Lower bounds for authenticated randomized Byzantine consensus under (partial) synchrony

the limits of standalone digital signatures

Abstract

This paper deals with lower bounds for *randomized* consensus against any *standard* adaptive Byzantine Environment, that is, which can corrupt players at any time, up to a total of t, but *not* delete messages sent. We assume that players can *sign* their messages, so that the Environment cannot forge a message on behalf of a honest player. This assumption is known as the "authenticated setting" and studied e.g. in the classical works of Pease et al, DLS, the lower bounds of Dolev-Strong/Reischuk, Hadzilacos-Halpern etc. We rename it instead "standalone digital signatures", not to confuse it with the weaker model of plain authenticated channels. We do not make any further assumption. By contrast, recall that Chen-Micali (Algorand) and Ouroboros Praos (EC'18) assume the ability to publish public keys on a board (registered PKI), plus a global coin (a.k.a. CRS: the hash of the previous block, or assuming a random value contained in the genesis block); alternatively: Abraham et al (PODC'19) assume a trusted PKI setup. We show that:

- Under all partially synchronous models, especially those where messages are *never* lost, for $n \geq 3t+1$ players. Then (Theorem 4) we have that, with nonnegligible probability, forever honest players need sending or forwarding a *quadratic* $\Omega(nt)$ number of *signatures* after the moment when the network becomes synchronous (a.k.a. "global stabilization time" GST). This also carries over the variant model where the maximum network delay is a priori unknown. Particularly, this is not a consequence of the synchronous one of Abraham et al (PODC'19), whose "rushing" Environment *can* in addition delete messages that were sent. This establishes a *separation* with the synchronous setting, where King-Saia (PODC'10) exhibited a protocol in $O(n\sqrt{n})$ bit complexity of communications (we assume like them that messages are confidential), which furthermore does not even use digital signatures.
- Under the same model (Theorem 6): consider protocols in which so-far honest players send a total *number* of messages after GST which is *linear* in the number n of players, then the number of rounds after GST before every player outputs is at least in $\log(t)$. This establishes a separation with the aforementioned message-linear and constant-round algorithms, which make further assumptions to achieve random committee/leader sortition.
- Under *synchrony*, when honest players only *multicast* messages, then (Theorem 9) we have that with nonnegligible probability, so-far honest players need sending a total *quadratic* number of *messages*. This strictly reinforces Abraham et al's (PODC'19) multicast lower bound, whose argument does not handle the case where messages carry digital signatures. This makes a separation with the aforementioned protocols, which also proceed by multicasts.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms

Keywords and phrases Author: Please fill in \keywords macro

Digital Object Identifier 10.4230/LIPIcs...

1 Model, motivations and results

Confidential message-passing with public metadata We consider players $\mathcal{P} := P_1, \ldots, P_n$, which are probabilistic machines whose identities are public and fixed. They are linked with pairwise authenticated confidential communication channels. In particular, players receiving a message automatically learn the identity of the sender. The Environment $\mathcal{E}$ is a polynomial probabilistic machine, to which the network leaks only the identity of the sender and of the recipient of ever message sent. $\mathcal{E}$ initializes every honest player with a binary input value.

© n;
licensed under Creative Commons License CC-BY

Standard adaptive Byzantine corruptions The Environment can corrupt any honest player at any time, up to a total of t corruptions. Corrupted players are *malicious*, in that they share all their state with the Environment and are totally controlled by it. In Definitions 1 and 2 below we distinguish actions performed by *so far honest* players, that is, sent while they are still honest. That is, we allow them to become corrupted later, as in [1, Theorem 3]. By contrast, when restricting to *forever honest* players, we count only the messages/signatures sent or forwarded by honest players who never become corrupted later in the execution.

Contrary to the lower bound [1, Theorem 1] (PODC'19), we do *not* assume that the Environment is rushing: it cannot delete any message sent.

Randomized has a twofold meaning. *First*, each player has access to a *local* random source, as [3]. But players do *not* have access for free to a global coin (that is: a public unpredictable value, as known as "global coin" or CRS) as in Rabin [20] or [10]. *Second*, the aforementioned specifications of consensus are relaxed, in that we can tolerate some negligible probability of safety or liveness failure (as in [7, 1]), which we will specify.

Partial synchrony [14, §2.2 (3)] and termination condition the Environment controls the delivery delay of every message sent on the network. It can deliver instantaneously messages between corrupted players, and arbitrarily delay any message sent. The Environment may however secretly commit, at some point called Global stabilization time (GST), to deliver all messages sent, *even those previously delayed*, within a fixed delay Δ. The value of Δ is fixed by the Environment at the beginning of the execution and made public to the players. We assume that players have a global clock which enables them to communicate in the form of successive rounds of duration Δ. Termination of consensus is required only *after* GST, if it ever happens. Notice that the Environment may well never set GST and still schedule the delivery of messages within Δ, as if synchrony held. Then, players playing consensus would anyway also ultimately output, since these conditions are indistinguishable from if GST did happen. Finally our communication lower bounds are interesting only in such partially synchronous models where messages *cannot* be lost. When messages can be lost before GST, then [14, §4.2 Remark 2] notice for any 1-resilient consensus algorithm, there always exists *a run in which a player never stops sending messages*, even after GST.

Partial synchrony [14, (2)] and termination condition In this model, messages are always delivered within a delay Δ that is set by the Environment at the beginning of the execution, but which is unknown to players. Let us introduce a possibly new termination condition. We say that, in an execution, honest players *"guess the actual value"* of Δ if, from some point in this execution, players play the protocol round by round with delay Δ between each round. Termination of consensus is then required only to happen *after* honest players have guessed the actual value of Δ, if this ever happens. This is restrictive, so makes the lower bounds stronger. This could be formalized with unreliable detectors controlled by the Environment.

a Consensus protocol as formalized in Ben Or [3] or DLS [14], is such that we have:

(Consistency) no two honest players output different values;

(Termination) after a polynomial number of rounds after GST (respectively: a polynomial time after all players have guessed Δ), every player ultimately ouputs a value (one could specify up to some probability, as we do in Theorem 9, and could be easily done in Theorems 4 and 6);

(Strong/Weak Unanimity) in the Strong variant, if all forever honest players have the same input value then this value should be the one they output. In the Weak variant, this is required only if, in addition, all players are forever honest.

$\mathcal{F}_{ds}$ the standalone digital signatures assumption (a.k.a. the authenticated setting) In this paper we consider the classical model as described concretely e.g. in Dolev-Reischuk [12]: "we allow the processors to share a signature scheme that enables each one to sign its messages so that every receiver will recognize them as being signed by it and no one can change the contents of a message or the signature undetectably. Such a scheme is the one suggested in [...] and its use for Byzantine algorithms is described in [...]. We allow faulty processors to collude for cheating. Therefore every message that contains only signatures of faulty processors can be produced by them." This is explicit in the classical works [13], [19, §4], [20], [14, §2.2], [15, §7], and also all works in which only the possibility to authenticate the original issuer of the message is actually used: PBFT [8], Cachin et al [6, §2.3.1] (nonwithstanding aggregation of multiple signatures) or the recent Kogias et al [18]: "we assume that messages are authenticated in a sense that if an honest party P_i receives a message m indicating that m was sent by an honest party P_j, then m was indeed generated by P_j at some prior time." This assumption is traditionally called the "authenticated" setting, but we prefer here the terminology *standalone digital signatures*, in order to avoid confusion with the weaker plain setting where only the communication channels are authenticated (see the discussion about [1, Theorem 3] in §1.1). Importantly, players *do not* have access to a trusted bulletin board that publishes their public keys matched with their identities (see §1.1.0.1 for this stronger model). Let us abstract out the model as follows.

Messages allowed by the network are constituted of concatenations of *two* well-separate datatypes: strings of bits; and signatures (so that a message can well consist in a single signature). signatures are produced and used as follows:

- On request $\mathsf{sign}(m)$ from any player P, where m is a message: $\mathcal{F}_{\text{ds}}$ outputs a *new* signature to P, that we denote $s_{P,i}(m)$ (i the number of previous identical requests) and adds $(m, s_{P,i}(m),$ "signed by P") to its internal registry.
- On request $\mathsf{check}(m, s, P)$ from any player Q with m a message and s a signature: *if* the registry contains $(m, s,$ "signed by P"), then $\mathcal{F}_{\text{ds}}$ outputs true; and false otherwise.

signatures are objects which can be *copied*, but which are otherwise *immutable.* signatures all look like *identical* from the points of view of players: the *only* information that players can extract from a given signature is the output of the check request to $\mathcal{F}_{\text{ds}}$. New signatures can *only* be created by $\mathcal{F}_{\text{ds}}$, from the sign request. [To fix ideas in a more concrete way, one could consider that on every sign request, then the signature output is a uniform random string in $\{0,1\}^\kappa$, with κ a security parameter. But since these strings are mutable, this would then require tedious discussions on what happens if a signature is sent into (possibly incomplete) pieces. The "number of signatures sent" would then be a fractional number, equal to the total quantity of information on these strings which is sent].

In particular, the Environment can *simulate a fairly sampled execution of a protocol in the $\mathcal{F}_{ds}$ model, up to a polynomial time/number of rounds, without actually corrupting the players involved.*

We say that a s is a *valid* signature of P for some message m if and only if s is the output of a request $\mathsf{sign}(m)$ from P; and it is *invalid* otherwise. So by definition, $\mathsf{check}(m, s, P)$ returns true if and only if s is a valid signature of P on m. signatures *issued* by P are by definition those which are valid for P on some message m. the definition of $\mathcal{F}_{\text{ds}}$ implies that the only way the Environment can obtain a true answer on a check query, is to have an actual corrupted player make the query with an actual valid signature s from P on m. Said otherwise, the Environment cannot guess if some message m was signed by P by brute-forcing many $\mathsf{check}(m, *, P)$ (which would not be tractable anyway since it is polynomial). In this model, what corresponds to a "signed message" in the concrete sense may be obtained as a message equal to a concatenation of the form $m' = m \| s$, where s is a valid $\mathsf{signature}$ on m. So by definition $\mathsf{check}(m', s, P)$ returns false, since s is thus *not* a valid signature on m'.

Number of signatures and number of messages complexities (but not bit-complexity) Let us define a measurement that formalizes, in our $\mathcal{F}_{\mathrm{ds}}$ model, the concrete [12]: "Since there exists an algorithm for reaching Byzantine Agreement without using signatures, the lower bound is meaningless unless we somehow count the messages that do not contain signatures. We make the technical assumption that every message in an authenticated algorithm carries at least the signature of its sender. Alternatively, the lower bound given below holds if one counts the number of signatures together with the number of messages without signatures.". For every player Q, in an execution e, let us denote $\mathsf{Sent}_e(Q)$ the sum over the messages m' sent by Q: after GST (resp.: after players have guessed Δ), if he is (forever / so far honest), of:

- the number of $\mathsf{signature}$s contained in m', $\Big[$Notice that if $m' = s_R\big(s_{Z_1}(m_1)\| \dots \| s_{Z_\ell}(m_\ell)\big)$, then by definition this number is 1 and not $\ell + 1$ because the $\mathsf{signatures}$ $s_{Z_j}(m_j)$ are *not* in the concatenation of objects forming m' (for instance if $j = 1$, notice that, concretely, someone receiving solely m would *not* be able to extract $s_{Z_1}(m_1)$ in any manner, nor even guess that this is the object signed by s_R, since she could not perform any successful check request without the $s_{Z_1}(m_1)$ at hand) $\Big]$
- *plus* 1 if m' is not of the form $m' = m\|s$, where s is a valid $\mathsf{signature}$ of Q on m (that is: if m' is "unsigned" in the concrete sense).

The rule of counting, as precised in the first item, makes our measurement comparable with the bit complexity (see §1.1): one signature in a message counts no more than one unit. The cost for this optimistic measurement is that this will require an additional "technical assumption" for the result of Theorem 4.

▶ **Definition 1** (the "number of signatures sent" by (forever / so far) honest players after GST)**.** *is* $\sum_{Q \in \mathcal{P}} \mathsf{Sent}_e(Q)$.

The next measurement is (strictly) more optimistic than the *total number of messages sent* in the concrete sense of [12], since we count only once the possibly multiple messages sent from Q to R in a given round.

▶ **Definition 2** (the "number of messages sent" by (forever / so far) honest players in an execution after GST)**.** *is the sum over all rounds r of the execution taking place after GST, over all (forever / so far) honest player Q, of all players R such that, in round r, Q sends at least one message to R on the network.*

1.1 Overview of the results

First we are unaware of any lower bound in the litterature specific to the communication complexity under partial synchrony. Recall that this model was introduced by Dwork et al [14] to circumvent the impossibility of fully asynchronous consensus. For instance, this setting is handled by popular state machine replication protocols such as Castro-Liskov's [8] or Hotstuff [23] (PODC'19), which are both deterministic and with a static environment. Precisely, recall that termination in these protocols relies on timing assumptions: if a honest leader does not benefit from a "synchronous enough" network, then he may be replaced before he could enforce a decision. Recall ([14, Theorem 4.4]) that authenticated Byzantine consensus under partial synchrony is possible only if $t < \frac{n}{3}$.

In Theorem 4 we show that in any randomized protocol for binary consensus resilient under standard adaptive Byzantine Environments, then there exists an Environment under which forever honest players always send a quadratic ($\Omega(nt)$) number of signatures after GST in any execution, in the sense of Definition 1. This is not a consequence of [1, Theorem 1] because in our model, the only possibility for the Environment to remove forever one message is to delay it forever and set GST=∞, but then players are not required anymore to `output`, contrary to the synchronous [1, Theorem 1]. The proof combines [12, Theorem 1] (recalled in §A.2.1), the simulation argument of e.g. [1, Theorem 3], and in addition the ability of the Environment to delay until GST any message sent before. Let us derive the consequences. In a concrete protocol in the sense of [12], when Q forwards to R the signatures of x distinct players P_i, then this represents at least x bits sent from Q to R. This means that the bit complexity is at least as large as the message complexity, and thus is also *quadratic*. By comparison, the synchronous consensus protocol of King-Saia [17] has a bit complexity of $O(n\sqrt{n})$ which is *strictly lower*, while under a *strictly weaker* model (apart synchrony): same standard adaptive Environments and confidential messages, no setup (the global coins are emulated), while assuming *no digital signatures at all*. Thus, Theorem 4 establishes a *strict separation* between the bit complexity of partial synchrony and synchrony. The above argument shows that closing this gap would require to enrich the model with the possibility to *aggregate* multiple signatures into one message of constant ($O(1)$) bit complexity. The state of the art implementation is [21], but makes the strong assumption of a trusted setup (see below). Alternatively, the asynchronous [18] achieves this assuming only standalone digital signatures as here, but with a $O(n^4)$ bit complexity for setup. Finally, the closest solution to the King-Saia setting may be [5, §5.2], which aggregates batches of signatures of logarithmic sizes, assuming the intermediate "registered PKI" model (see 1.1.0.1).

Second achieving a communication size which is *linear* in the number of players against an adaptive environment, plus a *constant* expected number of rounds, was recently popularized by Chen-Micali [10] (Algorand), Ouroboros-Praos [11], Abraham et al [1, Theorem 2] (PODC'19) and Chan et al [16] (EC'19). These papers adapt consensus protocols to a setting where only a small random set of players talks in each round. Precisely, these random "committees" are sampled with the help verifiable random functions (VRF). These functions enable a player to prove that his secret signature key satisfies some rare condition (analogous to a hash with many zeros in Bitcoin), without revealing his secret key. When reading the last two papers quickly, it might seem that a Public Key Infrastructure (PKI) is sufficient to achieve such protocols. Indeed, a PKI is the sole assumption in their [1, Theorem 2], while [16, Theorem 1.1] specifies only "standard bilinear group assumption". But actually if corrupted players could *choose* their secret keys before the protocol starts, then they could brute force it, and obtain membership in many committees. This would e.g. ruin the consensus of [1, §6.2] (to stick with the partially synchronous setting). This is why these papers actually assume that a *trusted* dealer gives the keys to the players (mentioned as the "Trusted PKI setup" page 19 of [16]). Another way around consists in assuming that an unpredictable random string is learned *after* the keys are set up. In [10] this is assumed to be satisfied by the hash of the previous block. In Praos [11] this is formalized as the ideal functionality $\mathcal{F}_{INIT}$ page 10, that creates a public random string $\eta \leftarrow \{0,1\}^{\lambda}$ (in a "genesis block"). This has to be done everytime new players receive a big amount of stake. A CRS actually also turns out to be assumed in [16, p19]. To be sure, although Hotstuff [23] has linear bit complexity, recall that termination there is also conditionned to a honest leader, which could be adaptively corrupted in our setting. Let alone that this protocol satisfies only Weak unanimity.

By contrast we show in Theorem 6 that, assuming standalone digital signatures only, then, for every protocol in which the number of *messages* sent by so-far honest (in the sense of Definition 2) players after GST is *linear*, then there exists an Environment under which the number of rounds before termination is at least *logarithmic*. The proof follows the pattern of the previous, but this time, the baseline Environments also adaptively corrupt players who receive many signatures. Let us point the related randomized lower bound of Bar Joseph and Ben Or [2] in $\Omega(\sqrt{n})$ rounds, but assuming that the Environment *knows* the full internal state of players, including the local randomness they sample before they send their messages. Notice that Theorem 6 considers the number of messages complexity, which is the most optimistic one. For instance, sending an aggregated signature anyway counts as one message. Considering existence of the previous committee-based algorithms, a direct consequence of Theorem 6 is:

▶ **Corollary 3.** *The functionality $\mathcal{F}_{mine}$, as defined in [1, §C.2] and which returns eligibility of a vote, cannot be implemented in constant round and linear communication complexity under standalone digital signatures, even assuming aggregated signatures.*

Third recall the *synchronous* lower bound of [1, Theorem 3], that states that any broadcast protocol in which players *multicast* messages only, and resilient against a standard adaptive Environment, has *quadratic* message complexity. The problem is that this is introduced by "we additionally investigate whether the remaining setup PKI assumption is necessary", which seems to refer to the *trusted setup model* of their [1, Theorem 2], whereas their lower bound *does not* even assumes digital signatures ("plain authenticated channels" only). We do not see why ruling out this weak model would make necessary the very strong assumption of a trusted setup. So in this paper we investigate the question for the intermediate standalone digital signatures. In §A.2.2 we revisit the proof of [1, Theorem 3], which we show would fail if assuming digital signatures on the messages sent or forwarded by honest players. We then show in Theorem 9 that this quadratic lower bound does also hold *even if* assuming digital signatures, for the problem of consensus, .

1.1.0.1 Extension to stronger models

The one called *registered PKI* in Boyle-Cohen-Goel [5, p9], for which they refer to e.g. Boldyreva [4] (PKC'03), is strictly stronger than our "standalone digital signatures". There, players can publish a public key on a public bulletin board before the execution starts (before they receive their input), which makes it known to all players at the beginning of the execution. From this, and additional cryptographic assumptions, players can exhibit random strings provably derived from the corresponding secret key, without disclosing it. This is leveraged in [11] and in [10] (which considers furthermore hashes of previous blocks as an unpredictable CRS) and in [1]&[16, p19] (assuming furthermore a trusted key setup). Our lower bounds *do* also hold in this model, but provided an *unlimited* Environment. Indeed in the proofs of Theorems 4, 6 and 9, it could then sample a distribution of secret keys conditioned to the public keys on the board.

One could also consider an intermediate model, without any public key board but with a trusted authority that delivers certificates to players, that match their public keys with their identities. A player P sending for the first time a signed message to player Q, with his public key, may then exhibit this certificate to Q. Our lower bounds would also hold in this model if the authority accepted to certify *multiple* keys from the same player. Indeed in our proofs, a freshly corrupted player could then ask for a *new* certificate related to the public/private key pair of the execution simulated by the Environment, use this new public key towards the isolated player p, while continuing to use the old one with the remaining honest players.

1.2 Roadmap

In §2.1 and §2.2 we make more explicit the model of behavior of players and provide additional notations for the proofs. We state precisely and prove Theorems 4, 6 and 9 in §2.3, §2.4 and §2.5. In the Appendix we revisit known synchronous lower bounds. Particularly, in §A.2.3 we take the opportunity to expose a detailed proof of the [12, Theorem 2], and provide what we think are fixes for two existing arguments in the litterature.

2 Precise statements and proofs

2.1 Disclaimer on a simplification made in the statements of Thms 4 & 6 and in the proof of Thm 9

For simplicity in the statements of Theorems 4 and 6, we consider protocols that always satisfy Termination and (Strong or Weak) unanimity, that is, with 100% probability. So that we can concentrate on exhibiting Consistency failures in the proof. Of course when considering protocols for which Termination or Unanimity can fail up to some probability, then our arguments as such would not work. But then, these failures sum up in the total probability of failure anyway. Adapting this intuition rigorously would follow exactly the same pattern as in the proofs of [1, Theorems 1 & 3]. As for Theorem 9, since the proof follows exactly the same pattern as the [1, Thm 3], then we allow ourselves to state it with the same probability of overall failure ($\leq 1/6$), although, for simplicity in the proof, we do as if Termination and Strong unanimity always held, leaving the reader to check the detailed proof of [1, Thm 3] in case he needed to be convinced about the 1/6.

2.2 General ingredients and precisions

We denote $|S|$ the cardinality of a set S.

the State of a honest player Q in a given execution e up to time τ, and denoted $e_\tau(Q)$, is his history up to τ, and consists of:

- His initial binary input value given by the Environment
- The messages he received so far, along with the time at which he received them, and the identity of their sender. We insist that players react identically to all signatures received in messages (or copied or obtained from $\mathcal{F}_{ds}$). The only extractable information from them are the outputs of check .
- The random coins he tossed so far (which are purely local computations), along with the time at which they were tossed
- Although these are local computations in the concrete sense, we specify that the state also contains the results of his check requests.

We insist that players do *not* receive any other incoming information than their input, messages from other players, and results of check . E.g. they do *not* receive any notification of public keys from some board as in §1.1.0.1 (let alone of a private key from a trusted dealer).

For instance, a player P receiving a message m_1 from P_1 containing a signature of P_2 on some message m_2 enclosed, will add everything to his current state: the time of delivery and issuer P_1, and the *whole* content of m_1, including the signature of P_2... irrespective to whether or not this content itself would be specifically of the form $m_1 = m \| s_{P_1}(m)$ (in the concrete terminology: where m would be the "body" of m_1 and $s_{P_1}(m)$ a valid signature of P_1 "on m_1", see Definition 1).

Actions taken by a honest player P (in the [14, §2.2 (3)]: at the beginning of a round) are completely *determined* by his current state, and consist of:

- local computations, including: tossing coins, performing sign requests, copying signatures, creating messages as concatenations of bitstrings and signatures etc.
- sending messages
- output a value
- check requests (although these are local computations in the concrete sense)

Random variables Let us transpose the vocabulary of the deterministic [12]. For a given protocol, we will often identify a given environment $\mathcal{E}$ with the probabilitic set of executions under $\mathcal{E}$. Then, for instance, the "number of signatures or messages sent" (Definitions 1 and 2) under $\mathcal{E}$ denotes actually the corresponding random variable with respect to this probabilistic set of executions. We call execution e *restricted* to a honest player, or group of honest players Q, and denote $e(Q) := e_\infty(Q)$, the history of internal states of players in Q during the whole execution (or up to some time τ when precised). For a *fixed* environment $\mathcal{E}$ and for a player, or group of players, Q, we note $\mathcal{E}(Q)$ the probabilistic set of executions restricted to Q under environment $\mathcal{E}$. In our arguments, we will meet the situation where distributions $\mathcal{E}(Q)$ and $\mathcal{E}'(Q)$ coincide on a probabilistic set of measure proba. In this event, then the probabilistic distribution of outputs of Q (if any) is the same. On the other hand, $(1 - \mathsf{proba})$ accounts for the events where the two sets of executions may *not* be equally distributed. Typically, because too much players interact with Q, and so $\mathcal{E}'$ is not able to "fake" the distribution $\mathcal{E}$ towards Q.

Players interacting with Q For a given player Q, in an execution e, let us consider the sets

- of players which receive a message from Q or a message containing a signature issued by Q (in the same strict sense than in Definition 1, we recall this in the next item). We call them the *players which are reached by* Q, and denote them $\mathrm{RB}_e(Q)$;
- of players such that Q either receives a message from them, or receives some message containing a signature issued by them. We call them the *players from which Q hears of*, and denote them $\mathrm{HO}_e(Q)$ (compare with [9]). [Notice that if Q receives the sole message $m = s_{R_2}(s_{R_3}(m_3))$, then by definition R_3 is *not* in the set, because the signature $s_{R_3}(m_3)$ is *not* in the concatenation of objects forming m (notice that, concretely, someone receiving solely m would *not* be able to extract $s_{R_3}(m_3)$ in any manner, nor even guess that this is the object signed by s_{R_2}, since she could not perform any successful check request without s_{R_2} at hand)]
- the union of the two previous sets is denoted $A_e(Q)$ (as in [12, Theorem 1]). We call them the *players interacting with* Q in execution e, and denote them $A_e(Q)$.

Under a probabilistic distribution of executions, in particular an Environment $\mathcal{E}$, then the previous become random variables: $\mathrm{RB}_{\mathcal{E}}(Q)$, $\mathrm{HO}_{\mathcal{E}}(Q)$ and $\left|A_{\mathcal{E}}(Q)\right|$. By definition we have

$$\left|A_{\mathcal{E}}(Q)\right| \leq \left|\mathrm{HO}_{\mathcal{E}}(Q)\right| + \left|\mathrm{RB}_{\mathcal{E}}(Q)\right| \tag{1}$$

Signatures received (and sent) In addition to the $\mathsf{Sent}_e(Q)$ of Definition 1, let us define for any player Q and execution e, the total number of signatures received by Q while he is honest, that we denote $\mathsf{Received}_e(Q)$. That is, we count every signature contained in messages received by Q, plus 1 for every message received which is not of the form $m' = m \| s_Q(m)$ (which means an "unsigned" message in the concrete sense). On the one hand we have:

$$\left|\mathrm{HO}_e(Q)\right| \leq \mathsf{Received}_e(Q) \tag{2}$$

but on the other hand there is no comparison between $\mathsf{Sent}_e(Q)$ and $\left|\mathrm{RB}_e(Q)\right|$.

2.3 Theorem 4

Of course, under partial synchrony, the results are meaningful only for $t < \frac{n}{3}$, otherwise no consensus protocol exists at all. The following theorem holds under all partially synchronous models, especially those where *every* message sent is ultimately delivered. We consider the two "optimistic" Environments $\mathcal{E}_0$ and $\mathcal{E}_1$ which: leave *all* players honest, set GST from the beginning, set the same public Δ network delay from the beginning (that is: GST=0), and give input 0, resp. 1, to all the players. Under these environments, then the Weak unanimity condition imposes that all players ultimately output 0, resp. 1.

A technical assumption We assume for Theorem 4 that honest players Q do not send signatures $s_P(m_P)$ in their messages to player R, such that the message m_P contains itself a signature $s_T(m_T)$ that Q did not send so far to R. Informally, Q does not send some signature to R on which R is unable to perform a check that returns true (even with unlimited computational power), from what Q sent to far to him.

▶ **Theorem 4.** *Consider the partially synchronous model [14, §2.3 (3)]. Assume a standard adaptive Byzantine Environment and standalone digital signatures. Then, for every $\eta > 0$ (small enough), consider a randomized protocol that solves consensus with Weak unanimity after GST and such that, under both the two optimistic environments $\mathcal{E}_0$ and $\mathcal{E}_1$, we have that: Unanimity and Termination hold with 100% probability, and the expected number of* signatures *sent by forever honest players after GST is $\leq \dfrac{1}{4}\eta n t$. Then the probability of a Consistency violation is at least $1 - \eta$.*

The same result holds under the model [14, (2)]: in the statement, replace "after GST" by "after honest players have guessed the actual network delivery delay".

We provide the detailed proof under the model [14, §2.3 (3)] (GST). At the end, we will briefly explain how to adapt the argument to the other model [14, (2)].

Notice that, under both optimistic environments $\mathcal{E}_0$ and $\mathcal{E}_1$, since all players are honest and all messages are delivered, we have that for every execution e, the number of signatures sent by (honest) players (Definition 1) is then the same as the number of signatures received by (honest) players:

$$\sum_{Q \in \mathcal{P}} \mathsf{Sent}_e(Q) = \sum_{Q \in \mathcal{P}} \mathsf{Received}_e(Q) \tag{3}$$

Also, since all players are honest, we have that:

$$\sum_{Q\in\mathcal{P}} \left|\mathrm{RB}_e(Q)\right| \leq \sum_{P\in\mathcal{P}} \mathsf{Sent}_e(P) \tag{4}$$

Indeed, for every player R reached by a given Q, we have at least one of the two events: either R receives a signature from Q sent by some honest player P, or R receives a message from the honest $P = Q$. Both cases count as 1 in the definition (Def 1) of $\mathsf{Sent}_e(P)$. When Q and/or R vary, then those events are *disjunct*. Thus, taking the sum over all R reached by Q, then over all Q, gives the inequality.

For each $b \in \{0,1\}$, let us sum the *expectations* E_b under $\mathcal{E}_b$ of the two equal quantities (signatures sent and signatures received) of (3). Then sum over $b \in \{0,1\}$. By assumption, the total is thus lower than $2 \times 2 \times \eta n \frac{t}{4} = \eta n t$. Applying (2) and (2), we obtain:

$$\eta n t \geq \sum_{Q\in\mathcal{P}} \mathrm{E}_0(\left|\mathrm{RB}_0(Q)\right|) + \mathrm{E}_0(\left|\mathrm{HO}_0(Q)\right|) + \mathrm{E}_1(\left|\mathrm{RB}_1(Q)\right|) + \mathrm{E}_1(\left|\mathrm{HO}_1(Q)\right|) \tag{5}$$

Dividing by n, we deduce that there exists a player p, such that in expectation:

$$\eta t \geq \mathrm{E}_0(\left|\mathrm{RB}_0(p)\right|) + \mathrm{E}_0(\left|\mathrm{HO}_0(p)\right|) + \mathrm{E}_1(\left|\mathrm{RB}_0(p)\right|) + \mathrm{E}_1(\left|\mathrm{HO}_0(p)\right|)\ . \tag{6}$$

▶ **Lemma 5.** *Sample two independent executions: one under $\mathcal{E}_0$, the other one under $\mathcal{E}_1$. Note $A(p)$ the set of players,* cumulated *over the two executions, such that* either *p is sent or forwarded a signature issued by some player in $A(p)$,* or *one of the players in $A(p)$ is sent or forwarded a signature issued by p. Then, with probability $1-\eta$, we have that the cardinality $\left|A(p)\right| \leq t$.*

Proof. First, notice that $A(p)$ is by definition lower than the RHS of equation (6). Then, the rest is a straightforward application of the Markov bound: divide the LHS of Equation (6) by η. ◀

Idea of the proof We consider the following Environment $\mathcal{E}$. It takes advantage that, under the optimistic environment $\mathcal{E}_0$, since we have GST=0 and unanimity of inputs, then all players must ultimately output 0 (respectively: 1 under $\mathcal{E}_1$). Its strategy will consist in having two groups of honest players: $\{p\}$ and $\mathcal{P}\backslash\{p \cup A(p)\}$ not interact with each other, and believe that they are facing respectively $\mathcal{E}_0$ and $\mathcal{E}_1$, and thus to ultimately output. Recall that in the classical deterministic proof of [12, Theorem 1] (which we adapt for convenience to consensus in A.2.1), the Environment knows these sets prior to the execution, so can corrupt $A(p)$ from the beginning, and succeed in his strategy with certainty. In our probabilistic setting instead, the Environment $\mathcal{E}$ *discovers* on the fly players which are sending messages to p, and those to which p sends messages. To achieve his strategy, $\mathcal{E}$ will: adaptively delay forever those problematic messages (so will never actually set GST), corrupt on the fly sufficiently many players (the set $A(p)$), and make them continue interacting with both $\{p\}$ and $\mathcal{P}\backslash\{p \cup A(p)\}$ as if they were still facing $\mathcal{E}_0$ and $\mathcal{E}_1$ (in particular: make them believe that GST holds).

Proof We consider the following Environment $\mathcal{E}$. It initially performs (a polynomial number of) simulations of executions (up to a polynomial time or number of rounds) under $\mathcal{E}_0$ and $\mathcal{E}_1$, to identify with overwhelming probability a player p such that (6) holds up to the time τ when all players output in both executions (which is a fortiori guaranteed by (6)). It gives input 1 to all players, which are all honest at the beginning, except to p, to which he assigns input 0. It sets a public Δ eventual delivery delay, the same as in the two optimistic environments $\mathcal{E}_0$ and $\mathcal{E}_1$. Then, $\mathcal{E}$ starts simulating the players in $\mathcal{P}\backslash\{p\}$ in this head, giving them inputs 0. Let us call this the *simulated* execution.

Then the real execution starts and runs at the same pace as the simulated execution, which is a little ahead in time, so that the Environment has the time to perform the following actions. Let $A(p)$ be the set of corrupted players so far (initially empty). Corrupted players continue their initial thread of playing honestly the real execution with the remaining so far honest players except p: $\mathcal{P}\backslash\{p \cup A(p)\}$, as if they still had input 1, and messages between them are still delivered within Δ. On the other hand, as soon as they are corrupted, players in $A(p)$ open a second thread that has the same behavior as in the *simulated* execution: messages sent to p by this thread are sent in the *real* execution (and de facto messages received from p), while messages sent to $\mathcal{P}\backslash\{p \cup A(p)\}$ are sent in the *simulated* execution. The corruptions enabling this strategy are as follows:

- As soon as any player q in the *real* execution sends a message to p, then corrupt this player q and arbitrarily delay the message that he sent to p
- As soon as p sends a message to a player q' in the *real* execution, then corrupt q'.
- As soon as new players are Reached By p in the *simulated* execution, then corrupt their counterparts in the *real* execution;
- At the beginning of each round of the real execution, $\mathcal{E}$ checks to see which players R will send a message m to p in this round of the simulated execution under $\mathcal{E}_0$. For every such R and m:
  - let $R_2, \ldots, R_k$ the (possibly empty) set of players which is added to the ones that p heard of so far in the simulated execution under $\mathcal{E}_0$, due to the reception of m.
  - $\mathcal{E}$ corrupts players $R, R_2, \ldots, R_k$ in the real execution before they could possibly sent any message in this round of the *real* execution;
  - by the technical assumption, the corrupted players (which by definition include the ones that p heard of so far in the simulated execution up to reception of m) are able to create all the signatures appearing on m in the simulated execution (recall that all signatures are indistinguishable from p, up to the output given by his check requests on them).
  - use these signatures to form the message m (which we recall is by definition a concatenation of these signatures and bitstrings). Have R send m to p in the *real* execution.

This guarantees preservation of the following *Invariant*: consider the set of real executions in which the Environment can afford his strategy. Then, restricted to the honest players $\mathcal{P}\backslash\{p \cup A(p)\}$, who did not interact with p, they follow that same distribution than randomly sampled executions under Environment $\mathcal{E}_1$. Symmetrically, restricted to p, they follow the same distribution than randomly sampled executions under Environment $\mathcal{E}_0$. Thus, in these executions, p must then ultimately output 0. Whereas the other honest players must ultimately output 1, by indistinguishability with the distribution under $\mathcal{E}_1$. So this is a consistency failure in the real execution.

The Environment can afford its strategy up to probability at least $1 - \eta$ because then, by Lemma 5, the corruption limit has not been achieved: $|A(p)| \leq t$.

Adapting the proof for the other model [14, (2)] (Δ unknown) We consider the same baseline optimistic Environments $\mathcal{E}_0$ and $\mathcal{E}_1$, but in addition which both set a (small) delivery delay Δ' for messages. Then, the Environment $\mathcal{E}$ sets $\Delta >> \Delta'$ high enough, and makes players guess the wrong Δ', such that, with the previous strategy, all honest players are very likely to output a value before Δ has elapsed. Concretely, in the previous strategy, the

Environment delivers all messages within delay Δ', except those that he wants to arbitrarily delay. Although these messages are still guaranteed to be delivered after Δ, the strategy will still lead to the same Consistency failure because all honest players will have output by then.

2.4 Theorem 6

2.4.1 Statement and idea of proof

For readability we it with a *fixed* number of rounds R, as in Rabin [20, §5], and also with a η-*worst case* message complexity, as in [1, Theorem 3]. This is no more than a matter of taste: [1, Theorem 1] was stated instead with *expected* complexity (as the previous Theorem 4), and, [1, Theorem 2] was stated in terms of expected number of rounds .

More importantly, the proof will involve two baseline Environments which, now, corrupt some players. This is the reason why we cannot anymore rule out consensus with weak unanimity. And also why we *enlarge* the message complexity measurement: we now consider messages sent by *all* so-far honest players, *including* those who may be corrupted later in the execution.

▶ **Theorem 6.** *Consider the partially synchronous model [14, §2.3 (3)]. Assume a standard adaptive Byzantine Environment and standalone digital signatures. Let $\eta > 0$ be small enough, $\Lambda > 0$, and consider a randomized protocol for consensus with Strong unanimity such that under* every *possible fixed Environment, we have that: Unanimity and Termination hold with* 100% *probability, and so-far honest players send a total $\leq \Lambda n$* messages *after GST, except with probability $\leq \eta$. Assume furthermore that players output within R rounds after GST in* every *execution, where R is such that*

$$R \leq \frac{\log(t) - \log 4}{\log \Lambda + \log 4 + \log(n/t)} .$$

Then the probability of a Consistency violation is at least

$$\geq \left(1 - \frac{t}{2n}\right)\left(1 - 2\eta\right) .$$

The same result holds under the model [14, (2)]: in the statement, replace "after GST" by "after honest players have guessed the actual network delivery delay".

Notice that $n/t = 3$ in the maximal corruption tolerance. The basic idea is that the Environment can silent forever the players which receive more than M *messages* in some round (M a parameter to be defined). Then, for every player Q which is still honest after R rounds, the number of *distinct* players of which Q will have received/been forwarded *signatures*, is bounded by $2 \times M^R$. Indeed: a message sent in the first round contains at most the signature of 1 player, thus, by assumption, a message sent in the second round contains at most the signature of M distinct players, then we conclude by recursion that Q hears at most from the signatures from M^R distinct players. And, seen the other way, the graph of communications from Q forms a tree with at most M branches at each node, so reaches at most M^R players. We can then adapt the proof of Theorem 4.

We adapt the proof of Theorem 4 by modifying the baseline Environments: $\mathcal{E}_0$ and $\mathcal{E}_1$, which now also *corrupt* players. As before, $\mathcal{E}_0$ and $\mathcal{E}_1$ set GST from the beginning and give all players input 0, resp. 1. But in addition, they adaptively silent in every round the players that collect many signatures. Precisely, in every round, they corrupt the "collector" players who received more than $M := \dfrac{4\Lambda n}{t}$ messages in the previous round, and instruct them not to send any message forever. When either $\mathcal{E}_0$ or $\mathcal{E}_1$ achieves the corruption quota of $t/4$, it raises a "give up" flag in his head and stops corrupting players. The following Lemma formalizes to what extent they can afford their strategy without raizing the flag. For $b \in \{0,1\}$, for simplicity, note C_b the probabilistic set of players that, *overall* in an execution under $\mathcal{E}_b$, receive more than $\dfrac{4\Lambda n}{t}$ messages.

▶ **Lemma 7.** *Consider the two environments $\mathcal{E}_0$ and $\mathcal{E}_1$. Sample (independently) one execution of the protocol under each of them. Then, with probability greater than $1-2\eta$, we have that both cardinalities $\left|C_0\right|$ and $\left|C_1\right|$ are* simultaneously *bounded.*

$$\left|C_b\right| \leq \frac{t}{4} \quad \textit{for } b \in \{0,1\}. \tag{7}$$

Proof. Let b be equal to 0 or 1. By definition of $\mathcal{E}_b$, only honest players send messages. And thus the total number of messages *received* by players is equal to the total number of messages sent by so-far honest players. Which is, by assumption, $\leq \Lambda n$ with probability $1-\eta$.

Thus, under this (likely) event, there are at most $\left|C_b\right| \leq \Lambda n \dfrac{t}{\Lambda 4n} = \dfrac{t}{4}$ distinct players receiving more than $\dfrac{4\Lambda n}{t}$ messages overall in the execution.

Finally, we have probability $1-2\eta$ that this holds *simultaneously* in both the sampled executions. ◀

Next, the following Lemma 8 parallels Equation (5), and determines when our final Environment will be able to "frame" one player as before. It is a consequence of the intuition given after the statement of Theorem 6.

2.4.1.1 (so far) HeardOf(Q) / ReachedBy(Q) in the most possible inclusive sense: players required to make the signatures (so far) received by Q / whose signatures (so far) received require the action of Q to be constructed

We do not make anymore the "technical assumption" made for Theorem 4. Now we enlarge the definition of players HeardOf(Q) to include all those necessary to construct the messages (so far) received by Q. That is, if Q receives the sole message $s_{R_2}(s_{R_3}(m_3))$, then we now say that Q heard of both R_2 and R_3 since this message cannot be created without actions from both of them. The set ReachedBy(Q) is equally enlarged in the same inclusive sense: players who (so far) received signatures such that an action of Q is necessary to construct them.

▶ **Lemma 8.** *We still consider two independent executions of the protocol under $\mathcal{E}_0$ and $\mathcal{E}_1$. Note $A(Q) := A_0(Q) \cup A_1(Q)$. Then, under the event of* (7) *(of probability $1-2\eta$), we have that for* every *forever honest Q:*

$$\frac{t}{2} \cong 4\Big(\frac{8\Lambda n}{t}\Big)^R \geq \left|\mathrm{RB}_0(Q)\right| + \left|\mathrm{HO}_0(Q)\right| + \left|\mathrm{RB}_1(Q)\right| + \left|\mathrm{HO}_1(Q)\right| \geq \left|A(Q)\right| \tag{8}$$

Proof. The RHS upper bound is by Equation (1).

First, let us recall the intuitive fact on the signatures sent and received. Consider $b \in \{0,1\}$, and an execution under $\mathcal{E}_b$. By assumption we are in the (likely) event where (7) holds. In particular, $\mathcal{E}_b$ is able to carry his strategy without raizing the "give up" flag. Hence, we have that throughout the execution, the players who send messages in a round are those who so-far received $\leq \dfrac{4\Lambda n}{t}$ messages in *every* previous round. To initialize the recursion: a player who sends message in the second round cannot have heard of so far from more than $\frac{4\Lambda n}{t}$ distinct players, and thus cannot send or forward signatures that require more than $\frac{4\Lambda n}{t}$ distinct players to construct them. We continue by recursion until the beginning of round R, where each player did not hear of more than $\left(\frac{4\Lambda n}{t}\right)^{R-1}$ distinct players. And thus in turn, all the signatures that every single honest player will send or forward in R, are guaranteed to be constructible from a set of players (in our inclusive sense of Heard Of) of size *lower* than this number. Thus overall in such an execution, any player *who remained honest at the end of round* R received signatures that are constructible from a set of players of size lower than $\mathrm{HO}_b(Q) \leq \left(\frac{4\Lambda n}{t}\right)^R$. Symmetrically, every player Q cannot reach more than $\mathrm{RB}_b(Q) \leq \left(\frac{4\Lambda n}{t}\right)^R$ distinct players in the inclusive sense above.

Summing both previous upper bounds, then over both executions $b \in \{0,1\}$, we deduce the upper bound in the middle of Equation (8).

Taking the log of $4\left(\frac{4\Lambda n}{t}\right)^R$ yields $\log(4) + \log(t) - \log(4)$, which is (nearly) the log of the LHS of (8). ◀

Proof of Thm 6 We can now describe the Environment $\mathcal{E}$ that will provoke consistency failures. The pattern is the same as in 4, we just change the baseline Environments $\mathcal{E}_0$ and $\mathcal{E}_1$. $\mathcal{E}$ selects a player p at random. It gives input 1 to all players, except to p to which it gives input 0. It initializes three sets of corrupted players, initially empty: the real ones $A(p)$ and C_1, and the virtual one C_0. It starts a simulation in his head where he runs environment $\mathcal{E}_0$. Particularly, it corrupts in his head every player that receives more than $\frac{4\Lambda n}{t}$ messages at the end of a round. Such virtual player is added to the virtual set C_0 and virtually silented forever. He does likewise in the real execution, adding silented players to the real set C_1. If by chance p gets corrupted in the simulated execution (added to C_0) or in the real execution (added to C_1), then the Environment raises an “abort” flag and stops his strategy.

The rest carries over unchanged: as soon as any player q in the *real* execution sends a signature to p, then $\mathcal{E}$ corrupts this player q and arbitrarily delay the original message he sent to p. As soon as p sends a message to a real player q', then corrupt q'. If in some round of the *simulated* execution there is a player Z which sends a message to p, which possibly requires participation from other players: $Z_2, \ldots, Z_k$ to be created (in our inclusive sense of Heard Of), then $\mathcal{E}$ corrupts the corresponding players $Z, Z_2, \ldots, Z_k$ in the real execution at the beginning of this round, before they had time to send any message. $\mathcal{E}$ makes them issue the signatures appearing in the simulated execution (and forge any possibly additional unsigned message forwarded by Z to p, when the case), creates from these signatures the message as in the simulated execution, then makes Z actually send it to p.

As soon as they are corrupted, players (in $A(p)$) are made to play the protocol with p exactly as in the simulated execution under $\mathcal{E}_0$, but continue playing the real execution with the remaining honest players, as if still under $\mathcal{E}_1$.

This guarantees preservation of the following *Invariant*: consider the set of real executions in which the Environment can afford his strategy. Then, restricted to the honest players $\mathcal{P}\backslash\{p \cup A(p) \cup C_1\}$, who did not interact with p, they follow the same distribution than randomly sampled executions under Environment $\mathcal{E}_1$. Symmetrically, restricted to p, they follow the same distribution than randomly sampled executions under Environment $\mathcal{E}_0$.

Thus, *if* p remains uncorrupted at the end of the R-th round, then it must output 0 as under $\mathcal{E}_0$ (Weak unanimity + after GST), whereas the other honest players must ultimately output 1 at the end of the R-th round as under $\mathcal{E}_1$ (Weak unanimity + after GST). So this is a consistency failure in the real execution.

Environment $\mathcal{E}$ can afford its strategy if those three sufficient conditions hold:

- p does not get corrupted in C_0 in the simulated execution, nor in C_1 in the real execution. This happens with probability at least $1 - t/(2n)$ because p was chosen at random, while the set to avoid is of size $\leq |C_0| + |C_1| \leq t/2$;
- $\mathcal{E}$ can silent players in both the real and simulated executions, without reaching the corruption quota $\frac{t}{4}$ in both. By Lemma 7, this happens with probability at least $1 - 2\eta$;
- the size of the set $A(p)$ that $\mathcal{E}$ also corrupts in the real execution remains smaller than $\frac{t}{2}$. But, under the two previous conditions, then notice that p is a forever honest player in *both* executions. Namely the real one, which has the same distribution than under $\mathcal{E}_1$ for the players which did not interacted with p so far, and the simulated one, which is exactly under $\mathcal{E}_0$. So we conclude by Lemma 8 that this third condition also holds.

In conclusion, those three sufficient conditions hold with probability greater than $\big(1 - \frac{t}{2n}\big)\big(1 - 2\eta\big)$.

2.5 Theorem 9

▶ **Theorem 9.** *Consider any* synchronous *randomized protocol for consensus, assuming standalone digital signatures, such that Termination + Consistency + Strong Unanimity hold with probability $p > 5/6$ under any standard adaptive Byzantine Environments with up to t corruptions; and such that honest players only* multicast *messages. Then, there are executions in which so-far honest players multicast more than t messages.*

We just describe the differences with the baseline argument that we set in A.2.2, and still follow the simplification discussed in 2.1. There is no more designated sender P_2. Instead, Environment $\mathcal{E}_0$ gives input 0 to all players except to the corrupted P_1, and input 1 to all players in the simulated execution. Likewise for $\mathcal{E}_0'$, which gives input 0 to all the players except to player P_1, to which it gives the input bit b, which we do not specify on purpose. Whereas in the simulated execution, $\mathcal{E}_0'$ gives input 1 to all players. The same modifications holds for $\mathcal{E}_1$ and $\mathcal{E}_1'$ (inputs 1 in the real executions, 0 in the simulated ones, and the same bit b to P_1 under $\mathcal{E}_1'$).

The second difference is that under $\mathcal{E}_0$ and $\mathcal{E}_1$, now, whenever a player q multicasts a message in the simulated execution, then it is immediatly *corrupted* in the real execution. This enables the Environment to use the signature of q and carry over the same strategy: make P_1 issue valid messages in the real execution, that he would have sent in the simulated execution.

Denote by Q_0 the probabilistic set of forever honest players under $\mathcal{E}_0$ *except* P_1. They must ultimately output 0, by Strong unanimity. Denote by Q_0' the probabilistic set of forever honest players under $\mathcal{E}_0'$ *except* P_1. The set of executions restricted to them has the same distribution than for Q_0 under $\mathcal{E}_0$. [1] Thus players in Q_0' also ultimately output 0. For the

[1] There are two ways to cope with this argument of a *probabilistic* group of players witnessing the same distributions of executions, which is also implicit e.g. in [1, Thm 1 & Thm 3]. The quick way around is just to fix once for all a q at random, and then conditioning on the events where he belongs to *both* Q_0 and Q_0' (considering two parallel samplings of executions under $\mathcal{E}_0$ and $\mathcal{E}_0'$). The formal way consists in seeing the set of honest players as one single big probabilistic machine that interacts with the

same reason (indistinguishability with $\mathcal{E}_1$), we have that forever honest players Q'_1 under $\mathcal{E}'_1$ must ultimately output 1. Finally, to ensure *Consistency* with Q'_0 in $\mathcal{E}'_0$, then the honest player P_1 must output 0. For the same Consistency reason, he must output 1 under $\mathcal{E}'_1$.

But $\mathcal{E}'_0$ and $\mathcal{E}'_1$ restricted to P_1 have identical distributions. So P_1 should output the same value in both, which is impossible. Notice that the value of his fixed input bit b played no role in the argument.

References

1 Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 2019 ACM PODC*, 2019.

2 Ziv Bar-Joseph and Michael Ben-Or. A tight lower bound for randomized synchronous consensus. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, 1998.

3 Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second ACM PODC*, 1983.

4 Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography — PKC 2003*, 2002.

5 Elette Boyle, Ran Cohen, and Aarushi Goel. Succinctly reconstructed distributed signatures and balanced byzantine agreement, 2020. arXiv:2002.02516.

6 Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*. Springer Berlin Heidelberg, 2001.

7 Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 2005.

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

9 B. Charron-Bost and A. Schiper. Improving fast paxos: being optimistic with no overhead. In *2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*, Dec 2006.

10 Jing Chen and Silvio Micali. Algorand, 2016. arXiv:1607.01341.

11 Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology – EUROCRYPT 2018*. Springer International Publishing.

12 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, January 1985.

13 Danny Dolev and H. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 1983.

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

15 Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. *Mathematical systems theory*, 1993.

16 T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Consensus through herding. In *Advances in Cryptology – EUROCRYPT 2019*, 2019.

17 Valerie King and Jared Saia. Breaking the o(n2) bit barrier: scalable byzantine agreement with an adaptive adversary. In *PODC '10*, 2010.

Environment (inputs and outputs) and with the corrupted players (messages). When some player q gets corrupted, then we model this by shutting down the program of q in the machine, and creating one new message interface port with the corrupted q, for every so-far honest player remaining in the machine.

18 Eleftherios Kokoris-Kogias, Alexander Spiegelman, Dahlia Malkhi, and Ittai Abraham. Bootstrapping consensus without trusted setup: Fully asynchronous distributed key generation. *IACR Cryptology ePrint Archive*, 2019:1015, 2019.
19 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 1982.
20 Michael O. Rabin. Randomized byzantine generals. In *Proceedings of the 24th Annual SFCS*, 1983.
21 Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Golan Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *to appear in IEEE Security & Privacy 2020*, 2020.
22 Vinod Vaikuntanathan. *Randomized algorithms for reliable broadcast.* PhD thesis, MIT, 2009.
23 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, New York, NY, USA, 2019. ACM.

A Revisiting state of the art lower bounds, their proofs and their limitations

A.1 Remainder: from broadcast to consensus

First, recall that consensus under synchrony is possible only with a *honest majority* $t < \frac{n}{2}$. Indeed, otherwise, consider four players: P_1 and P_2 honest with inputs 1 and 2. Plus P_3 and P_4 corrupted, who play honestly as if they had inputs 1 and 2. Then, to satisfy unanimity, P_1 must decide 1, because from his point of view it could be the case that only P_1 and P_3 are noncorrupted. And likewise P_2 must decide 2 to satisfy unanimity with the possibly honest P_4.

Second, let us recall the problem of broadcast, which is also known as "Byzantine agreement" [19]. In this problem there is a priviledged player P_1, the "sender", and the goal is to have every player output the same value, such that (validity) if P_1 is honest, then this value is P_1's input. Under synchrony *and* with a honest majority, then we have that the problem of broadcast is essentially *weaker* than consensus. Indeed, consensus then compiles itself into Broadcast with no communication overhead [22, §2.2]: have P_1 send his input s_1 to every player in the first round. Then in the second round players start a consensus protocol with input: the value that they received from P_1, if any, or a default value otherwise —e.g. 0. This is why all the known lower bounds for synchronous broadcast, that we recall below, *automatically hold* for consensus.

Third, we do not consider broadcast under partial synchrony in this paper because it is *trivially impossible*: consider a malicious sender which remains silent forever but GST holds from the beginning. Then honest players must output some value in a finite time. The problem is that from the point of view of honest players, the situation is undistinguishable from a scenario where the sender P_1 is honest with an input s_1 that they ignore, while GST is never established: the Environment delays all outcoming messages from P_1 forever. Since other players output anyway some value, this value is likely to be different from s_1, which violates validity.

A.2 Revisiting communication lower bounds and their proofs for *synchronous* broadcast

Let us revisit some synchronous bounds for authenticated broadcast and their proofs, from Dolev-Reischuk [12] (expanded from PODC'82) and Abraham et al [1] (PODC'19). Apart from these papers, let us mention that Hadzilacos-Halpern [15, Figure 1] also consider lower bounds for the communication complexity of authenticated broadcast. They measure the complexity over *optimistic* executions. That is, those in which the Environment corrupts no player. Notice that in Thm 4 we will *also* consider a complexity on average on two optimistic Environments (the one where all players have input 0, the other where all players have input 1). The difference being that [15, Figure 1] deals with the *message* complexity (and hold only in the deterministic setting). This measurement is, as we stressed above, *smaller* than the number of signatures complexity that we consider instead in Thm 4. Hence, they prove that the message complexity is linear under these optimistic environments.

Notice that in deterministic consensus, in which both players and the environment are deterministic, then an execution is *fully determined* by the Environment. This is why in this context we sometimes identity a given Environment $\mathcal{E}$ with the corresponding execution.

A.2.1 [12, Theorem 1]: ***deterministic*** broadcast has a quadratic $\frac{nt}{4}$ worst case complexity in the number of ***signatures*** (sent or forwarded by honest players)

Let us briefly recall the argument, that we trivially adapt to consensus here, because we will build on this version for Theorem 4. Let us consider the two executions following from the two "optimistic" Environments $\mathcal{E}_0$ and $\mathcal{E}_1$ which: leave all players honest, and give them all input 0 —resp. 1. Suppose by contradiction that, in *each* execution/Environment $\mathcal{E}_0$ and $\mathcal{E}_1$, honest players send $\leq \frac{nt}{4}$ signatures. For any player Q, note $A(Q)$ the set of players, cumulated over both executions/Environments $\mathcal{E}_0$ and $\mathcal{E}_1$, which: either receive a message with the signature of Q, or such that Q receives a message with the signature of one of them. This is by definition the set of players with which Q interacts. Then by assumption, there must exist at least one player p such that the cardinality $|A(p)| \leq t$. [Indeed notice that, since all players are honest, summing over every honest player p' the signatures *received* is then equal to summing over every honest player p' the total number of signatures sent]. Now, consider an Environment $\mathcal{E}_{01}^{A(p)}$ which: assigns input 0 to p and leaves him honest, corrupts all players in $A(p)$, and assigns 1 to the remaining honest players $U := \mathcal{P} - \{p\} - A(p)$. Then the Environment makes, on the one hand, players in $A(p)$ behave towards p as in the optimistic execution $\mathcal{E}_0$ where everyone has input 0. Thus p must ultimately output 1. And on the other hand, the Environment makes $A(p)$ behave towards U as if in the execution $\mathcal{E}_1$ where everyone has input 1. Thus players in U ultimately output 1. Notice that the Environment $\mathcal{E}_{01}^{A(p)}$ is able to carry out this strategy, since, by construction of the set $A(p)$, players in U never see the signature of p in $\mathcal{E}_1$, nor does p sees any signature from U in $\mathcal{E}_0$.

A.2.2 [1, Theorem 3]: assuming ***no digital signatures***, and that players ***only multicast*** messages, then ***randomized*** broadcast has a quadratic $\Omega(nt)$ worst case complexity in the number of messages sent by so-far honest players

Let us reformulate the argument for two reasons. First because we will build on it for our Theorem 4. Second, to emphasize that the existence of digital signatures (=standalone PKI) would break the argument. Which is something that we will repair in our synchronous Theorem 9 —at the price of considering *consensus* but not broadcast anymore. For simplicity we do not consider probabilities in the argument.

The idea of the proof is that honest player P_1 will be made "schizophrenic" and receive messages from two executions in parallel. He will not be able to distinguish what execution are playing the corrupted players and which one are playing the honest ones (depending whether he is under $\mathcal{E}'_0$ or $\mathcal{E}'_1$, to be defined). We consider a protocol for broadcast with sender P_2. Let us consider two baseline Environments $\mathcal{E}_0$ and $\mathcal{E}_1$, that have the following strategy in each execution. $\mathcal{E}_0$ gives input 0 to P_2, corrupts P_1 and simulates an execution in its head where P_2 would have input 1 instead. P_1 sends messages in the real execution as if he received *both* messages from the real execution, and from the simulated one. $\mathcal{E}_1$ follows a symmetric strategy: it gives input 0 to P_2, corrupts P_1 and simulates an execution in its head where P_2 would have input 0 instead. As before, P_1 sends messages in the real execution as if he received *both* messages from the real execution, and from the simulated one. Since the sender P_2 is honest under $\mathcal{E}_0$, by validity of broadcast, honest players ultimately output 0 in every execution. Likewise, since the sender P_2 is honest under $\mathcal{E}_1$, by validity of broadcast, honest players ultimately output 1 in every execution.

Now, consider Environments $\mathcal{E}'_0$ and $\mathcal{E}'_1$ as follows. $\mathcal{E}'_0$ initially gives input 0 to P_2 then simulates an execution in its head where P_2 would have input 1 instead. Whenever a player different from P_1 is going to multicast a message in the simulated execution, it adaptively corrupts it. Let C'_0 be the set of corrupted players so far. Players in C'_0 are instructed to send, in the real execution, the messages that they would have sent in the simulated execution, but, instead of multicasting them, they send it only to P_1. Apart from this, players in C'_0 also continue to play the real execution. $\mathcal{E}'_1$ has the symmetric strategy: initially gives input 1 to P_2 then simulates an execution in its head where P_2 would have input 0 instead. Whenever a player different from P_1 is going to multicast a message in the simulated execution, it adaptively corrupts it. Let C'_1 be the set of corrupted players so far. Players in C'_1 are instructed to send, in the real execution, the messages that they would have sent in the simulated execution, but, instead of multicasting them, they send it only to P_1. Apart from this, players in C'_1 also continue to play the real execution.

On the one side, consider honest players apart from P_1 under Environment E'_0. The set of executions restricted to them has an identical distribution to the set of executions under E_0 (they see a player P_1 which looks "schizophrenic", since his messages are also triggered from what he receives in the simulated execution, resp., from players in C'_0). Thus they must ultimately output the same value: 0. Likewise, honest players apart from P_1 under Environment E'_1, must ultimately ouput 1. In the middle, both Environments E'_0 and E'_1 provide the honest P_1 with equally distributed executions. But, under E'_0, P_1 must, on the one side, output 0 to be consistent with the other honest players. And on the one side, under E'_1, he must output 1 to be consistent with the other honest players. Thus, although going through equally distributed executions, he would have to output two different values, which is impossible.

A.2.2.1 Two limits of the previous argument

First, *assuming now that messages are digitally signed*, then Environments $\mathcal{E}_0$ and $\mathcal{E}_1$ would be unable to carry their strategy. Indeed, although P_1 is the only corrupted player, he is supposed to issue messages compatible with what he received in the simulated executions. For instance, the protocol, as played in the *simulated* execution under $\mathcal{E}_0$, is likely to require P_1 to multicast a message m', that would likely contain the *signed input value* $b = 1$ that he would have received from the sender P_2 in the simulated execution. Then, recall that the strategy of $\mathcal{E}_0$ is to make P_1 multicast message m' in the real execution. But actually, since P_2 did *not* produce his signature on the value $b = 1$ in the real execution, and is *not* corrupted, then $\mathcal{E}_0$ is unable to forge P_2's signature on $b = 1$, and so would *fail* to make P_1 send the desired valid message m' in the real execution.

Second, one could consider a fix for this and have the Environment $\mathcal{E}_0$ corrupt the sender P_2, in order to force him to sign the messages that P_1 needs to forward in the real execution (and likewise corrupt any other player whose signature is needed for P_1). The problem is that, since now the sender is *corrupted*, we would then *not* be able to conclude anymore that honest players need to output the specific value 0 under $\mathcal{E}_0$. The same problem occurs for $\mathcal{E}_1$. This issue is actually stressed in [1, §6] ("so we need to show a violation of consistency, not validity").

A.2.3 [12, Theorem 2]: *deterministic* broadcast has a quadratic $\frac{t^2}{4}$ worst case complexity in the number of *messages* (sent by honest players)

A.2.3.1 Highlights

The situation of [12, Theorem 2] is tricky in many aspects. (i) On the one hand, the only argument in the litterature by which we could be convinced is the one of…the *randomized* lower bound of [1, Theorem 1]. We adapt it to the deterministic synchronous setting, and highlight its specificities in brackets at two places of the revisited proof below. (ii) Then, we briefly discuss other proofs for [12, Theorem 2] in the litterature by which we could not be rigorously convinced. First, the original argument of [12, Theorem 2]. Second, the one sketched in the warmup of [1, §3]. We explain why in our opinion this alternative argument is incomplete, then provide a possible fix, at the cost of a suboptimal bound. (iii) In any case, none of the aforementionned arguments that we revisit carries over partial synchrony. Indeed, a player receiving no message *before* GST, is not supposed to output, even if he actually does not know when GST will happen.

[For *any* deterministic binary broadcast protocol Π, then *at least* one of the following two statements (9) and (10) is true:

$$\begin{gathered}\text{There exists a majority of players } q \text{ such that:}\\ C_0(\Pi, q) : \text{If}\,\{q \text{ receives no message}\}\,,\ \text{then}\ \{q \text{ ultimately outputs } 0\}\end{gathered} \tag{9}$$

$$\begin{gathered}\text{There exists a majority of players } q \text{ such that:}\\ C_1(\Pi, q) : \text{If}\,\{q \text{ receives no message}\}\,,\ \text{then}\ \{q \text{ ultimately outputs } 1\}\end{gathered} \tag{10}$$

This is trivial since every q is supposed to ultimately output some value, 0 or 1, in *any* execution, by the liveness condition of *synchronous* broadcast. Let us insist anyway on a point: in —many— broadcast protocols Π, we may well have that some player q, or even all players, *always* receives messages, in every execution. In which case the condition "If $\{q$ receives no message$\}$" is *empty* for this Π and q. And thus *both predicates* $C_0(\Pi, q)$ and $C_1(\Pi, q)$ are then trivially true.

Let us fix now a protocol Π, and assume without loss of generality that the statement (9) above holds. By the previous point, there a fortiori exists a set V of $\frac{t}{2}$ players, *not* containing the sender, such that: every honest $q \in V$ ultimately outputs 0 in executions —if any— in which he receives no message.]

Let us consider first the Environment $\mathcal{E}$ which gives input 1 to the sender and corrupts V. He instructs every player in V to behave honestly, except that they: ignore the first $\frac{t}{2}$ messages sent to each of them —i.e. they behave as if they did not receive them— and they never communicate with each other. By definition of broadcast, all honest players $\mathcal{P}\backslash V$ ultimately output the honest sender's value 1.

Let us now assume by contradiction that, for every Environment, the number of messages sent by honest players in Π is $\leq \dfrac{t^2}{4}$. In particular, under our specific $\mathcal{E}$, honest players would send $\leq \dfrac{t^2}{4}$ messages to V. And thus there exists some fixed player $p \in V$ which receives $\leq \dfrac{t}{2}$ messages from honest players. Note $A(p)$ the set of honest players sending messages to p under $\mathcal{E}$, possibly containing the sender, of cardinality at most $|A(p)| \leq \dfrac{t}{2}$. We can now consider the Environment $\mathcal{E}'$ that behaves the same than $\mathcal{E}'$ with the following differences: (i) it corrupts all players in V *except* p (ii) and also corrupts $A(p)$, to which he instructs *not* to send any message to p, but otherwise to behave honestly. Now: on the one hand, $\big[$ p ultimately outputs 0 because he belongs to V and does not receive any message $\big]$. On the other hand, consider the remaining honest players $U := \mathcal{P} - \{V \cup A(p)\}$. Their joint history under $\mathcal{E}'$ is undistinguishable from the one under $\mathcal{E}$, because, under $\mathcal{E}$, p *also* ignored the first $\dfrac{t}{2}$ messages that he received, and *did not* receive any other message by assumption. So players in U still ultimately output 0 under $\mathcal{E}'$.

A.2.3.2 Discussing other arguments in the litterature for this bound

To start with, consider the very end of the proof of [12, Theorem 2] in the original paper. The authors claim that p does not output at the end of the execution. We do not see why, since in our opinion, by definition of broadcast, the honest player p *should* anyway ultimately output some value. We now consider the alternative argument given in warmup of [1, §3]. At the very end the authors mention a "symmetric" scenario, where the sender sends 1 instead, in which the same player p would output 0. But we do not see actually why the Environment would be able to isolate the *same* player p in this symmetric scenario. One can manage to keep their overall argumentation by just fixing this point as follows, at the price of a suboptimal bound. Precisely, assume that only less than $\dfrac{t^2}{8}$ messages are sent in the worst case by honest players, instead of the previous $\dfrac{t^2}{4}$. Consider the two symmetric executions $\mathcal{E}_0$ and $\mathcal{E}_1$ in which the Environment corrupts the same arbitrary set V as before, and which differ only by the input value of the honest sender: 0 or 1. Then, as in our counting argument for [12, Theorem 1] above, there exists necessarily a player $p \in V$ which receives $\leq \dfrac{t}{2}$ messages in *both* those executions. We can now consider the variant executions $\mathcal{E}_0'$ and $\mathcal{E}_1'$ of $\mathcal{E}_0$ and $\mathcal{E}_1$, where, as before, the environment leaves p uncorrupted. The *same* p receives no message in both executions, so should ultimately output the *same* value b. We see that, whatever the value of b, this leads to a safety violation either in $\mathcal{E}_0'$ or in $\mathcal{E}_1'$.

A.2.4 [1, Theorem 1]: when the Environment is *strongly rushing*, then the previous [12, Theorem 2] extends to *randomized* broadcast

Strongly rushing means that, here, the Environment has the additional power to *withdraw* messages that were sent by honest players just before it corrupted them. They obtain that to achieve a probability of failure smaller than $1/2 + \epsilon$, then honest players need sending in expectation at least $(\epsilon t/2)^2$ messages.