 9), than demanded traditionally ([Cac+01, Def 4.1] or as in [AMS19, Lemma 23 of full paper]). Notice that this new Definition 10 *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 8.

Lemma 11. *(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 10)*

(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. □

B.3 Handling for *Interactive* 4-move PoE in the Consensus of Main Theorem 1

B.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 12. 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 $\mathsf{PoE}(v)$ in the sense of Definition 10.

We call a 4*-move interactive Not Responsive* PoE *protocol*, in the model [DLS88, p. 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 $\mathsf{PoE}(v)$, satifies the Exclusivity Predicate (Definition 10).

B.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 19) in Main Theorem 1, 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 12), 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 12). 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 12).

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).

Sketch proof of this alternative enrichening. Responsiveness with Optimal Latency. If in case **1**(ii) then it is guaranteed by Definition 12 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 PoR 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 4. 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 10, 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 10 by Lemma 21.]

B.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, p. 3.1] with parameter Δ.

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 12 is indeed guaranteed to select in **1**, and to issue a PoE in **3**.

0 Report 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 full lock certificate(v_i, ϕ_i) if $\phi_i \geq 1$. He *Initiates* an instance of a 4-move PoE (Definition 12) 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 $\mathsf{flc}_{max} :=$ full lock certificate(v_{max}, ϕ_{max}). Then he multicasts propose$\big(v_{max}, \phi, \{\mathsf{PoR}(\phi_{max}, \phi), \mathsf{flc}_{max}\}\big)$;
(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 propose$\big(v_{select}, \phi, \{\mathsf{PoR}(0, \phi), \perp\}\big)$; *else* does nothing.

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

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 PoE(v);
(ii) *Else* this means that he selected v with respect to the PoE protocol, in **1**. Thus, by Definition 12, he now obtains a PoE(v).

He AGGREGATES$_{2t+1}$ the 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 Every player, upon receiving from L_ϕ a full lock certificate(v, ϕ) for the first time in ϕ (whatever v), replies with a signed decision vote(v, ϕ).

5 Decision certificate Leader L_ϕ, upon receiving decision vote(v, ϕ) from $2t+1$ distinct players for the same value v, issues from them a 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

C ELEMENTARY INTERACTIVE PoE

C.1 An elementary interactive Responsive PoE of bitsize $O(\log(V)|TSS|)$

C.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, send *both* testifies when they have input 0.

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

(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 11 (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.

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$_{2t+1}$ 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]

C.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 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$_{t+1}$ those reports. This constitutes a PoE(v) by Lemma 11 (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 11(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 $i \in [1 \ldots \log V]$, the i-th bit b_i of w is the one which received the majority (so $\geq t + 1$) of testifies. He AGGREGATES$_{t+1}$ each of those majority declarations, with τ, to obtain $\log V$ threshold signatures: one for every bit b_i of w. *Claim2*: by Lemma 11(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 11(ii) shows that this then constitutes a PoE(*any*).

0 Report Every player:

(i) *Initiates* an instance of a (nonresponsive) 4-move PoE (Definition 12) 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 12, he now obtains a PoE(v) from this protocol.

He AGGREGATES$_{2t+1}$ the 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, ϕ), $PoE(v)$) 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$_{2t+1}$ 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

(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$_{t+1}$ the signatures of these negative answers. By Lemma 11(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 13. *This is a* 4*-move interactive* PoE *protocol (Definition 12), 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_{t+1} signatures on bits, plus one AGGREGATED_{t+1} signature of players declaring not to have input w.

Let us now prove that the protocol satisfies Definition 12. 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_{t+1} signature testifies that b_i was present in the input of at least $t+1$ 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. □

C.2 A three-move PoE under partial synchrony with size in $5|TSS|$

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

Theorem 14. *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 11. 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 15 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 $\text{PoE}(a_j)$.

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_{t+1} those reports. This constitutes a $\text{PoE}(v)$ by Lemma 11 (ii).

- (α) *either* v is valid, in which case he outputs v and the $\text{PoE}(v)$;
- (β) *or* v is zero, in which case the $\text{PoE}(v)$ constitutes a $\text{PoE}(any)$ by Lemma 11(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 $\text{PoE}(any)$.

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 $t + 1$. [6] More details on the construction are given in Appendix C.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 11. [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 14. 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 15. *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

$$|\overline{I_j}| = 3t + 1 - m - |I_j| \tag{4}$$

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

$$d_j \geq |\overline{I_j}| - t = 2t + 1 - m - |I_j| \tag{5}$$

[6] 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**.

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

$$t \geq d_j + m \tag{6}$$

Combining with the previous inequality, we get

$$t \geq d_j + m \geq 3t + 1 - |I_j| - t = 2t + 1 - |I_j| \tag{7}$$

and thus

$$|I_j| \geq t + 1 \tag{8}$$

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

C.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. $|I_1| = t'$, $I_2 = [a_2]$ repeated $t' + 1$ times,$|I_3| = t'$, $I_4 = [a_4]$ repeated $t' + 1$ times and $|I_5| = t - 1$.

D A PoR WITH SIZE $|TSS|$, PRODUCED WITH $O(n\phi)$ COMMUNICATION

As in Appendix B.2, we make here a weakened definition of PoR, compared to the Equation (1). This weakened definition will allow elementary constructions, and still, it exactly addresses the property needed in the proof of Lemma 3 (ii). Precisely, this Lemma (ii) it is the only place where PoR is needed in the proof of Main Theorem 1, where the actually property of a PoR(ϕ_{max}, ϕ) which is used, is namely that a PoR 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 16 (*PoR*).* 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 Recency*: PoR(ϕ_{max}, ϕ) 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 PoR 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 PoR(ϕ_{max}, ϕ).

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$_{2t+1}$ the testimonies (with σ). *Claim2*: this constitutes a PoR on (ϕ_{max}, ϕ).

PROPOSITION 17. *The previous protocol is a* PoR *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* PoR *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 PoR 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 $\phi_{max} = 0$). □

E OTHER CONTRIBUTIONS

E.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 [Yin+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 [Yin+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 [Yin+19] and repeat. *Plugging* our Main Theorem 1 in place of [Yin+19] reduces the latency of the first phase from 7 to 5 (and of the subsequent phases from 8 to 6).

E.2 Reconciling Halting in Finite time with no Amortized Overhead, and External Validity

The modifications of [Nay+20, Figure 4] are: players perform as baseline a leader-based consensus with linear complexity, e.g. the one of Section 3.1, possibly enriched with Strong Unanimity and a Fast Track as in Section 3.1. 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 22 for a formalization). The message complexity of Main Theorem 1 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 18. *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 [Nay+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 [Nay+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 [Nay+20] consider more generic proof systems for this purpose, denoted as "cryptographic accumulators".]

F EASY LEMMAS AND VARIOUS EXTENSIONS

F.1 Lemmas

F.1.1 Deriving a Fast Track from Strong Unanimity for Free. The following Lemma formalizes the trick used in 3.3 to compile Strong Unanimity into a Fast Track with no cost.

Lemma 19. *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_t *he 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.

F.1.2 PoE *implies Exclusivity.*

Lemma 20. *Specify Definition 5 of* PoE *as follows: $\mathcal{S} := \mathcal{D}$,* tag *equal to the metadata: the protocol instance number and the type of message (here:"*declare*"); and $v := v_L$. It is then equivalent to Equation* (2).

Proof. Starting from Equation (2), the Definition 5 holds trivially (either v repeats $t + 1$ times in $\mathcal{D}$, or it does not, in any case $\mathcal{S} := \mathcal{D}$ works). Starting now from Definition 5. In the first alternative, if the prover both knows a set $\mathcal{S}$ of $2t + 1$ signed reported values, and also knows a value v signed $t + 1$ times or more, then he is a fortiori able to exhibit a set $\mathcal{D}$ of $2t + 1$ signed values in which v is the only value repeated $t + 1$ times. In the second alternative, taking $\mathcal{D} := \mathcal{S}$, any value v in this set will satisfy Equation (2). □

F.1.3 Exclusivity Implies that No Other Value Can Be Unanimous.

Lemma 21. *Suppose that 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, as defined in §4.2. In particular no* 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'. □

F.1.4 Enforcing External Validity. Consider as in [Cac+01, Def 4.1]: an external entity Q_{ID}, that may deliver to some honest players "certificates of validity" to their inputs.

Lemma 22. *Then all the protocols in this paper can be modified without any additional cost on the 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 [Cac+01, 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.]*

In addition, (optimal) latency is preserved in executions in which every honest player has a valid input.

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 always appends the validity certicate in the lock certificates and decision certificates that he makes.

F.1.5 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 23. *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$_t$ *hem into a threshold signature and multicasts* propose$(\perp, \phi)$ *along with a* PoR$(0, \phi)$*.*

2 *a player accepts* propose$(\perp, \phi)$ *if: it comes with a valid threshold signature, and, if the* PoR$(0, \phi)$ *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.

□

F.2 Other Various Extensions

F.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 c 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 [Gol+18; DGV05; Aub+15]. The constructions of PoE and PoR (generic, or with the alternative weaker Definitions 10 & 16) carry unchanged over these new parameters.

F.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 (§3.1) of Main Theorem 1 with plain Responsiveness and Optimal Latency in the transaction system [Lib19], in place of the consensus of [Yin+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 1 with the Transparent TSS of [ACR20], also removes this limitation, since it does not use any trusted setup.

F.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 F.1 is satisfied by [Gol+18], although we define it for simplicity: in the maximal adversary threshold (see F.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 the definition of consensus (§2, or, alternatively, the Consistency and Consensus Weak Unanimity combined with External Validity, as formalized in Definition 8) But is *not* compatible with strong unanimity (as defined in §2, or alternatively, the definition combined with External Validity, in Definition 8). It also comes at the top of the top of the definition of optimal latency (§2, although now with the stronger requirement where 5 messages delays are in addition required in the first phase).

Definition F.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 (§A.2, [DLS88, §3.1]), this holds only if the network is initially synchronous. The reason for requiring here 5 messages delays in the first phase despite this weakened fast track, instead of always 6 as defined in §2, is that a weakened fast track in 3 messages is not subject to the impossibility of [Ram20, B']. Notice also the Dictatorial Fast tracks in 2 messages of, e.g., [Aub+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 [Gol+18], with succinct proofs used as drop-in replacement to forwarding of many messages. Players i who issued a 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 fast dec cert$(v, 1)$ for a value v (only relatively to phase $\phi = 1$) is the AGGREGATE$_o$f $3t + 1$ identical 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 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 $2t + 1$ 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 PoE(v_L) out of $\mathcal{D}$. Then he multicasts propose(v_L, ϕ) appended with the justification PoE(v_L) (in addition to the justification PoR$(0, \phi)$, 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 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)

3 Lock/Fast certificate *if* $\phi = 1$*, In addition:* Leader L_1, upon receiving $3t + 1$ lock vote$(v, 1)$ for the same value v AGGREGATES$_t$hem into a 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 fast dec cert$(v, 1)$, output v.

Fig. 9. Adding 3-steps Fast Track in $O(n|\text{PoE}|)$ 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; Yin+19; CS20]. Recall that this consists in having the leader at step **3** of instance i, upon forming a lock certificate$(v_i, .)$, start simultaneously an instance $i + 1$. The propose(v_{i+1}) of this new instance may then be appended with the lock certificate$(v_i, .)$. Players then accept a propose in instance $i + 1$ only if it comes appended with a 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 of bit complexity, as stated in §2, would need to be changed accordingly, e.g., by dividing the bitsize by the number of honest leaders in the same phase).