Distributed and Parallel Databases (2024) 42:179–216
https://doi.org/10.1007/s10619-022-07410-0

Cob: a leaderless protocol for parallel Byzantine agreement in incomplete networks

Andrea Flamini[1] · Riccardo Longo[1] · Alessio Meneghetti[1]

Accepted: 8 April 2022 / Published online: 28 May 2022

Abstract
In this paper we extend the *Multidimensional Byzantine Agreement (MBA) Protocol*, a leaderless Byzantine agreement for lists of arbitrary values, into a protocol suitable for wide gossiping networks: *Cob*. This generalization allows the consensus process to be run by an incomplete network of nodes provided with (non-synchronized) same-speed clocks. Not all nodes are active in every step, so the network size does not hamper the efficiency, as long as the gossiping broadcast delivers the messages to every node in reasonable time. These network assumptions model more closely real-life communication channels, so Cob may be applicable to a variety of practical problems, such as blockchain platforms implementing sharding. Cob has the same Bernoulli-like distribution that upper-bounds the number of steps as the MBA protocol. We prove its correctness and security assuming a supermajority of honest nodes in the network, and compare its performance with Algorand.

Keywords Distributed consensus · Fault-tolerant protocols · Blockchain

1 Introduction

One of the main issues that blockchain platforms must deal with is the lack of *scalability*. Scalability is the ability of a platform to grow and manage an increasing number of requests. In particular, we say that a blockchain *scales* if it can easily adapt to changes in the number of users that decide to join in, as well as in the number of transaction requests that such users broadcast to the nodes maintaining the blockchain platform.

✉ Riccardo Longo
riccardolongomath@gmail.com

Andrea Flamini
andrea.flamini.1995@gmail.com

Alessio Meneghetti
alessio.meneghetti@unitn.it

1 Department of Mathematics, University of Trento, Povo, 38123 Trento, Italy

In order to solve the scalability issues of blockchain platforms, many approaches have been proposed over the years. Some of them are the *block size increase*, the use of *off-chain state channels*, *segregated witness (SegWit)* and *sharding*. Among the proposed approaches, sharding seems to be the most promising [1].

The term sharding comes from database management, where it identifies a particular type of database partitioning, that consist in dividing large databases into smaller parts, called shards. Shards are more manageable in terms of server hosting and other aspects of database maintenance, and allow to have faster query time by diversifying the responsibility of a database structure.

In the context of blockchain design, sharding consists of breaking the blockchain into small parts that are managed in parallel by node subsets, called *shards*. This augments throughput, since many transactions can be simultaneously validated, allowing blockchains to effectively scale for a huge number of users. Many blockchain platforms use sharding as a means to reach scalability, for example Ethereum 2.0 [2], Zilliqa [3] or EOS [4]. For a better description of blockchain sharding we refer to the survey of Meneghetti et al. [1]. Together with the security concerns regarding how to distribute the nodes among different shards (so that groups of cooperating malicious nodes are not assigned to the same shard), one of the main issues the protocol designer must deal with is the way the transactions validated by different shards can reach compatibility with one another. This problem is also referred to as the *reconciliation problem.*

Since blockchains are distributed ledgers, one of the core component is represented by the consensus protocol that the blockchain network must execute. The consensus protocol allows the nodes of the network to update their copy of the ledger in the very same way. Therefore, a blockchain implementing sharding might benefit from being able to bring the network to agreement (reach consensus) on which blocks are legitimately created by each shard, so that the network can proceed with the reconciliation of such transactions. One possible way to achieve this goal is to let the network execute a consensus protocol and decide which shards legitimately produced a valid block.

1.1 Cob protocol

In this paper we present *Cob*, a novel consensus protocol which efficiently solves the problem of reaching consensus on a set of blocks legitimately created by each shard. In particular we propose a viable solution for permissionless blockchain networks.

This problem can be easily extended to the following more general problem:

Problem 1.1 *Given a set of events which a network of nodes can observe, how can the nodes reach consensus on some relevant information about such events?*

In the context of blockchains implementing sharding the events to be observed are the creation and diffusion of a block by each shard; the relevant information about the event is the content of the block or some data which identifies it (e.g. the digest of the block computed via an hash function). In this paper we will describe Cob following the more general problem (i.e. Problem 1.1), however, the reader can keep in mind the specific application of reaching consensus over the shards.

Given a set of m events the nodes can observe, an instance of Cob requires every node in the network to build a list with m components. Each event (e.g. creation of a block of a shard) is associated to a component of the list and, once the nodes observe an event, they locally record in the corresponding component the relevant information about such event (e.g. the digest of the newly created block). This can be referred to as the *observation phase*.

After the observation phase, the nodes of the network will continue with a Cob protocol execution, exchanging messages until they reach consensus on a list of relevant information.

1.1.1 Cob: a parallel consensus protocol

We say that Cob is a *parallel* protocol since it is designed in a way that the consensus process is carried out simultaneously on each component of the list by every node involved in the consensus protocol. Every message broadcast by the nodes contains some information about each component, but the consensus achievement on each component is independent from the others.

In particular the nodes will exchange lists of values during the whole protocol execution. In the first 3 steps they will exchange lists of strings (the relevant information) and in the following steps they will exchange lists of bits in order to reduce the bandwidth required.

Agreement might be reached faster in some components, however the nodes will stop the protocol execution only when they realize that they agree on every component, therefore on the whole list.

If an event is not detected (e.g. a shard did not broadcast any block) or it is impossible to reach consensus (e.g. multiple blocks have been broadcast by the same shard), agreement will be reached on the special value $\perp$.

It is crucial that agreement is carried out in parallel on the list components instead of on the whole list, otherwise a widespread disagreement on a single component would affect the consensus achievement on the other components as well. Conversely, if agreement is carried out in parallel and independently, it is possible to preserve and finalize the agreed upon components, and to set to $\perp$ the controversial components.

1.1.2 Cob: a leaderless consensus protocol

Since anyone can join the network of a permissionless blockchain, we must consider an attacker that may try to disrupt the consensus process in several ways. Let us consider again the example of a blockchain implementing sharding. In this context, a malicious user may broadcast different information to different nodes advertising different blocks related to the same shard. This would cause a network partition in groups with different views about what the attacker has seen. Another attack it may perform is a censorship attack, pretending not to have received a block by a specific shard, setting the related component to $\perp$.

Many classical consensus protocols used by blockchain networks to record transactions [3, 5–8] (e.g. cryptocurrency transfers, smart contract execution requests) do not guarantee that a specific transaction will be included into the blockchain as soon as it is broadcast, in the block which is created right after the transaction diffusion.

In many cases this is just fine, in fact the transaction can be included into one of the following blocks after waiting a reasonable time interval.

For this reason, many consensus protocols are *leader based*, which means that there is a node which proposes a new block, and then the network decides whether to accept it or not. If the network decides to discard the leader proposal it elects a new leader and starts over. However, it is essential that the leader must change from time to time so that, if an attacker tries to undermine the liveness of the platform or to practice censorship, a (eventually honest) new leader will propose a new block (in one case), or will include the deliberately excluded transactions (in the other case).

The Problem 1.1 differs from the problem of (eventually) recording transaction requests. The events observed by the network must be discussed right after they are observed and the relevant information must be included in the agreed upon list or excluded once and for all.

Therefore, it is essential for our consensus protocol to be *leaderless*. In fact, a leader, if honest, would propose a list of relevant information which is heavily influenced by their own point of view (which in some cases might lead them to take incorrect decisions), and if the leader is malicious may easily perform censorship attacks refusing to include some information in the list, or deliberately include invalid information. In both cases, if the network does not agree even with one component proposed by the leader, it will reject the leader proposal, and this process is repeated until a leader proposes a list which gets accepted by a majority of the network. Note that this might not even happen, in fact, if there is a wide disagreement among the nodes about one or more components, there might not exist a list which is accepted by such majority.

So we have shown that there are many reasons that suggest to abandon the leader-based approach in favour of a leaderless approach. A leaderless consensus protocol, instead of questioning a single node, requires several nodes to share their own proposal. Then, based on these proposals, the protocol will bring the network to a consensus on a shared output as we will show in Sect. 3.2 and prove in Sect. 4.

We will show in Sect. 3.2 that in the first step of Cob, a set of randomly chosen nodes will share with the network their observed values, namely the list of relevant information (e.g. the hash of the shard blocks they have received). Starting from this information, the network will carry on with the consensus protocol to decide which components of the list can have a value, since the nodes of the network agree on some relevant information, and which will be set to $\perp$.

1.2 State of the art and considerations

In [9] is presented the MBA Protocol, a solution to Problem 1.1 for a relatively small network of a fixed number n of nodes, under some strong communication assumptions, and under the threat of an attacker that controls less than $\frac{1}{3}$ of the nodes. In particular, it is assumed a strongly synchronous communication model and a complete network, where every node could instantaneously send a message to each other. These assumption are unrealistic or dramatically reduce the possible application contexts.

In this paper we go a step further and define an analogous protocol called Cob, which works under more realistic assumptions and can be executed by a network of nodes of any size and the nodes communicate gossiping the messages broadcast in the network. This model makes Cob a viable solution to Problem 1.1 for a permissionless blockchain platform network. In fact, as we will see in Sect. 2.1, every step of the protocol is executed by a randomly selected set of nodes, whose cardinality is constant in expected value: this guarantees that, independently of the size of the network, the number of messages broadcast during each step will be constant.

As we have stated in Sect. 1.1.2, we are interested in leaderless consensus protocols which work under realistic network and communication assumptions.

There is no doubt that *asynchronous* BFT protocols would be the best solution for building high-assurance and resistant consensus protocols. Unlike *synchronous* or *weakly synchronous* protocols, whose liveness relies on communication assumptions, the asynchronous protocols do not put their liveness at risk.

In 2016, Miller et al. presented the leaderless asynchronous BFT protocol HoneyBadgerBFT [10], which significantly improved prior asynchronous BFT protocols [11–13]. HoneyBadgerBFT is based on the Asynchronous Common Subset (ACS) [11] implemented in combination with the asynchronous binary consensus protocol of Mostéfaoui et al. [14]. HoneyBadgerBFT can be used as a consensus protocol for blockchains, achieving a throughput of 200 KB/s of data appended to the ledger using 10 MB blocks (therefore it requires 5 min for each protocol run) using 104 participating servers. Later, in 2018, Duan et al. improved the HoneyBadgerBFT protocol presenting BEAT [15], a family of five asynchronous consensus protocols designed to meet different goals, such as different performance metrics (scalability, bandwidth or latency). Another improvement to HoneyBadgerBFT was proposed by Guo et al. with the Dumbo protocol [16].

However, these leaderless asynchronous BFT protocols have performance (latency, throughput) issues when they are executed by a number of replicas which exceeds the hundreds. For this reason, these protocols, or their variations, could be adopted only by permissioned or private blockchain platforms, because they are often controlled by relatively few nodes, but they do not provide a viable solution to the consensus problem in the context of permissionless networks. Moreover, these protocols must be executed by a fixed set of nodes who actively partake to the communication protocol. Therefore, these protocols can be incapable to guarantee resistance to targeted attacks that either compromise the servers involved or disconnect them from the network.

Since in this paper we are interested in solving a problem which can be applied to permissionless blockchain platforms implementing sharding, we assume that the number of nodes in the network can grow with no limit (potentially reaching millions of nodes) still guaranteeing the highest level of decentralization.

Therefore the asynchronous protocols mentioned above, classical primary-backup protocols such as PBFT [17] and other recent concurrent protocols such as Mir-BFT [18] or RCC [19] must be considered impractical for our use case due to the reduced and fixed number of actors involved in the protocol execution.

For this reason we must make some compromises, and find a solution which allows the implementation of a scalable platform settling for a protocol which relies on assumptions that are as weak as possible.

In this regard, Algorand [7] is a blockchain platform for cryptocurrency which adopts a BFT consensus protocol that faces three main challenges:

1. *avoid Sybil attacks*: this is done using *weighted users*. Every user is weighted based on the amount of money in their account, therefore as long as more than $\frac{2}{3}$ of the money is in honest hands, then the protocol is proven to be secure.
2. *can scale to millions of users*: this is achieved by choosing for every step a different committee, a small set of representative randomly selected from the set of users based on the users' weight.
3. *resistant to denial of service attacks or disconnection of users performed by an attacker*: in fact, relying on a committee which performs the operations, gives the possibility of targeted attacks against the chosen committee members. To prevent this, the protocol selects the committee members in a private and non interactive way via a verifiable random function (VRF) on the users' private key and some public data from the blockchain, a technique pioneered by Rabin in [20] which simulates a random lottery.
   Once a player realizes they are selected in a committee, then they must broadcast a message containing, among other things, the proof of their selection. At this point the attacker clearly knows about the selection and can try to corrupt such player. However, the nodes executing the protocol randomly change at every step and the attacker can not know in advance which node will be selected to broadcast a new message. This property is called *player replaceability* and protects the network from targeted attacks.

All these properties are achieved assuming that the attacker can corrupt or take control of any user in the network but, as we mentioned before, more than $\frac{2}{3}$ of the money (on which each user's weight is based) must always be in honest hands.

Algorand, to guarantee liveness, makes a strong synchrony assumption [21] requiring that almost every honest node (e.g. 95%), when it broadcasts a message, reaches almost every other honest node (e.g. 95%) within a predetermined time interval. The time is measured by each node by using *same speed clocks* which might not be synchronized (pointing to different times) as long as they have the same speed.

Moreover, to guarantee safety a weak synchrony assumption is used: the network can be asynchronous (i.e. controlled by an adversary) for a long period of time, as long as this time is bounded (e.g. 1 day, 1 week). After that the network must be synchronous for a reasonably long period of time (e.g. few hours, 1 day) in order to ensure safety [21].

In this paper we build on top of Algorand's network and communication model a consensus protocol which solves the problem of reaching consensus on a list of relevant information about some observed events. In fact, for our scope, it is essential to involve a high number of nodes to safely determine that this information has been correctly recorded, so we cannot rely on asynchronous protocols (due to their limitations) and must compromise on the network assumptions.

Outline In Sect. 2 we establish the preliminaries necessary to describe Cob. We define our network assumptions, we recall some useful notation, then we describe the sortition mechanism that selects which nodes are active in each step of the protocol, giving the necessary definitions. Finally we describe our assumptions on the honesty of the nodes.

In Sect. 3 we introduce the actual protocol, presenting a reference list of all the parameters and then describing in detail every step.

Then, in Sect. 4 we formally analyze the properties of Cob, proving that it is a Byzantine agreement through a series of preparatory lemmas and propositions. The main theorem also gives a probabilistic upper bound on the number of steps that are necessary to halt the execution.

In Sect. 5 we analyze the message complexity and the weight of the data broadcast in the network in each protocol execution. We also compare the performance of Cob with the one of Algorand's consensus protocol [7] in solving Problem 1.1 in the context of blockchain platforms implementing sharding.

Finally in Sect. 6 we draw some conclusions and remarks, and outline future works to improve the applicability of the protocol.

2 Preliminaries

In this section we define some assumptions, preliminary concepts, and notations that will be used later on to describe Cob and prove its properties.

2.1 Network assumptions

In complete networks, the number of messages exchanged through the network grows exponentially with the number of network participants, so for practical applications it is more convenient to consider a different network model, such as the *Asynchronous Gossiping Network* (AG networks)[1] presented by Micali in Algorand [7].

In this model messages are broadcast in the network in a gossiping fashion: a procedure characteristic of peer-to-peer communications where messages pass from one node to its neighbours and so on until they reach every node. In gossiping networks we rely on each member to pass messages along to its neighbours, therefore it is reasonable to envisage the network as an incomplete, connected and non-directed graph. We assume that a message sent by an honest node reaches every honest node within a time limit that depends on the size of the message itself. Since malicious nodes can behave arbitrarily, this assumption means that malicious nodes cannot be cut vertices in the network graph, that is the graph remains connected even without the edges connected to malicious nodes. We will also require that the ratio of malicious or faulty nodes is less than $\frac{1}{3}$.

In an AG network there does not exist a common clock, but we assume that all network participants are provided with *Same-Speed Clocks* [7]. In other words, we assume that each network participant has its own clock and that the clocks all have the same speed, even if they are not synchronized in any other way.

We take as time frame reference the earliest clock in the network, and suppose that the protocol execution starts at time 0, i.e. time starts when the first player begins the

[1] Algorand describes the environment in which it operates as asynchronous. This is because the communications between nodes happen via gossip and the protocol steps, which for a single user are non-overlapping time intervals, for different users may overlap due to asynchrony. However, since Algorand assumes that there exists a predetermined upper-bound on the time required by a message to reach (almost) every node, and therefore exists an upper-bound on the delay between different nodes, Algorand can not be considered an asynchronous protocol.

protocol execution. Moreover we assume that the discrepancy between any two clocks is at most a constant λ, that also upper bounds the time required to diffuse a "short" message of the protocol to the whole network (see Sect. 3.1), so each player will start the execution of the protocol at a time comprised in the interval $[0, \lambda]$. For example, this discrepancy could be observed in a scenario in which the first player triggers the start of the protocol execution, broadcasting a signal and resetting its clock, and then each player starts the execution (and resets its clock) when it receives this signal, with the network delay causing the discrepancies. Afterwards the time discrepancies do not vary because of the same-speed nature of the clocks.

2.2 A cryptographic sortition mechanism

In the protocol described in Sect. 3.2 not every player in the network is always active (i.e. authorized to broadcast messages), on the contrary at every step some players are selected to be active, while the others have a passive role. In order to better clear up this distinction, from now on in a specific step we will call *players* only the nodes selected to be active and broadcast their message, while a generic node of the network will be referred as a *user*. We will denote with $\mathtt{P}^{(s)}$ the set of players of step s.

We want this selection to be random, and furthermore we would like it to be *private* and performed without the aid of a trusted third party. With private we mean that each user should be able to privately check if it will be selected to be active (i.e. a player) in a step, and then be able to prove its selection to the other players. This concept is closely related to that of *verifiable random functions* (VRF), i.e. pseudo-random functions which provide publicly verifiable proofs of their outputs' correctness.

In our protocol the sortition is implemented through a cryptographic hash function H (modeled as a random oracle) and a digital signature scheme (G, S, V) with the uniqueness property, which is defined as follows.

Definition 2.1 (*Digital Signature Scheme with Unique Signature*) A digital signature scheme with unique signature is a triple of algorithms (G, S, V) such that:

- G is the key generation algorithm that outputs a secret key $\mathtt{sk}$ and a public key $\mathtt{pk}$;
- S is the signing algorithm, that given a message m and a private key $\mathtt{sk}$ outputs a signature $\sigma = S(\mathtt{sk}, m)$;
- V is the verification algorithm that given a message m, a signature σ and a public key $\mathtt{pk}$ outputs either $\mathtt{true}$ or $\mathtt{false}$, and such that:
  - the scheme is correct, i.e. for every $(\mathtt{sk}, \mathtt{pk})$ generated with G it holds

$$V(\mathtt{pk}, S(\mathtt{sk}, m), m) = \mathtt{true} \quad \forall m \ ;$$

  - the signature is unique, i.e. for any probabilistic polynomial time algorithm F that given a message outputs a public key $\hat{\mathtt{pk}}$ and two distinct signatures $\hat{\sigma} \neq \tilde{\sigma}$, we have that:

$$\mathbb{P}\left(V\left(\hat{\mathtt{pk}}, \hat{\sigma}, m\right) = V\left(\hat{\mathtt{pk}}, \tilde{\sigma}, m\right) = \mathtt{true}\right) < \varepsilon \quad \forall m$$

where ε is negligible. Note that this property holds also for public keys whose relative private key is known, and even for values $\hat{\texttt{pk}}$ that are not legitimately generated public keys.

Given this definition, we can now describe the sortition of the active players in each step of the protocol.

Definition 2.2 (*Sortition Mechanism*) Let (G, S, V) be a digital signature scheme with unique signature, and suppose that every user $1 \leq i \leq N$ is identified by a public key $\texttt{pk}_i$, let r be a random string independent from $\texttt{pk}_i$ for every $1 \leq i \leq N$, and suppose that every user knows r and $\{\texttt{pk}_i\}_{1 \leq i \leq N}$. Moreover let $n \leq N$ be the desired number of players during each step s of the protocol, let $H : \{0,1\}^* \longrightarrow \{0,1\}^d$ be a hash function, and let $\phi : \{0,1\}^d \longrightarrow (0,1]$ be the standard decoding of a bit string into the unit interval $\phi(h) = \frac{1+\sum_{i=0}^{d-1} h_i 2^i}{2^d}$.

User i is selected to be a player during step s of the protocol, i.e. $i \in \texttt{P}^{(s)}$, if:

$$\phi\left(H\left(\sigma_i^{(s)}\right)\right) \leq \frac{n}{N} \quad \bigwedge \quad V\left(\texttt{pk}_i, \sigma_i^{(s)}, H(r \| s)\right) = \texttt{true} .$$

Where $\sigma_i^{(s)} = S(\texttt{sk}_i, s \| r)$. The signature $\sigma_i^{(s)}$ can then be used to prove that $i \in \texttt{P}^{(s)}$.

Note that, when H is modeled as a random oracle, $\phi\big(H\big(\sigma_i^{(s)}\big)\big)$ is uniformly distributed, so the probability of a player to be selected is $\frac{n}{N}$, and the expected number of active players is indeed n.

Note that the same sortition mechanism can be implemented with a weaker notion of signature scheme that allows signing failures. That is, a scheme where the output of S is a special symbol $\perp$ with fixed probability f (supposing the message to be uniform in $\{0,1\}^d$), with $V(\texttt{pk}, \perp, m) = \texttt{false}$ for every $\texttt{pk}$ and m. The sole adjustment required is to increase the threshold $\frac{n}{N}$ to account for the signing failure, using $\frac{n}{N(1-f)}$ instead.

In some applications it might be desirable that players are selected with nonuniform probability. There is a simple trick to adjust the selection probability for each player: let $p \in (0,1]$ be a fixed probability and $t_i \geq 1$ be a publicly known threshold for player i. The tweaked process selects player i if it can provide a pair signature-counter $\left(\sigma_i^{(s)}, c_i\right)$ such that $\phi\left(H\left(\sigma_i^{(s)}\right)\right) \leq p$, $V\left(\texttt{pk}_i, \sigma_i^{(s)}, H(r \| s \| c_i)\right) = \texttt{true}$, and $c_i \leq t_i$. In other words the player i has t_i attempts to produce a *winning* signature, so its probability to be selected is $1-(1-p)^{t_i}$.

2.3 Sortition assumptions

Similarly to the MBA protocol [9], Cob requires that the number of honest nodes at each step s is more than two times the number of malicious players active at step s. Since the active players are randomly selected, we require that at each step there are enough active honest players with high probability.

We now define a probabilistic concept which will be widely used throughout this paper.

Definition 2.3 We will write that an event E happens with *overwhelming probability* if $P(E) \geq 1 - \epsilon$, where $\epsilon \in (0, 1)$ is a parameter sufficiently close to 0.

A good choice for practical applications could be $\epsilon = 10^{-12}$.

Definition 2.4 Let n be the expected number of active players in a step, we define the threshold $\mathtt{t_H} = \lfloor \frac{2n}{3} \rfloor + 1$. For every step s we choose the parameter n in a way that the following relationships between the number of honest players $\mathtt{HP}^{(s)}$ and the number of malicious players $\mathtt{MP}^{(s)}$ hold with overwhelming probability:

1. $|\mathtt{HP}^{(s)}| > \mathtt{t_H}$;
2. $|\mathtt{HP}^{(s)}| + 2|\mathtt{MP}^{(s)}| < 2\mathtt{t_H}$.

Note that these two conditions imply that $\mathtt{HP}^{(s)} > 2\mathtt{MP}^{(s)}$. In practice, they imply that with overwhelming probability:

– the protocol has, at each step, the required $\frac{2}{3}$ honest majority of players.
– at every step there is a sufficient number of honest players who can certify a new list or finalize a list component;
– two distinct nodes can not finalize the same component with two distinct values.

Note that the closer to 1 the ratio of honest users in the network is, the smaller the number of players for each step needs to be.

The parameter choice necessary to meet the requirements is done using variants of Chernoff bounds, as in Algorand [7] and the analysis of such bound can be found in [22].

2.4 Notation

We will typeset lists in boldface and in general subscript will be used to denote the player who created the value and the index of list components, while superscripts will refer to the protocol step in which the value has been produced. So $\mathbf{v}_i^{(s)}$ will be a list created by user i during step s of the protocol, while $v_{i,c}^{(s)}$ will denote the c-th component of said list. As shorthand, $\mathbf{1}$ denotes a list where each component is equal to 1, and similarly $\mathbf{0}$ denotes an all-zero list.

As in [7] and [9], the notation $\#_i^{(s)}(v, c)$, for $1 \leq c \leq m$ represents the number of players from which player i has received during step s a valid message containing a list $\mathbf{v}^{(s)} = (v_1, \ldots, v_m)$ such that $v_c = v$ considering, possibly, also its own message. We recall that honest players consider at most one message from player j as valid (discarding all contrasting and not properly formatted messages, and counting identical messages as one), so only *valid* messages are considered and counted, and $\sum_v \#_i^{(s)}(v, c) \leq |\mathtt{P}^{(s)}| \quad \forall i, s, c$.

In the protocol the players try to reach agreement on a list of arbitrary values, where each player j starts the protocol knowing an m-dimensional list $\mathbf{v}_j = \left(v_{j,1}, \ldots, v_{j,m}\right) \in V = \prod_{c=1}^{m} V_c$. We say that the players have reached *c-agreement*, where $1 \leq c \leq m$ is a specific component, when there exists $v \in V_c$ such that for every honest player j, $v_{j,c} = v$. When c-agreement is reached on all the components of the list, we have that for all honest players i, j, $\mathbf{v}_i = \mathbf{v}_j$, hence also agreement is reached.

3 Cob protocol

We now present Cob, a protocol that allows a wide gossiping network to reach agreement on a list of arbitrary values. The properties of the protocol will be formally stated and proved in Sect. 4.

3.1 Protocol parameters and components

For the sake of clarity and easy reference, we now provide a list with the definition of the parameters and the notation that we will use to describe and analyze Cob:

- H: a cryptographic hash function, modelled as a random oracle;
- (G, S, V): a digital signature scheme with unique signature (see Definition 2.1);
- $N \in \mathbb{Z}^+$: the number of nodes in the network, i.e. the users of the protocol;
- $\mathtt{PK}$: the set of public keys of the users, each user i, with $1 \leq i \leq N$, is univocally identified by its public key $\mathtt{pk}_i \in \mathtt{PK}$ and has a private key $\mathtt{sk}_i$;
- $\frac{2}{3} < h \leq 1$: the ratio of the honest users in $\mathtt{PK}$;
- r: a reference string, i.e. a random string independent from every $\mathtt{pk}_i \in \mathtt{PK}$ and known by every user;
- n: the expected number of players active in each step of the protocol;
- $\mathtt{t}_\mathtt{H} = \lfloor \frac{2n}{3} \rfloor + 1$: a threshold used in the protocol, derived from the expected lower bound of the number of honest players in each step;
- m: the number of components of the list of arbitrary values upon which agreement has to be achieved, and which is common knowledge since the nodes know the events they must observe and describe on the ledger;
- $V = \prod_{c=1}^{m} V_c$: the set the list to be agreed upon belongs to, each set V_c contains all the possible values of the c-th component of said list;
- $\perp$: a special value that represents a *non meaningful value* for any component, we require that $\perp \in V_c$ for every $1 \leq c \leq m$;
- $\mathbf{O}_i \in V$: the list built by player i at the start of the protocol, for every $1 \leq c \leq m$ we will say that c is an *unambiguous component* if $O_{i,c} = O_{j,c}$ for every couple of honest users i, j, otherwise, if there exist two honest users i and j such that $O_{i,c} \neq O_{j,c}$, we will say that c is an *ambiguous component*;
- Ω: the amount of time spent by each player i at the start of the protocol to build its private list $\mathbf{O}_i$ (e.g. by observing some events and reporting some relevant information about them), the protocol will then try to reconcile all these lists into a shared one;
- Λ: the upper bound to the time needed to propagate the messages in each of the first two steps of the protocol;
- λ: the upper bound to the time needed to propagate the messages of the third and following steps of the protocol. The difference between λ and Λ depends on the size of the elements of V. We assume that $\Lambda = \mathcal{O}(\lambda)$;
- $s \in \mathbb{Z}^+$: the current step of the protocol;
- $\mathtt{P}^{(s)}$: the active players that partake in step s of the protocol;
- $\sigma_i^{(s)} = S(\mathtt{sk}_i, H(r \| s))$: the credential of user i for step s, used to check if $i \in \mathtt{P}^{(s)}$;

- $p \in (0, 1)$: for each time-slot s, each user in $\texttt{PK}$ is chosen to be in $\texttt{P}^{(s)}$ with probability $p = \frac{n}{N}$;
- $\texttt{MP}^{(s)}$ and $\texttt{HP}^{(s)}$: they are respectively the set of malicious and honest players in step s, note that $\texttt{MP}^{(s)} \cup \texttt{HP}^{(s)} = \texttt{P}^{(s)}$ and $\texttt{MP}^{(s)} \cap \texttt{HP}^{(s)} = \emptyset$;
- $\texttt{m}_i^{(s)}$: the message broadcast by player i during step s;
- $\mathbf{v}_i^{(s)}$: the list of information contained in the message $\texttt{m}_i^{(s)}$, we will see that $\mathbf{v}_i^{(s)} \in V$ if $s \leq 2$, $\mathbf{v}_i^{(s)} \in \{0, 1\}^m$ if $s \geq 3$;
- $\texttt{sig}_i^{(s)}(x) = (x, S(\texttt{sk}_i, s\|x))$: the value x broadcast by player i during step s certified by its signature, it is included in $\texttt{m}_i^{(s)}$;
- C_i: the certificate built by player i which attests that the final list has network agreement, each user i continues running the protocol until it can build a certificate C_i;
- $\alpha_i \in [0, \lambda]$: the time at which user i starts the execution of each step of the protocol;
- $\beta_i^{(s)}$: the time at which user i ends the execution of step s of the protocol;
- $t^{(s)}$: the amount of time that players of step s have to wait in order to harvest all the information required to compute the message to broadcast, if $i \in \texttt{P}^{(s)}$ then $t^{(s)} = \beta_i^{(s)} - \alpha_i$;
- T_c: the time at which the first honest user finalizes component c;
- T: the time at which the first honest user produces a certificate;
- L: a random variable representing the number of Bernoulli trials needed to see the output 1, when each trial outputs 1 with probability $\frac{h}{2}$;
- $\chi_{l,\frac{h}{2}}$: a random variable representing the number of steps required to end the probability game described in [9] with parameters l and $\frac{h}{2}$. The probability game consists of flipping l distinct but equal coins (which flip heads with probability $\frac{h}{2}$) until each of them flipped head at least once. Its probability distribution is computed in [9] but is also reported in Sect. 5.1.

3.2 Cob protocol description

We now describe in detail how Cob works. The honest users will be the ones who follow the protocol described below, and, even if not elected as players on any step, they are supposed to stay online to support message propagation during the whole protocol execution.

The protocol is a variant of the MBA, where the first three steps are essentially the Multidimensional Graded Consensus presented in [9], then from the step 4 onward it is a three-step loop that corresponds to the Multidimensional Binary Byzantine Algorithm, also presented in [9].

As in the protocol MBA, each user i privately saves a list $\mathbf{f}_i$ initialized to $\mathbf{0}$ that keeps track of the finalization of the components. A component is finalized when the network is in agreement on it, and from that moment on the protocol will not change it anymore. Once every component has been finalized, the protocol enables the creation of certificates that attest that the list is indeed shared by the network, and then terminates. That is, the three-step loop is repeated until the ending condition is met, which corresponds to the creation of a certificate for the agreed-upon final list.

Every honest user i in the system starts the protocol execution when its own private clock signs 0. Note that, right from the start, each user i can build its credentials $\sigma_i^{(s)}$ and check for which s it will be $\phi\big(H\big(\sigma_i^{(s)}\big)\big) \leq p$ and therefore $i \in \mathtt{P}^{(s)}$.

We now describe Cob, followed by the Ending Condition to be performed in each step $s \geq 4$ to determine whether agreement has been achieved.

STEP 1 *(first step of m-dimensional GC)*

- Each user i computes its credential $\sigma_i^{(1)}$ and checks if $i \in \mathtt{P}^{(1)}$;
- if $i \notin \mathtt{P}^{(1)}$ then i ends its step 1 right away;
- if $i \in \mathtt{P}^{(1)}$, i spends $t^{(1)} = \Omega$ time building its own private list $\mathbf{O}_i \in V$, then:
  - sets $\mathbf{v}_i^{(1)} = \mathbf{O}_i$;
  - broadcasts the message:

$$\mathtt{m}_i^{(1)} = \left(1, \sigma_i^{(1)}, \mathtt{sig}_i^{(1)}\left(\mathbf{v}_i^{(1)}\right)\right) .$$

STEP 2 *(second step of m-dimensional GC)*

- Each user i computes its credential $\sigma_i^{(2)}$, if $i \notin \mathtt{P}^{(2)}$ then i ends its step 2;
- if $i \in \mathtt{P}^{(2)}$, after waiting an amount of time $t^{(2)} = t^{(1)} + \Lambda + \lambda$, player i does the following:
  - sets $\mathbf{v}_i^{(2)}$, where $v_{i,c}^{(2)} = v_c \neq \perp$ if and only if $\#_i^{(1)}(v_c, c) \geq \mathtt{t}_{\mathtt{H}}$, and $v_{i,c}^{(2)} = \perp$ otherwise;
  - broadcasts the message:

$$\mathtt{m}_i^{(2)} = \left(2, \sigma_i^{(2)}, \mathtt{sig}_i^{(2)}\left(\mathbf{v}_i^{(2)}\right)\right) .$$

STEP 3 *(output determination of m-dimensional GC and starting broadcast in m-dimensional BBA)*

- Each user i collects and locally saves the messages received from the players of step 2;
- after waiting an amount of time $t^{(3)} = t^{(2)} + \lambda + \Lambda$, i updates its private list $\mathbf{O}_i$ and computes the list $\mathbf{g}_i$ where, for each component c, $O_{i,c}$ and $g_{i,c}$ are computed as follows:
  - if, for some $x \neq \perp$, $\#_i^{(2)}(x, c) \geq \mathtt{t}_{\mathtt{H}}$, then $\left(O_{i,c}, g_{i,c}\right) = (x, 2)$;

- else, if, for some $x \neq \perp$, $\#_i^{(2)}(x, c) \geq \frac{\mathtt{t}_\mathtt{H}}{2}$, then $\left(O_{i,c}, g_{i,c}\right) = (x, 1)$;
- otherwise, $\left(O_{i,c}, g_{i,c}\right) = (\perp, 0)$;

- if $i \notin \mathtt{P}^{(3)}$, then i ends the execution of step 3;
- if $i \in \mathtt{P}^{(3)}$, then player i does the following:
  - builds the list $\mathbf{v}_i^{(3)} \in \{0, 1\}^m$ such that $v_{i,c}^{(3)} = 0$ if $g_{i,c} = 2$, $v_{i,c}^{(3)} = 1$ otherwise;
  - computes the list $\mathbf{\Theta}_i^{(3)}$ such that $\Theta_{i,c}^{(3)} = \perp$ if $v_{i,c}^{(3)} = 1$, $\Theta_{i,c}^{(3)} = O_{i,c}$ when $v_{i,c}^{(3)} = 0$;
  - broadcasts the message:

$$\mathtt{m}_i^{(3)} = \left(3, \sigma_i^{(3)}, \mathtt{sig}_i^{(3)}\left(\mathbf{v}_i^{(3)}\right), \mathtt{sig}_i^{(3)}\left(H\left(\mathbf{\Theta}_i^{(3)}\right)\right)\right) .$$

***STEP* s** $4 \leq s, s - 1 \equiv 0 \mod 3$ *(Coin-Fixed-To-0 step and starting broadcast of Coin-Fixed-To-1 step in m-dimensional BBA)*

- Each user i collects the messages received from the players active during step $s - 1$;
- after waiting an amount of time $t^{(s)} = t^{(s-1)} + 2\lambda$, the user i starts building the list $\mathbf{v}_i^{(s)}$ performing the following operations:
  - verifies the ENDING CONDITION;
  - sets $v_{i,c}^{(s)} = v_{i,c}^{(s-1)}$ for all $1 \leq c \leq m$ such that $f_{i,c} = 1$;
  - performs the FINALIZATION CHECK 0;
  - performs the FINALIZATION CHECK 1;
- if $i \notin \mathtt{P}^{(s)}$, the user i ends the execution of step s;
- if $i \in \mathtt{P}^{(s)}$, the player i does the following:
  - completes the list $\mathbf{v}_i^{(s)}$ depending on the lists $\mathbf{v}_j^{(s-1)}$ included in the valid messages it has received, in particular, for each component c such that $f_{i,c} = 0$:
    - if $\#_i^{(s-1)}(1, c) \geq \mathtt{t}_\mathtt{H}$, then i sets $v_{i,c}^{(s)} = 1$;
    - else i sets $v_{i,c}^{(s)} = 0$;
  - computes the list $\mathbf{\Theta}_i^{(s)}$ such that $\Theta_{i,c}^{(s)} = \perp$ when $v_{i,c}^{(s)} = 1$, $\Theta_{i,c}^{(s)} = O_{i,c}$ when $v_{i,c}^{(s)} = 0$;
  - broadcasts the message:

$$\mathtt{m}_i^{(s)} = \left(s, \sigma_i^{(s)}, \mathtt{sig}_i^{(s)}\left(\mathbf{v}_i^{(s)}\right), \mathtt{sig}_i^{(s)}\left(H\left(\mathbf{\Theta}_i^{(s)}\right)\right)\right) .$$

***STEP* s** $5 \leq s, s - 1 \equiv 1 \mod 3$ *(Coin-Fixed-To-1 step and starting broadcast of Coin-Genuinely-Flipped step in m-dimensional BBA)*

- Each user i collects the messages received from the players active during step $s-1$;
- after waiting an amount of time $t^{(s)} = t^{(s-1)} + 2\lambda$, the user i starts building the list $\mathbf{v}_i^{(s)}$ performing the following operations:
  - verifies the ENDING CONDITION;
  - sets $v_{i,c}^{(s)} = v_{i,c}^{(s-1)}$ for all $1 \le c \le m$ such that $f_{i,c} = 1$;
  - performs the FINALIZATION CHECK 0;
  - performs the FINALIZATION CHECK 1;
- if $i \notin \mathtt{P}^{(s)}$, the user i ends the execution of step s;
- if $i \in \mathtt{P}^{(s)}$, the player i does the following:
  - completes the list $\mathbf{v}_i^{(s)}$ depending on the lists $\mathbf{v}_j^{(s-1)}$ included in the valid messages it has received, in particular, for each component c such that $f_{i,c} = 0$:
    - if $\#_i^{(s-1)}(0,c) \ge \mathtt{t}_\mathtt{H}$, then i sets $v_{i,c}^{(s)} = 0$;
    - else i sets $v_{i,c}^{(s)} = 1$;
  - computes the list $\mathbf{\Theta}_i^{(s)}$ such that $\Theta_{i,c}^{(s)} = \perp$ when $v_{i,c}^{(s)} = 1$, $\Theta_{i,c}^{(s)} = O_{i,c}$ when $v_{i,c}^{(s)} = 0$;
  - broadcasts the message:

$$\mathtt{m}_i^{(s)} = \left(s, \sigma_i^{(s)}, \mathtt{sig}_i^{(s)}\left(\mathbf{v}_i^{(s)}\right), \mathtt{sig}_i^{(s)}\left(H\left(\mathbf{\Theta}_i^{(s)}\right)\right)\right) .$$

***STEP* s** $6 \le s,\ s-1 \equiv 2 \pmod 3$ *(Coin-Genuinely-Flipped step and starting broadcast of Coin-Fixed-To-0 step in m-dimensional BBA)*

- Each user i collects the messages received from the players active during step $s-1$;
- after waiting an amount of time $t^{(s)} = t^{(s-1)} + 2\lambda$, the user i starts building the list $\mathbf{v}_i^{(s)}$ performing the following operations:
  - verifies the ENDING CONDITION;
  - sets $v_{i,c}^{(s)} = v_{i,c}^{(s-1)}$ for all $1 \le c \le m$ such that $f_{i,c} = 1$;
  - performs the FINALIZATION CHECK 0;
  - performs the FINALIZATION CHECK 1;
- if $i \notin \mathtt{P}^{(s)}$, the user i ends the execution of step s;
- if $i \in \mathtt{P}^{(s)}$, the player i does the following:
  - completes the list $\mathbf{v}_i^{(s)}$ depending on the lists $\mathbf{v}_j^{(s-1)}$ included in the valid messages it has received, in particular, for each component c such that $f_{i,c} = 0$:
    - if $\#_i^{(s-1)}(0,c) \ge \mathtt{t}_\mathtt{H}$, then i sets $v_{i,c}^{(s)} = 0$;
    - if $\#_i^{(s-1)}(1,c) \ge \mathtt{t}_\mathtt{H}$, then i sets $v_{i,c}^{(s)} = 1$;

- otherwise, letting $\mathtt{P}_i^{(s-1)} \subseteq \mathtt{P}^{(s-1)}$ be the set of players who sent i a valid message in the previous step, then i sets $v_{i,c}^{(s)} = k_c$, where $k = H\left(\min_{j \in \mathtt{P}_i^{(s-1)}} H\left(\text{œ}_j^{(s-1)}\right)\right)$;

  we will refer to the player whose hashed credential is minimal from i's point of view as the *coin flipper* selected by i during step s;
- computes the list $\mathbf{\Theta}_i^{(s)}$ such that $\Theta_{i,c}^{(s)} = \perp$ when $v_{i,c}^{(s)} = 1$, $\Theta_{i,c}^{(s)} = O_{i,c}$ when $v_{i,c}^{(s)} = 0$;
- broadcasts the message:

$$\mathtt{m}_i^{(s)} = \left(s, \sigma_i^{(s)}, \mathtt{sig}_i^{(s)}\left(\mathbf{v}_i^{(s)}\right), \mathtt{sig}_i^{(s)}\left(H\left(\mathbf{\Theta}_i^{(s)}\right)\right)\right) .$$

ENDING CONDITION

If, while user i waits for the end of the current step (step s), there exist a string $\theta \in \{0,1\}^d$ and a step s' such that:

- $4 \le s'$ with $s' - 1 \equiv 0 \mod 3$;
- user i has received at least $\mathtt{t_H}$ messages $\mathtt{m}_j^{(s'-1)}$ containing the signature of $s' - 1 \| \theta = s' - 1 \| H\left(\mathbf{\Theta}_j^{(s'-1)}\right)$ and at least $\mathtt{t_H}$ messages $\mathtt{m}_j^{(s')}$ containing the signature of $s' \| \theta = s' \| H\left(\mathbf{\Theta}_j^{(s')}\right)$;

then i can build its certificate C_i, diffuse it in the network and terminate its execution of the protocol.

For the collision resistance of H we can assume that there is a list $\mathbf{\Theta} \in V$ such that $\theta = H(\mathbf{2})$, i.e. i has collected $2\mathtt{t_H}$ signatures that refer to the same list $\mathbf{\Theta}$. Lemma 4.2 states that in this case i has received, for each $1 \le c \le m$, at least a message $\mathtt{m}_j^{(2)}$ with $v_{j,c}^{(2)} = \Theta_c$, hence it can reconstruct the list $\mathbf{\Theta}$. Let $\hat{\mathtt{P}}^{(s-1)} \subseteq \mathtt{P}^{(s-1)}$ and $\hat{\mathtt{P}}^{(s)} \subseteq \mathtt{P}^{(s)}$ be the the sets of players (each of cardinality at least $\mathtt{t_H}$) that have sent messages with the signature of $H(\mathbf{\Theta})$ as stated above, then i can build its certificate as:

$$C_i = \left(\mathbf{\Theta}, s, \left\{\left(\mathtt{sig}_j^{(s-1)}\left(H\left(\mathbf{\Theta}\right)\right), \sigma_j^{(s-1)}\right)\right\}_{j \in \hat{\mathtt{P}}^{(s-1)}}, \left\{\left(\mathtt{sig}_j^{(s)}\left(H\left(\mathbf{\Theta}\right)\right), \sigma_j^{(s)}\right)\right\}_{j \in \hat{\mathtt{P}}^{(s)}}\right) .$$

FINALIZATION CHECK 0

Let s' be a step such that $4 \le s' \le s$ and $s' - 1 \equiv 0 \mod 3$ (that is, step s' is a Coin-Fixed-To-0 step).

For each component $c \in \{1, \ldots, m\}$ such that $f_{i,c} = 0$, if, considering the lists $\mathbf{v}_j^{(s'-1)}$ contained in the valid messages $\mathtt{m}_j^{(s'-1)}$ received by i, we have that $\#_i^{(s'-1)}(0, c) \ge \mathtt{t_H}$, then i sets:

- $v_{i,c}^{(s)} = 0$;
- $f_{i,c} = 1$.

FINALIZATION CHECK 1

Let s' be a step such that $4 \leq s' \leq s$ and $s' - 1 \equiv 1 \mod 3$ (that is, step s' is a Coin-Fixed-To-1 step).

For each component $c \in \{1, \ldots, m\}$ such that $f_{i,c} = 0$, if, considering the lists $\mathbf{v}_j^{(s'-1)}$ contained in the valid messages $\mathtt{m}_j^{(s'-1)}$ received by i, we have that $\#_i^{(s'-1)}(1, c) \geq \mathtt{t}_\mathtt{H}$, then i sets:

- $v_{i,c}^{(s)} = 1$;
- $f_{i,c} = 1$.

User i keeps following the protocol instruction until the ending conditions are satisfied and i is able to build a certificate C_i. A certificate C_i contains the list $\mathbf{\Theta}$ on which the network has reached agreement, alongside a set of digital signatures for $H(\mathbf{\Theta})$ together with a proof that those who signed were indeed players of a specific step. In particular C_i contains at least $\mathtt{t}_\mathtt{H}$ signatures from players of a Coin-Genuinely-Flipped step s and $\mathtt{t}_\mathtt{H}$ signatures from players of the subsequent subsequent Coin-Fixed-To-0 step $s + 1$.

4 Security analysis

We now outline the security proof for the protocol Cob.
In the main theorem, Theorem 4.1, we determine:

1. an upper bound to the time needed by the first honest node to produce a certificate for the agreed upon list $\mathbf{\Theta}$;
2. the time interval when every honest user gets to know $\mathbf{\Theta}$.

In order to prove this, in Sect. 4.1 we show some preliminary results characterising the gossiping communications under our assumptions, and prove a lemma that justifies the construction of a certificate as described in Sect. 3.2.

In Sect. 4.2 we prove some propositions and lemmas regarding the time needed by a network of nodes to reach consensus on the single list components. In particular we distinguish two cases:

1. if the nodes observed unambiguous events (i.e. the honest nodes agree on the same value at the beginning of the protocol) or very ambiguous events (which means that there is not a majority of the nodes who observed the same value), then they will reach c-agreement not later than step 5;
2. otherwise, the number of steps required to reach c-agreement is upper bounded by $3L + 5$, where L is a Bernoulli-like random variable with parameter $\frac{h}{2} > \frac{1}{3}$.

Finally, we use the results above to prove that the nodes of the network will be in possession of a certificate for the agreed upon list $\boldsymbol{\Theta}$ within a number of steps upper-bounded by $5 + 3\chi_{\ell, \frac{h}{2}}$, and that only one list can be certified.

Theorem 4.1 (Main Theorem) *Given an instance of Cob, described in Sect. 3.2, the following properties about each protocol execution hold with overwhelming probability:*

1. *if* $0 \leq \ell \leq m$ *is the number of ambiguous components, then we have that* $T \leq \Omega + 2\Lambda + \left(7 + 6\chi_{\ell, \frac{h}{2}}\right)\lambda$, *where* $\chi_{\ell, \frac{h}{2}}$ *is the random variable described in [9];*
2. *all honest users agree on the same list* $\boldsymbol{\Theta}$ *and know it in the interval* $[T, T + \lambda]$.

Remark 4.1 When there are no ambiguous components, i.e. all honest nodes at the beginning of the protocol build the same list, then we have $\chi_{\ell, \frac{h}{2}} = 0$, so $T \leq \Omega + 2\Lambda + 7\lambda$.

4.1 Preliminary results

In order to prove our Main Theorem 4.1, we first need to prove some preliminary lemmas and propositions which characterize Cob under our communication and network model.

Lemma 4.1 *For each step* $s \geq 1$ *of a protocol run we have:*

1. *if* $i \in \mathtt{P}^{(s)}$ *is honest, then* $\beta_i^{(s)} \in [t^{(s)}, t^{(s)} + \lambda]$*;*
2. *if* $i \in \mathtt{P}^{(s)}$ *is honest, then by time* $\beta_i^{(s)}$ *it has received all messages sent by all honest players* $j \in \mathtt{HP}^{(s')}$ *for all steps* $s' < s$*;*
3. *for each step* $\bar{s} > s$*, fixing a component* $1 \leq c \leq m$*, with overwhelming probability there do not exist two players* $i, i' \in \mathtt{P}^{(\bar{s})}$ *such that:*
   - *i has received at least than* $\mathtt{t_H}$ *messages* $\mathtt{m}_j^{(s)}$ *advertising* $v_{j,c}^{(s)} = b$*;*
   - *i' has received at least than* $\mathtt{t_H}$ *messages* $\mathtt{m}_j^{(s)}$ *advertising* $v_{j,c}^{(s)} = b'$ *with* $b' \neq b$.

   Note that for step 2 and 3 we have $b, b' \in V_c$*, while for the next steps* $b, b' \in \{0, 1\}$.

Proof Property 1 holds as a consequence of the network assumptions regarding the same speed clocks delay. In fact we know that each user i starts its protocol execution at a time $\alpha_i \in [0, \lambda]$ and waits for time $t^{(s)}$ before acting and then ending its step. This means that $\beta_i^{(s)} = \alpha_i + t^{(s)} \in [t^{(s)}, t^{(s)} + \lambda]$.

Property 2 holds by the definition of the protocol, noticing that $t^{(s)} \geq t^{(s')}$ for all $s' < s$. If $s = 2, 3$, then for all $s' < s$, $i \in \mathtt{P}^{(s)}$, we have that

$$\beta_i^{(s)} \geq t^{(s)} = t^{(s-1)} + \lambda + \Lambda \geq t^{(s')} + \lambda + \Lambda \geq \beta_j^{(s')} + \Lambda,$$

since the honest players $j \in \mathtt{HP}^{(s')}$ send their messages of step s' at time $\beta_j^{(s')}$ and the messages reach all honest users in at most Λ time, then player i has received all the messages from honest players of the previous steps.

If $s \geq 4$, then:

$$\beta_i^{(s)} \geq t^{(s)} = t^{(s-1)} + 2\lambda \geq t^{(s')} + \lambda + \lambda \geq \beta_j^{(s')} + \lambda,$$

since each honest player $j \in \mathtt{HP}^{(s')}$ sends its message of step s' at time $\beta_j^{(s')}$, then it will reach all honest players by time $\beta_j^{(s')} + \lambda \leq \beta_i^{(s)}$.

Finally we prove Property 3. Let us assume for sake of contradiction that the two players i, i' and the two values b, b' of Property 3 do exist. Note that each malicious player $j \in \mathtt{MP}^{(s)}$ may have signed both a list $\mathbf{v}_j^{(s)}$ with $v_{j,c}^{(s)} = b$ and another list $\mathbf{v}'^{(s)}_j$ with $v'^{(s)}_{j,c} = b'$, but all honest players have signed exactly one list, hence their c-th component is unequivocal.

Therefore, at least $\mathtt{t_H} - \mathtt{MP}^{(s)}$ of the messages received by i advertising b must come from a set $\mathtt{H}$ of honest players, and $\mathtt{t_H} - \mathtt{MP}^{(s)}$ must come from another set $\mathtt{H}'$ of honest players with $\mathtt{H}$ and $\mathtt{H}'$ disjoint sets. Note that the messages advertising different values in the c-th component must be distinct messages, this means that we are considering at least $2\mathtt{t_H}$ distinct messages.

Let M be this set of at least $\mathtt{t_H}$ messages collected by i and M' the analogous set of messages collected by i', then:

$$2\mathtt{t_H} \leq |M| + |M'| \leq |\mathtt{H}| + |\mathtt{MP}^{(s)}| + |\mathtt{H}'| + |\mathtt{MP}^{(s)}| \leq |\mathtt{HP}^{(s)}| + 2|\mathtt{MP}^{(s)}| < 2\mathtt{t_H},$$

where the last inequality holds with overwhelming probability thanks to the assumptions of Definition 2.4. This is a contradiction, therefore such players i and i' do not exist. □

Now we prove a lemma that justifies the construction of a certificate as described in Sect. 3.2. In particular, the messages which constitute a valid certificate do not contain the list that the network is certifying, but just its digest. Therefore a node must be able to determine which is the list associated to that digest. In Lemma 4.2 we prove that a node can find the candidate values for each components from the messages it has received in step 2.

Lemma 4.2 *If a user i builds a certificate C_i for the list $\mathbf{\Theta}$, then, for each $1 \leq c \leq m$, i has received at least one step 2 message from $j \in \mathtt{HP}^{(2)}$ with $v_{j,c}^{(2)} = \Theta_c$.*

Proof Note that from the assumptions of Definition 2.4 on the number of malicious players we have that $|\mathtt{MP}^{(s)}| < \frac{2\mathtt{t_H} - |\mathtt{HP}^{(s)}|}{2} < \frac{2\mathtt{t_H} - \mathtt{t_H}}{2} = \frac{\mathtt{t_H}}{2}$, so at least one of the signatures in the certificate C_i must come from an honest player $k \in \mathtt{HP}^{(s)}$. Then k must have received, during step 3, at least $\frac{\mathtt{t_H}}{2}$ messages for Θ_c in c-th component. Again, since $|\mathtt{MP}^{(s)}| < \frac{\mathtt{t_H}}{2}$, at least one of them must come from an honest player $j \in \mathtt{HP}^{(2)}$ and, according to Item 2 of Lemma 4.1, his message must have reached also i within time $\beta_i^{(3)}$.

To conclude, note that $\beta_i^{(3)}$ is the ending time of step 3 for player i and it is before any possible certificate production time. □

4.2 Component-wise finalization

In this section we prove some properties about the finalization of a single component, distinguishing between the associated ambiguous and unambiguous events to be recorded.

We recall that the finalization checks are performed after every step $s \geq 4$ and refer to messages exchanged during step $s' \geq 3$, and $s' \equiv 0 \mod 3$ (i.e. STEP 3 and all subsequent Coin-Genuinely-Flipped steps) for what concerns FINALIZATION CHECK 0 and step $s'' \geq 4$, and $s'' \equiv 1 \mod 3$ (i.e. all Coin-Fixed-To-0 steps)for what concerns FINALIZATION CHECK 1.

4.2.1 Unambiguous components

In the following proposition we will show how the network behaves if the event associated to a specific component is unambiguous. In particular, we will explain, following the protocol steps, why every honest player will finalize that component within the end of STEP 5.

Proposition 4.1 (c-Agreement on Unambiguous Components) *Let c be an unambiguous component, then the following happens with overwhelming probability:*

- *all honest users have their c-th component finalized by step* 5 *(and in particular there is c-agreement on the lists* $\mathbf{\Theta}_i^{(s)}$ *for all* $s \geq 5$*);*
- $T_c \leq t^{(5)} + \lambda$.

Proof Note that every honest player $i \in \mathtt{HP}^{(s)}$ starts its step s at time $\alpha_i \in [0, \lambda]$. Now we analyse the protocol step by step.

STEP 1 Since component c is unambiguous, then for a certain value $x \in V_c$ each honest player $i \in \mathtt{HP}^{(1)}$ will build a list $\mathbf{v}_i^{(1)}$ with $v_{i,c}^{(1)} = x$. Then i will propagate its message $\mathtt{m}_i^{(1)}$ at time $\beta_i^{(1)} = \alpha_i + \Omega$.

STEP 2 When an honest player $i \in \mathtt{HP}^{(2)}$ stops waiting at time $\beta_i^{(2)} = \alpha_i + t^{(2)}$, i has received all step 1 messages sent by the other honest players.
By our assumptions we have, with overwhelming probability, $|\mathtt{HP}^{(1)}| > \mathtt{t_H}$, hence more than $\mathtt{t_H}$ step 1 messages $\mathtt{m}_j^{(1)}$ that i has received contain a list $\mathbf{v}_j^{(1)}$ with $v_{j,c}^{(1)} = x$. Then, whether $x = \perp$ or $x \neq \perp$, player i builds a list $\mathbf{v}_i^{(2)}$ with $v_{i,c}^{(2)} = x$ and broadcasts the message $\mathtt{m}_i^{(2)}$ containing the digital signature of this list.

STEP 3 When an honest player $i \in \mathtt{HP}^{(3)}$ stops waiting at time $\beta_i^{(3)} = \alpha_i + t^{(3)}$, i has received all step 2 messages from all the honest players.
Since the lists in their messages $\mathtt{m}_j^{(2)}$ have $v_{j,c}^{(2)} = x$, and with overwhelming probability $|\mathtt{HP}^{(2)}| > \mathtt{t_H}$, then player i will set $\left(O_{i,c}, g_{i,c}\right) = (x, 2)$ if $x \neq \perp$, $\left(O_{i,c}, g_{i,c}\right) = (x, 0)$ if $x = \perp$. So, i will build the list $\mathbf{v}_i^{(3)}$ with $v_{i,c}^{(3)} = 0$ if $x \neq \perp$ or $v_{i,c}^{(3)} = 1$ if $x = \perp$, and broadcast its message $\mathtt{m}_i^{(3)}$.

STEP 4 When an honest player $i \in \mathtt{HP}^{(4)}$ stops waiting at time $\beta_i^{(4)} = \alpha_i + t^{(4)}$, i has received all step 3 messages from all the honest players. We now consider separately two cases:

- $x \neq \perp$, in this case player i enters the FINALIZATION CHECK 0 ($4 - 1 \equiv 0 \mod 3$) and since the number of honest players is $|\mathtt{HP}^{(3)}| \geq \mathtt{t}_{\mathtt{H}}$ with overwhelming probability, player i sets $v_{i,c}^{(4)} = 0$ and $f_{i,c} = 1$. This means that all honest players have finalized the c-th component of the list and they will get $\Theta_{i,c}^{(s)} = x$ for all $s \geq 4$.
- $x = \perp$, in this case player i will neither enter the ENDING CONDITION nor any FINALIZATION CHECK. It will build a list $\mathbf{v}_i^{(4)}$ such that $v_{i,c}^{(4)} = 1$ since with overwhelming probability $|\mathtt{HP}^{(3)}| > \mathtt{t}_{\mathtt{H}}$. Player i will broadcast its message $\mathtt{m}_i^{(4)}$ containing the digital signature of $\mathbf{v}_i^{(4)}$.

Thus c-agreement has been reached if $x \neq \perp$, otherwise we will see that it will be reached in the next step.

STEP 5 When an honest player $i \in \mathtt{HP}^{(5)}$ stops waiting at time $\beta_i^{(5)} = \alpha_i + t^{(5)}$, i has received all step 4 messages from all the honest players. Again, we consider two cases:

- $x \neq \perp$, in this case c-agreement on lists $\boldsymbol{\Theta}_i^{(s)}$ has already been reached, and the c-th component has already been finalized by the honest players.
- $x = \perp$, in this case, i has received with overwhelming probability at least $\mathtt{t}_{\mathtt{H}}$ messages from all the other honest players $j \in \mathtt{HP}^{(4)}$ containing the digital signature of a list $\mathbf{v}_j^{(4)}$ with $v_{j,c}^{(4)} = 1$. Then i enters the FINALIZATION CHECK 1 and sets $v_{i,c}^{(5)} = 1$ and $f_{i,c} = 1$. This means that all honest players have finalized the c-th component of the list and they will get $\Theta_{i,c}^{(s)} = \perp$ for all $s \geq 5$.

Since the malicious players are less than $\mathtt{t}_{\mathtt{H}}$, they will not be able to produce the number of messages required to mislead the honest players. We have seen that by the end of step 5 all honest users have finalized the c-th component (and they are in agreement with each other), so we have that $T_c \leq t^{(5)} + \lambda$. □

4.2.2 Ambiguous components

Let us now tackle the more difficult case of ambiguous components. In Proposition 4.2 we deal with the simpler sub-case, when no honest player sets $g_{i,c} = 2$ during step 3, which means that there is a wide disagreement among the network about that specific component. As we will see, this case will resolve with the achievement of agreement on the symbol $\perp$.

Then, we complete the analysis by considering the case when some honest node sets $g_{i,c} = 2$. In Lemma 4.3 we prove that in this case each honest user j has saved the same value $O_{j,c} = x \in V_c$ at the beginning of step 3. They do not know yet that they already are in agreement, so each of them tries to figure out whether to preserve that component of the final list or to discard it by setting it to $\perp$. This decision will be made by exchanging the bit lists from step 3 onward. In fact, as it is stated in Lemma 4.4, once c-agreement is reached on the bit list either on 0 or 1, it is also reached on the list $\mathbf{2}_j \in V$ respectively on $x \in V_c$ or $\perp$.

Then, it becomes essential to prove that c-agreement is achievable on each component with probability 1, and also to upper-bound the time required to achieve it. To do that, we prove in Lemma 4.5 that the network will reach c-agreement on the bit list with probability greater than $\frac{1}{3}$ after every Coin-Genuinely-Flipped. Therefore, in Lemma 4.6 we prove that c-agreement is eventually reached with probability 1 and that all the honest nodes will finalize the component c with the same value finalized by the first node in the network who can do it (even if the first user who can finalize the component is malicious).

These Lemmas are then used to prove Proposition 4.3, where we present an upper bound to the number of steps and time required to finalize a single component. Finally, the results of the previous lemmas and propositions are used to prove Theorem 4.1.

Proposition 4.2 (c-Agreement on Very Ambiguous Components) *Let c be an ambiguous component and assume that all honest step 3 players set $g_{i,c} < 2$, then with overwhelming probability we have that:*

- *all honest users have their c-th component finalized (in particular will be in c-agreement on the list $\mathbf{\Theta}_i^{(s)}$) in step 5, setting $\Theta_{i,c}^{(s)} = \perp$;*
- $T_c \leq t^{(5)} + \lambda$.

Proof By definition of the protocol Cob, for each $i \in \mathtt{HP}^{(3)}$, i sets $v_{i,c}^{(3)} = 1$, since $g_{i,c} < 2$. This means that the honest step 3 players start in agreement on the component c of the list $\mathbf{v}_i^{(3)}$. They may not be in agreement on the list component $O_{i,c}$, but this does not matter. In fact, during step 4 no honest player is able to finalize component c by collecting more than $\mathtt{t_H}$ messages with $v_{k,c}^{(3)} = 0$, since $|\mathtt{MP}^{(3)}| < \mathtt{t_H}$. In the same way even if the honest players have received more than $\mathtt{t_H}$ valid messages with $v_{k,c}^{(3)} = 1$ they will not finalize the component c because $4 - 1 \not\equiv 1 \mod 3$. Anyway, each honest step 4 player i has received all honest messages, hence more than $\mathtt{t_H}$ advertising $v_{k,c}^{(3)} = 1$. This means that i will create a message $\mathtt{m}_i^{(4)}$ with $v_{i,c}^{(4)} = 1$.

During step 5, which is a Coin-Fixed-To-1 step, each honest user i will receive all messages by other honest players before $\beta_i^{(5)}$, hence with overwhelming probability i will receive $\mathtt{t_H}$ messages for $v_{k,c}^{(4)} = 1$. Thus i will enter the FINALIZATION CHECK 1 and will finalize the c-th component by setting $v_{i,c}^{(5)} = 1$ and $f_{i,c} = 1$.

This means that the honest users reach c-agreement on $\Theta_{i,c}^{(s)} = \perp$ by the end of step 5 (which happens in the interval $[t^{(5)}, t^{(5)} + \lambda]$), and it will hold for all $s \geq 5$. Note that in this case, a strong disagreement among the players results in the component to be set to $\perp$ as it happens in Proposition 4.1 when every honest players starts with $\perp$. □

Proposition 4.3 (c-Agreement on Ambiguous Components) *Let c be an ambiguous component and assume that there exists a player $i \in \mathtt{HP}^{(3)}$ which sets $g_{i,c} = 2$, then:*

- *all honest users have their c-th component finalized (in particular will be in c-agreement on the list $\mathbf{\Theta}_i^{(s)}$) within step $3L + 5$;*
- $T_c \leq t^{(3L+5)} + \lambda$.

Where L is a random variable representing the number of Bernoulli trials needed to see the output 1*, when each trial outputs* 1 *with probability* $\frac{h}{2}$.

In order to prove Proposition 4.3 we need the results stated in the following four lemmas.

Lemma 4.3 *Under the assumption of Proposition 4.3, we show that the following properties hold:*

1. $g_{j,c} \geq 1$ *for all* $j \in \mathtt{HP}^{(3)}$*;*
2. *there is a value* $x \in V_c$ *such that* $O_{j,c} = x$ *for all* $j \in \mathtt{HP}^{(3)}$.

Proof Since player $i \in \mathtt{HP}^{(3)}$ is honest and sets $g^{(3)}_{i,c} = 2$, then:

1. i sets $O_{i,c} = x$ since it has received more than $\mathtt{t_H}$ messages $\mathtt{m}^{(2)}_k$ advertising $v^{(2)}_{k,c} = x$. By Property 3 of Lemma 4.1 we know that no honest player $j \in \mathtt{HP}^{(3)}$ has received $\mathtt{t_H}$ messages $\mathtt{m}^{(2)}_k$ for $v^{(2)}_{k,c} = x' \neq x$, hence if $g^{(3)}_{j,c} = 2$ it must be $O_{j,c} = x$.
   As showed in the proof of Lemma 4.2, we have that with owerwhelming probability $|\mathtt{MP}^{(s)}| < \frac{\mathtt{t_H}}{2}$, so we can state that more than $\frac{\mathtt{t_H}}{2}$ honest players must have signed for x. Therefore, if $g^{(3)}_{j,c} < 2$, then $g^{(3)}_{j,c} = 1$, and Property 1 holds.
2. We now show that, even if j sets $g_{j,c} = 1$, it will set $O_{j,c} = x$. In fact, there can not exist a value $x' \neq \perp$ and $x' \neq x$ such that j has received also more than $\frac{\mathtt{t_H}}{2}$ step 2 messages $\mathtt{m}^{(2)}_k$ with $v^{(2)}_{k,c} = x'$. For the sake of contradiction, we suppose that these messages exist; many of them may come from malicious players in $\mathtt{MP}^{(2)}$, but at least one of them must come from an honest player $p \in \mathtt{HP}^{(2)}$. This means that p has received $\mathtt{t_H}$ step 1 messages $\mathtt{m}^{(1)}_k$ with $v^{(1)}_{k,c} = x'$. Since we have seen that some other honest step 2 players have signed a step 2 message advertising $v^{(2)}_{k,c} = x$, this implies that they have seen $\mathtt{t_H}$ step 1 messages with $v^{(1)}_{k,c} = x$, which, by Lemma 4.1, is a contradiction. This means that Property 2 holds. □

Lemma 4.4 *Under the assumptions of Proposition 4.3 we have that c-agreement on the list* $\mathbf{\Theta}$ *is reached when c-agreement is reached on the bit list* $\mathbf{v}^{()}$.

Proof Since the honest player $i \in \mathtt{HP}^{(3)}$ sets $g_{i,c} = 2$, it will set $v_{i,c} = 0$, therefore it is possible that c-agreement is reached on 0. By the analysis of Property 2 of Lemma 4.3 the honest users may not have an agreement on their $v^{(3)}_{i,c}$ at the end of step 3 but they will have an agreement on $O^{(3)}_{i,c} = x$. This means that, by the definition of Cob, when c-agreement is reached on the bit list (either on 0 or 1), then it will also be reached on x in $\mathbf{\Theta}$ (respectively on x or $\perp$). □

Remark 4.2 This property is true also in the general case, but we have explicitly proved it only under the assumptions of Proposition 4.3.

Lemma 4.5 *Being c a component on which agreement among the honest user does not hold, at every Coin-Genuinely-Flipped step c-agreement is reached with probability at least* $\frac{h}{2}$.

Proof Assuming that c-agreement is not reached at the beginning of a Coin-Genuinely-Flipped step s where $s \geq 6$, $s-1 \equiv 2 \mod 3$, let an honest player $i \in \mathtt{HP}^{(s)}$ be in the condition that it must flip the coin during such step.

This means that the player i has not received more than $\mathtt{t}_\mathtt{H}$ messages for a bit $b \in \{0, 1\}$ in component c, so i selects its own coin flipper ℓ', and i will set $v_{i,c}^{(s)} = b_i = H\left(H\left(\sigma_{\ell'}^{(s-1)}\right)\right)_c$, where $H(K)_c$ is the c-th bit of $H(K)$.

Note that, by Lemma 4.1, if an honest player has seen more than $\mathtt{t}_\mathtt{H}$ messages for the same bit b in component c, then no honest player has seen more than $\mathtt{t}_\mathtt{H}$ messages for $1-b$ in the same component. This means that i will be in c-agreement with the honest players who did not flip the coin only if $b_i = b$, and this happens with probability $\frac{1}{2}$ with the Random Oracle assumption for the hash function H.

Therefore, all honest players in $\mathtt{HP}^{(s)}$ will be in agreement with probability $\frac{1}{2}$. Actually, this is true if the coin flipper is an honest player, in fact in this case b_i can be assumed to be randomly chosen and globally shared among the honest players flipping the coin. If the player with minimal hashed credential is a malicious player, then we cannot say much about the probability distribution of the output of the bit extraction, since some players may not have seen its message.

However, with our assumptions on the common reference string r, we can state that, with probability h, the coin flipper will be honest, and in this case all honest players will be in agreement with probability $\frac{1}{2}$.

Combining these two independent probabilities we get that with probability at least $\frac{h}{2}$ the honest players reach c-agreement every time they enter a Coin-Genuinely-Flipped step. Note that they will finalize this component within the following 2 steps: in $s+1$ if $b_i = 0$, in $s+2$ if $b_i = 1$. □

Lemma 4.6 *Under the assumptions of Proposition 4.3, we have that:*

1. *being E the event "there exists a step $\hat{s} \geq 4$ such that, for the first time, some user $\hat{\imath} \in \mathtt{PK}$ (either malicious or honest) should finalize its c-th component of list $\mathbf{v}_{\hat{\imath}}^{(\hat{s})}$", E happens with probability 1;*
2. *$T_c \leq t^{(\hat{s}+3)} + \lambda$ and c-agreement is reached in step $\hat{s}$ on the same value finalized by $\hat{\imath}$ (however, the c-th component might be finalized 3 steps later).*

Proof 1. As proven in Lemma 4.5, if c-agreement is not reached by the honest users, at every Coin-Genuinely-Flipped step they will reach it with probability at least $\frac{h}{2} > \frac{1}{3}$.
Therefore, once every 3 steps the c-th component will be finalized with probability greater than $\frac{1}{3}$, therefore the probability that the event E happens converges to 1.
2. Step $\hat{s}$ is the first step in which a user $\hat{\imath}$ can finalize the c-th component. By the construction of the protocol, this happens in two possible ways:

E_a: $\hat{\imath}$ is able to collect or generate (and then propagate) at least $\mathtt{t}_\mathtt{H}$ valid messages $\mathtt{m}_k^{(s'-1)}$ with $v_{k,c}^{(s'-1)} = 0$, $4 \leq s' \leq \hat{s}$, and $s'-1 \equiv 0 \mod 3$;
E_b: $\hat{\imath}$ is able to collect or generate (and then propagate) at least $\mathtt{t}_\mathtt{H}$ valid messages $\mathtt{m}_k^{(s'-1)}$ with $v_{k,c}^{(s'-1)} = 1$, $4 \leq s' \leq \hat{s}$, and $s'-1 \equiv 1 \mod 3$;

Because the messages produced during step $s' - 1$ by honest players are received by every user before they are done waiting in step s', and because the adversary receives everything no later than the honest users, without loss of generality we can assume that $s' = \hat{s}$, and that the user $\hat{\imath}$ is malicious.
For any step $s \geq 4$, every honest player $i \in \mathtt{HP}^{(s)}$ who has waited time $t^{(s)}$ has received all honest step $s - 1$ messages (thanks to Lemma 4.1), and all honest players in $\mathtt{HP}^{(s)}$ have set $O_{i,c} = x$ (according to Lemma 4.3).
We now consider step $\hat{s}$ and examine 4 exhaustive ways in which event E may happen.

Case 2.1.a: *event E_a happens and there is an honest user $i' \in \mathtt{PK}$ who should also finalize the c-th component.*
In this case, we have $\hat{s} - 1 \equiv 0 \mod 3$, hence Step $\hat{s}$ is a Coin-Fixed-To-0 step. By assumption, i' has received at least $\mathtt{t_H}$ valid step $\hat{s} - 1$ messages $\mathtt{m}_k^{(\hat{s}-1)}$ with $v_{k,c}^{(\hat{s}-1)} = 0$. Thus i' finalizes its component c setting $v_{i',c}^{(\hat{s})} = 0$ and $f_{i',c} = 1$.
Now we show that any other honest user i has either finalized its c-th component, setting $v_{i,c}^{(\hat{s})} = 0$ and $f_{i,c} = 1$, or has set $v_{i,c}^{(\hat{s})} = 0$ without finalizing such component.
Because step $\hat{s}$ is the first time any player i should finalize component c of the list $\mathbf{v}_i^{(s)}$, there does not exist a Coin-Fixed-To-1 step $s' < \hat{s}$ (hence $s' - 1 \equiv 1 \mod 3$) such that $\mathtt{t_H}$ players have signed $v_{i,c}^{(s'-1)} = 1$. Accordingly, no online user in $\mathtt{PK}$ finalizes the list component c in step s' setting $v_{i,c}^{(s')} = 1$. Moreover, if an honest user has waited for a time $t^{(\hat{s})}$, then it must have received all step $\hat{s} - 1$ messages from honest players, and (considering the messages received by i') at least $\mathtt{t_H} - |\mathtt{MP}^{(\hat{s}-1)}| \geq 1$ must have $v_{k,c}^{(\hat{s}-1)} = 0$. According to Property 4 of Lemma 4.1, an honest player i cannot collect $\mathtt{t_H}$ messages with $v_{k,c}^{(\hat{s}-1)} = 1$, therefore it sets $v_{i,c}^{(\hat{s})} = 0$.
For step $\hat{s} + 1$, since user i' has helped propagating the messages that have let it finalize the c-th component on or before time $\alpha_{i'} + t^{(\hat{s})}$, then on or before time $\beta_i^{(\hat{s}+1)}$ each honest user i has received at least $\mathtt{t_H}$ valid $\hat{s} - 1$ messages for the bit 0. In fact, even if some of the $\mathtt{t_H}$ messages received by i' were not broadcast in time by a malicious user, within time $\beta_i^{(\hat{s}+1)}$ they have reached i, for all $i \in \mathtt{PK}$. This is true because user i' received the messages within time $\beta_{i'}^{(\hat{s})}$ and helped propagating them, hence within time $\beta_{i'}^{(\hat{s})} + \lambda$ they have reached all honest players, and for all honest players $i \in \mathtt{HP}^{(\hat{s}+1)}$ we have:

$$\beta_i^{(\hat{s}+1)} \geq t^{(\hat{s})} + 2\lambda \geq \alpha_{i'} + t^{(\hat{s})} + \lambda = \beta_{i'}^{(\hat{s})} + \lambda.$$

Furthermore, honest players will not end step $\hat{s}+1$ before receiving those step $\hat{s}-1$ messages, because there do not exist other $\mathtt{t_H}$ valid step $s'-1$ messages for 1 in the component c with $s'-1 \equiv 1 \mod 3$ and $5 \leq s' < \hat{s}+1$, by the definition of Step $\hat{s}$ in assumption E_a (step $\hat{s}$ is the first step in which a user should finalize component c). In particular, step $\hat{s}+1$ itself is a Coin-Fixed-To-1 step, but no honest player has propagated during step $\hat{s}$ a message for 1 (as we have shown they have reached c-agreement on 0), and $|\mathtt{MP}^{(\hat{s})}| < \mathtt{t_H}$. Thus all honest users finalize their c-th component, setting $v_{i,c}^{(\hat{s}+1)} = 0$ and $f_{i,c} = 1$.

So, we have proven that, for all $s \geq \hat{s}+1$, the honest users set $v_{i,c}^{(s)} = 0$ and $f_{i,c} = 1$ (i.e. they finalize the component c within the end of step $\hat{s}+1$). We have already seen that $\exists x \in V_c$ such that $O_{i,c} = x$ for every honest user i, so they will set $\Theta_{i,c}^{(s)} = x$ and therefore c-agreement reached on $\boldsymbol{\Theta}_i^{(s)}$ for all $s \geq \hat{s}+1$.

Case 2.1.b: *event E_b happens and there is an honest user $i' \in \mathtt{PK}$ who should also finalize c-th component.*

In this case we have $\hat{s}-1 \equiv 1 \mod 3$, then step $\hat{s}$ is a Coin-Fixed-To-1 step. The analysis is similar to *Case 2.1.a* and we will omit many details. As in the previous case, i' must have received $\mathtt{t_H}$ valid step $\hat{s}-1$ messages with $v_{k,c}^{(\hat{s}-1)} = 1$. Again, by the definition of step $\hat{s}$, there does not exist a step s', with $4 \leq s' \leq \hat{s}$ and $s'-1 \equiv 0 \mod 3$, where at least $\mathtt{t_H}$ players have signed a message with $v_{k,c}^{(s'-1)} = 0$. Thus, i' finalizes the c-th component and sets $v_{i',c}^{(\hat{s})} = 1$ and $f_{i',c} = 1$. Moreover, any other honest user $i \in \mathtt{PK}$ has either finalized its c-th component if it has received $\mathtt{t_H}$ messages with $v_{k,c}^{(\hat{s}-1)} = 1$ or has set $v_{i,c}^{(\hat{s})} = 1$ and broadcast its $\mathtt{m}_i^{(\hat{s})}$ message. Since i' has helped propagating the step $\hat{s}-1$ messages it has received by time $\alpha_{i'} + t^{(\hat{s})}$, all honest users finalize the c-th component during step $\hat{s}+1$.

Again, they will reach c-agreement over $\perp$ on $\boldsymbol{\Theta}_i^{(s)}$ for all $s \geq \hat{s}$, they will set $v_{i,c}^{(\hat{s}+1)} = 1$ and $f_{i,c} = 1$, finalizing the component c within the end of step $\hat{s}+1$.

Case 2.2.a: *event E_a happens and there does not exist an honest user $i' \in \mathtt{PK}$ who should also finalize c-th component.*

In this case, note that $\hat{\imath}$ could have received or generated $\mathtt{t_H}$ step $\hat{s}-1$ messages with $v_{k,c}^{(s-1)} = 0$. However, the malicious users may not help propagating those messages, so we cannot conclude that the honest users will receive them after time λ. In fact, $|\mathtt{MP}^{(\hat{s}-1)}|$ of those messages may be from malicious players, who did not propagate their messages at all and only sent them to the other malicious players cooperating with them.

Therefore, the honest users will wait for time $t^{(\hat{s})}$ without finalizing component c. However, by Property 4 of Lemma 4.1, they will not see more than $\mathtt{t_H}$ of the messages received with $v_{k,c}^{(\hat{s}-1)} = 1$, again because

with overwhelming probability $|\mathtt{HP}^{(\hat{s}-1)}| + 2|\mathtt{MP}^{(\hat{s}-1)}| < 2\mathtt{t}_\mathtt{H}$. Since step $\hat{s}$ is a Coin-Fixed-To-0 step, every honest player $i \in \mathtt{HP}^{(\hat{s})}$ thus sets $v_{i,c}^{(\hat{s})} = 0$ and propagates its message at time $\alpha_i + t^{(\hat{s})}$.
During step $\hat{s} + 1$, which is a Coin-Fixed-To-1 step, two things may happen:

1 an honest user receives the $\mathtt{t}_\mathtt{H}$ messages received by $\hat{\imath}$ (who decided to propagate them and let it finalize component c): in this case the situation is similar to *Case 2.1.a*, and every honest user i will finalize its c-th component within time $\alpha_i + t^{(\hat{s}+1)} + \lambda$;

2 the honest users will receive at least $|\mathtt{HP}^{(\hat{s})}|$ ($> \mathtt{t}_\mathtt{H}$ with overwhelming probability) messages with $v_{k,c}^{(\hat{s})} = 0$ from the honest players. Then they propagate their messages with $v_{i,c}^{(\hat{s}+1)} = 0$ but do not finalize since step $\hat{s} + 1$ is not a Coin-Fixed-To-0 step.

In this case, in step $\hat{s} + 2$ which is a Coin-Genuinely-Flipped step, two things may happen:

2.1. if $\hat{\imath}$ broadcasts the step $\hat{s} - 1$ messages that let it finalize its c-th component, then the honest users will finalize their c-th component as well setting $v_{i,c}^{(\hat{s}+2)} = 0$ and $f_{i,c} = 1$, hence reaching c-agreement over $\Theta_{i,c}^{(s)} = x$ for all $s \geq \hat{s} + 2$ within time $\alpha_i + t^{(\hat{s}+2)}$;

2.2. otherwise all honest users have received all step $\hat{s} + 1$ messages from the honest players with $v_{i,c}^{(\hat{s}+1)} = 0$. Again, there are more honest players than $\mathtt{t}_\mathtt{H}$, so the honest users set $v_{i,c}^{(\hat{s}+2)} = 0$ without flipping the coin. Again, they do not finalize component c since $\hat{s} + 2$ is not a Coin-Fixed-To-0 step so they just broadcast their step $\hat{s} + 2$ messages.
Finally, step $\hat{s} + 3$ is a Coin-Fixed-To-0 step, so everyone will receive at least $\mathtt{t}_\mathtt{H}$ messages with $v_i^{(\hat{s}+2)} = 0$ where $i \in \mathtt{HP}^{(\hat{s}+2)}$. Then all honest users k at time $\alpha_k + t^{(\hat{s}+3)}$ can finalize component c setting $v_{k,c}^{(\hat{s}+3)} = 0$ and $f_{k,c} = 1$ and hence reach c-agreement over $\mathbf{\Theta}_k^{(s)} = x$ for all $s \geq \hat{s}+3$.

Depending on how $\hat{\imath}$ and in general the malicious users behave, some users may finalize the component c within the end of step s (with $s \in \{\hat{s}, \hat{s} + 1, \hat{s} + 2\}$) using step $\hat{s} - 1$ messages, or within the end of step $\hat{s} + 3$ with step $\hat{s} + 2$ messages. It does not matter since c-agreement is reached anyway over $\mathbf{\Theta}_i^{(s)}$ for all $s \geq \hat{s}$, the component c is finalized setting $f_{i,c} = 1$ within step $\hat{s} + 3$, and $v_{i,c}^{(s)} = 0$ for every step s such that $s \geq \hat{s} + 3$.

Case 2.2.b: *event E.b happens and there does not exist an honest user* $i' \in \mathtt{PK}$ *who should also finalize c-th component.*
The analysis in this case is similar to *Case 2.1.b* and *Case 2.2.a*, thus many details have been omitted.
We know that $\hat{\imath}$ has collected or generated at least $\mathtt{t}_\mathtt{H}$ step $\hat{s} - 1$ messages with $v_{k,c}^{(\hat{s}-1)} = 1$ and $\hat{s} - 1 \equiv 1 \mod 3$ (hence $\hat{s}$ is a Coin-Fixed-To-1

step) and that no honest player could have seen more than $\mathtt{t_H}$ messages for 0. Thus each honest player $i \in \mathtt{HP}^{(\hat{s})}$ sets $v_{i,c}^{(\hat{s})} = 1$ and propagates its message $\mathtt{m}_i^{(\hat{s})}$ at time $\alpha_i + t^{(\hat{s})}$. Similar to *Case 2.2.a*, within 3 steps user i will finalize their c-th component.
Then c-agreement is reached over $\mathbf{\Theta}_i^{(s)}$ for all $s \geq \hat{s}$, and the component c is finalized setting $f_{i,c} = 1$ within step $\hat{s} + 3$ and $v_{i,c}^{(s)} = 0$ for all steps $s, s \geq \hat{s} + 3$.

Combining the four sub-cases, we obtain:

- $T_c \leq t^{(\hat{s})} + \lambda$ in *Case 2.1.a* and *Case 2.1.b*;
- $T_c \leq t^{(\hat{s}+3)} + \lambda$ in *Case 2.2.a* and *Case 2.2.b*;

but we also have that they all are in agreement at the end of step $\hat{s}$, and that c-agreement is reached over $\mathbf{\Theta}_i^{(s)}$ for all $s \geq \hat{s}$.

□

Now we can prove Proposition 4.3.

Proof of Proposition 4.3 Given the results of Lemma 4.6, it remains to upper-bound $\hat{s}$ and thus T_c. We do that by considering how many times the Coin-Genuinely-Flipped steps are executed by at least one honest player.

If no honest player flips the coin in a Coin-Genuinely-Flipped step s, it means that they all have received more than $\mathtt{t_H}$ messages with $v_{k,c}^{(s-1)} = b \in \{0, 1\}$ and c-agreement has been reached, letting them finalize the component in at most 2 more protocol steps. Moreover if they reach c-agreement over 0, this means that they agree on the same value $O_{k,c}$ to insert as c-th component of the list $\mathbf{\Theta}_k^{(s)}$. Once c-agreement is reached in step s, the honest players will finalize the c-th component either in step $s + 1$ or step $s + 2$ depending on whether $b = 0$ or $b = 1$.

By Lemma 4.5, at every Coin-Genuinely-Flipped step c-agreement is reached with probability at least $\frac{h}{2}$, so we can compare this step to a Bernoulli trial that outputs 1 if c-agreement is reached. This means that, before step $\hat{s}$ (the first step in which a user can finalize the c-th component), the distribution of the number of times the Coin-Genuinely-Flipped steps are executed to finalize a component c can be upper-bounded by to the random variable L, which we recall represents the number of Bernoulli trials needed to see a 1 when each trial gives 1 with probability $\frac{h}{2} > \frac{1}{3}$. Letting s' be the last Coin-Genuinely-Flipped step before the finalization of the c-th component, then we have, by the protocol construction, $s' = 3 + 3L$.

Assuming that the adversary knows the outcome of L in advance, when should the adversary make step $\hat{s}$ happen to maximize the delay of the finalization time T_c of the c-th component by an honest user?

If $\hat{s} > s'$ (hence $\hat{s} = s' + 1$ or $\hat{s} = s' + 2$) then this means that we are in *Case 2.1.a* or *Case 2.1.b* of Lemma 4.6 since at the end of step s' the honest players are already in agreement, so when a malicious player could finalize, also the honest users can, hence

$$T_c \leq t^{(\hat{s})} + \lambda \leq t^{(s'+2)} + \lambda.$$

If $\hat{s} < s' - 3$, that is $\hat{s}$ is before the second to last Coin-Genuinely-Flipped step, then by the analysis of *Case 2.2.a* or *Case 2.2.b* we get

$$T_c \leq t^{(\hat{s}+3)} + \lambda \leq t^{(s')} + \lambda,$$

that is, the Adversary is making the agreement on component c happen faster.

If $\hat{s} = s' - 1$ or $\hat{s} = s' - 2$, then $\hat{s}$ is the Coin-Fixed-To-0 or Coin-Fixed-To-1 step before s'. By the analysis of the 4 sub-cases we know that the honest players never flip the coin and finalize the c component within the next two steps. Therefore, the following holds:

$$T_c \leq t^{(\hat{s}+3)} + \lambda \leq t^{(s'+2)} + \lambda.$$

To summarize, no matter what $\hat{s}$ is, we have:

$$T_c \leq t^{(s'+2)} + \lambda = t^{(3L+5)} + \lambda,$$

which upper-bounds the time needed to reach agreement on the c-th list component. □

Now we will prove Theorem 4.1, using the results of the previous lemmas and propositions. We will prove that all the honest users will agree on the same $\mathbf{\Theta}$, that no malicious user can build a valid certificate for a different $\hat{\mathbf{\Theta}}$, and that the honest users will be able to produce a certificate for $\mathbf{\Theta}$ within time $t^{\left(5+3\chi_{\ell,\frac{h}{2}}\right)} + \lambda$. We also prove that only one list can be certified, therefore the nodes will reach agreement on a list and it is not possible for the malicious users to produce a valid certificate for another list. This guarantees the consistency property of Cob.

Proof of Theorem 4.1 It is sufficient to note that, since $t^{(s+1)} = t^{(s)} + 2\lambda$ for all $s \geq 3$, we have that:

$$\begin{aligned} t^{\left(5+3\chi_{\ell,\frac{h}{2}}\right)} &= t^{(1)} + \sum_{s=2}^{5+3\chi_{\ell,\frac{h}{2}}} \left(t^{(s)} - t^{(s-1)}\right) & (1) \\ &= t^{(1)} + \left(t^{(2)} - t^{(1)}\right) + \left(t^{(3)} - t^{(2)}\right) + \sum_{s=4}^{5+3\chi_{\ell,\frac{h}{2}}} \left(t^{(s)} - t^{(s-1)}\right) \\ &= \Omega + \Lambda + \lambda + \Lambda + \lambda + \sum_{s=4}^{5+3\chi_{\ell,\frac{h}{2}}} 2\lambda \\ &= \Omega + 2\Lambda + 2\lambda + \left(2 + 3\chi_{\ell,\frac{h}{2}}\right) 2\lambda \\ &= \Omega + 2\Lambda + \left(6 + 6\chi_{\ell,\frac{h}{2}}\right) \lambda. & (2) \end{aligned}$$

Let $m - \ell$ be the number of unambiguous components, we have shown in Proposition 4.1 that these components will reach agreement in at most 5 steps. The same will happen for some of the ℓ ambiguous components according to Proposition 4.2, while the others, as shown in Proposition 4.3, will reach agreement within a number of steps whose distribution is upper-bounded by the random variable $3L + 5$.

Cob runs until a certificate for a list $\mathbf{\Theta} \in V$ is created, this happens *no later* than the moment in which every component is finalized by the honest players, which happens once c-agreement is reached on each of the ℓ ambiguous components.

Since we have shown in Lemma 4.5 that with probability at least $\frac{h}{2}$ the honest players will reach c-agreement on a single component, and once agreement is reached it is maintained for the whole protocol run, then agreement will be reached in at most $\chi_{\ell,\frac{h}{2}}$ Coin-Genuinely-Flipped steps (accordingly to the analysis in Proposition 4.3, malicious users might speed the consensus process up!) where $\chi_{\ell,\frac{h}{2}}$ is the same random variable described in [9].

Every honest user will be able to obtain a certificate for a block at the end of the two steps following the $\chi_{\ell,\frac{h}{2}}$-th Coin-Genuinely-Flipped step according to the definition and analysis of Cob shown above. Therefore it holds that $T \leq t^{\left(5+3\chi_{\ell,\frac{h}{2}}\right)} + \lambda$.

Now we show that if a certificate is created for the first time in step s for a list $\mathbf{\Theta} \in V$, then any certificate created will be for the same list $\mathbf{\Theta}$.

Let step s be the first step in which a user k is able to collect a certificate C_k for $\mathbf{\Theta}$. We recall that, by the assumptions in Definition 2.4, given two distinct players $i, j \in \mathtt{P}^{(s)}$ of the same step, it is negligible the probability that i collects $\mathtt{t_H}$ messages for a list $\mathbf{\Theta}_i^{(s-1)}$ and j collects $\mathtt{t_H}$ messages for a distinct list $\mathbf{\Theta}_j^{(s-1)}$.

As before, we can assume that the user k is malicious and the certificate is made of step $s-2$ and step $s-1$ messages, where step s is a Coin-Fixed-To-1 step.

We distinguish two cases:

- *There is an honest user k' who also can collect a certificate $C_{k'}$ for $\mathbf{\Theta}$ in step s.* In this case k' has propagated the messages which let it certify the list $\mathbf{\Theta}$, hence all honest users will be in possession of a certificate (possibly a different one) for the same list $\mathbf{\Theta}$, so the honest users will agree on the same list $\mathbf{\Theta}$. Also, the honest users will end the protocol execution, so there is no chance that another certificate is produced in the following steps since with overwhelming probability $\mathtt{MP}^{(s)} < \mathtt{t_H}\ \forall s$.
- *There is no honest user k' who also can collect a certificate $C_{k'}$ for $\mathbf{\Theta}$ in step s.* In this case the honest users will keep executing the protocol until they can create a certificate for a block $\hat{\mathbf{\Theta}}$ or until they receive from k the messages that allowed k to create the certificate in step s.

  We will show that if they do not receive the certificate from k, then they will build a certificate for $\hat{\mathbf{\Theta}} = \mathbf{\Theta}$. This guarantees that honest users will agree on the same list, since there are no two valid certificates around the network for two distinct lists.

  If the user k has built a certificate for $\mathbf{\Theta}$ in step s, it means that k has received $\mathtt{t_H}$ messages from step $s-2$ and step $s-1$ for $\mathbf{\Theta}$. In particular, k collected $\mathtt{t_H}$ step $s-2$ messages for 0 in every component c such that $\Theta_c \neq \perp$. We will

call $I_0 = \{c : \Theta_c \neq \bot, 1 \leq c \leq m\}$. By the assumptions in Definition 2.4, no honest user has received at least $\mathtt{t}_{\mathrm{H}}$ messages from step $s-2$ that sponsors 1 in a component $c \in I_0$. This implies that all the honest players have sent in step $s-1$ a message with 0 in each component $c \in I_0$. This brings all the honest users in c-agreement on such components and it will keep holding in the following steps. We also know that the user k has received $\mathtt{t}_{\mathrm{H}}$ messages from step $s-1$ for $\boldsymbol{\Theta}$. This means that k has received at least $\mathtt{t}_{\mathrm{H}}$ messages for 1 in every component c such that $\Theta_c = \bot$. We will call $I_1 = \{c : \Theta_c = \bot, 1 \leq c \leq m\}$. Again, by the assumptions in Definition 2.4, no honest user has received at least $\mathtt{t}_{\mathrm{H}}$ messages from step $s-1$ that sponsor 0 in a component $c \in I_1$. Therefore, all the honest players send, in step s, a message with 1 in each component $c \in I_1$. This brings all the honest users in c-agreement on such components and it will keep holding in the following steps.
Now we note that $I_0 \cup I_1 = \{1, \ldots, m\}$, hence all the honest users are in agreement on all list components. Therefore, in step $s+3$, which is again a Coin-Fixed-To-1 step, they will be able to build a certificate for the list of relevant information $\boldsymbol{\Theta}$ using their messages of step $s+1$ and step $s+2$. □

5 Performance analysis

Given Problem 1.1 and the context of application of Cob, described in Sect. 1.2 (i.e. a consensus protocol for incomplete networks with millions of nodes), we present a comparison which highlights the advantages of using the leaderless and parallel protocol Cob instead of executing ℓ instances of Algorand to achieve the same result.

For the evaluation we will consider the most relevant use case of Cob, namely the example in Sect. 1 of a blockchain implementing sharding where the network must agree on the blocks created by each shard.

In our performance analysis we will compare the total amount of data that is broadcast by the nodes to let the network reach consensus with the two approaches. We quantify the amount of data broadcast by the network to reach consensus on a list of ℓ ambiguous events (which are the most problematic case, hence the worst case scenario) when the ratio of honest users is 80% (i.e. $h = 0.8$) and the expected number of players in each step is $n = 4000$, this choice of parameters which reflects the one used in [7].

First, we will determine the expected number of steps in which the elected players broadcast their messages before producing a certificate both for Cob and for Algorand. Then, we will approximate the weight of the messages broadcast in both protocols, and finally we will compare the total amount of data broadcast as the parameter ℓ, the number of ambiguous components, changes.

5.1 Expected number of steps

Starting with Cob, let us make explicit the probability distribution of the random variable $\chi_{\ell,\frac{h}{2}}$ used to upper-bound the number of steps needed to reach consensus.

In the analysis of protocol MBA [9], the probability distribution of $\chi_{\ell,\frac{h}{2}}$ is shown to be:

$$\mathbb{P}\left(\chi_{\ell,\frac{h}{2}} = w\right) = \left(1-\left(1-\frac{h}{2}\right)^{w}\right)^{\ell} - \left(1-\left(1-\frac{h}{2}\right)^{w-1}\right)^{\ell},$$

from which it is possible to compute the expected number of loops of Coin-Fixed-To-0, Coin-Fixed-To-1 and Coin-Genuinely-Flipped steps the network must execute if the adversary manages to delay as much as possible the consensus achievement. Theorem 4.1 states that the number of steps needed to produce a certificate for the agreed upon list is $5+3\chi_{\ell,\frac{h}{2}}$, therefore the last step in which messages are broadcast is step $4+3\chi_{\ell,\frac{h}{2}}$. This means that the expected number of protocol steps with message broadcasting that are needed to produce a certificate can be computed as:

$$\mathbb{E}\left[\text{Cob steps}_{\ell,\frac{h}{2}}\right] = 4+3\mathbb{E}\left[\chi_{\ell,\frac{h}{2}}\right] = 4+3\sum_{w=1}^{\infty} w\mathbb{P}\left(\chi_{\ell,\frac{h}{2}} = w\right).$$

In Fig. 1 we plot the expected value of the number of steps of Cob $\mathbb{E}\left[\text{Cob steps}_{\ell,\frac{h}{2}}\right]$, according to some possible values of h.

With Algorand, the expected number of loops required to bring the nodes to agreement is $\frac{2}{h}$. Before entering the loop that performs the binary consensus (Binary Byzantine Agreement [23]), the nodes execute 4 extra steps (Graded Consensus [24]). Finally, in the worst case, the nodes will be able to produce a certificate 2 steps after

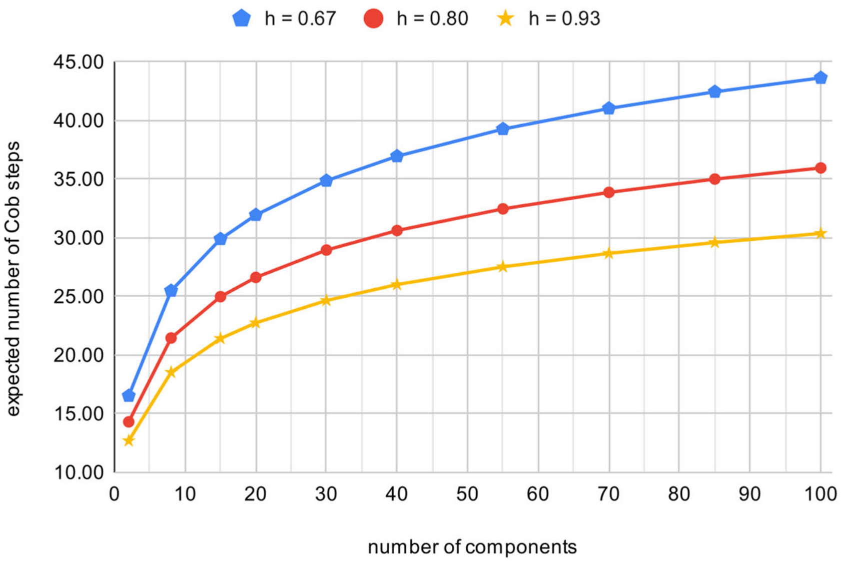

Fig. 1 Expected number of steps with message broadcasting that are needed to produce a certificate using Cob Protocol, in terms of the number of components ℓ and ratio of honest users h.

the last Coin-Genuinely-Flipped step. This means that we must consider one more step with message broadcasting after the last Coin-Genuinely-Flipped step. Therefore, the expected number of steps with message broadcasting required by Algorand is:

$$\mathbb{E}\left[\text{Alg steps}_{\frac{h}{2}}\right] = 4 + 3\frac{2}{h} + 1 = 5 + \frac{6}{h} .$$

This is the expected number of steps to reach consensus on a single event. Note that the network must execute an instance of Algorand for each of the ℓ events they observe.

5.2 Weight of the messages

We begin with Algorand: in the first step Algorand selects a small number of nodes (e.g. 35 potential leaders) who must broadcast their proposal, namely the hash of a block created by a shard plus a verifiable credential. The following steps are expected to be performed by 4000 nodes who broadcast messages of 200 bytes as specified in [7]. Since in this use case the weight of the step 1 messages is close to the weight of the other steps, but the number of messages broadcast is much less, we can neglect it while determining the whole amount of broadcast data. The messages created from step 4 onwards weigh around 200 bytes (see [7]) and contain the signature of a single bit, the signature of a digest and a verifiable credential. Finally, the messages created in step 2 and step 3, contain the signature of a digest (the digest of the block created by a shard) together with the verifiable credential of the selected player, therefore we approximate the weight of credential and signature to around 100 bytes, and add the weight of the digest, i.e. 32 bytes, for a total of 132 B.

With Cob, in the first two steps the nodes of the network broadcast a message containing the signature of the list of digests of the blocks created by the shards, together with the verifiable credential of the node who created the message. The weight of such messages can be approximated, with the same reasoning as before, to $32\ell + 100$ bytes. In the following steps, the nodes produce messages containing the signature of a list of ℓ bits, the signature of a digest and a verifiable credential. These messages weight around $\frac{\ell}{8} + 200$ bytes, because they contain the same data as a message of a step s, with $s \geq 4$, of Algorand, once we substitute the single bit of Algorand with the ℓ bit list of Cob.

In Tables 1 and 2 we summarize the weight of the messages for each step of both Algorand and Cob applied to the sharding use case. In Table 1 the weights are associated to the single steps of each protocol, while in Table 2 the weight of messages are associated to the corresponding protocol phase.

5.3 Comparison

We recall that the choice of using a different instance of Algorand for each list component comes from the considerations of Sects. 1.1.1 and 1.1.2. Recall that Algorand is a leader-based consensus protocol, so it is preferable to have one leader (and one protocol instance) for each single event, in order to let consensus achievement on each single component to be independent from the others. In fact, if we used Algorand

Table 1 Weight of messages of each step in a single run of Cob and Algorand

Step	Cob	Algorand
step 1	$32\ell + 100$ bytes	200 bytes ($\star$)
step 2	$32\ell + 100$ bytes	132 bytes
step 3	$\frac{\ell}{8} + 200$ bytes	132 bytes
following steps	$\frac{\ell}{8} + 200$ bytes	200 bytes

The symbol ($\star$) recalls that the first step of Algorand will not be considered in our comparison. The parameter ℓ is the number of events the network must observe. The dimensions reported are derived from the analysis of [7]

Table 2 Weight of messages of each step in the corresponding phase of the Cob and Algorand protocols

Phase	Cob	Algorand
Leader selection	No leader	200 bytes ($\star$)
graded consensus	$32\ell + 100$ bytes	132 bytes
binary agreement	$\frac{\ell}{8} + 200$ bytes	200 bytes

See also Table 1

on the whole list (only a single instance of Algorand), then disagreement on a single component would cause the whole list to be discarded. Therefore, we proceed with the comparison between the execution of Cob and the multiple instances of Algorand.

5.3.1 Algorand

We can put the information of Table 1 together and compute the amount of data broadcast in the network in a single Algorand run. As we mentioned earlier, we will not consider the messages broadcast in the first step because their contribution is practically negligible.

We also recall that, since Algorand is a leader-based consensus protocol, a malicious leader might deliberately produce a proposal which finds all (or the majority) of the nodes in the network at odds. In this case the consensus process drops to the symbol $\perp$ and a certificate is built with step 5 messages. This means that even if the event the network observes is ambiguous, if a leader acts maliciously, agreement will be reached fast. Therefore, since it is not possible to predict how a malicious leader will act, in our comparison we will consider the two border cases: the case when the leader acts honestly, and the case in which every malicious leader will broadcast a controversial message making the consensus drop to $\perp$ in 5 steps.

Case 1 If all leaders (i.e. the players of step 1) act honestly, the amount of data broadcast in the network in a single run is expected to be:

$$\begin{aligned}\mathbb{E}[\text{weight}(\text{Alg}_{1,h,n,\text{honest}})] &= n \cdot \left(2 \cdot 132 + \left(\mathbb{E}\left[\text{Alg steps}_{\frac{h}{2}}\right] - 3\right) 200\right) \\ &= n \cdot \left(264 + \left(2 + \frac{6}{h}\right) \cdot 200\right) .\end{aligned}$$

Multiplying this value by ℓ we obtain the amount of data broadcast in the network by the nodes to reach consensus on ℓ ambiguous events:

$$\mathbb{E}[\text{weight}(\text{Alg}_{\ell,h,n,\text{honest}})] = \ell \cdot n \cdot \left(264 + \left(2 + \frac{6}{h}\right) \cdot 200\right) .$$

Case 2 If all malicious leaders cause the drop of their event data, the total weight slightly decreases. Recall that, in Algorand, a leader is honest with probability at least $p_h = h^2(1 + h - h^2)$ (see [7]). In this case every instance of the protocol whose leader is malicious will end after the broadcast of step 5 and the consensus drops to $\perp$. This means that:

$$\begin{aligned}
&\mathbb{E}[\text{weight}(\text{Alg}_{1,h,n,\text{drop}})] \\
&\quad \geq n \cdot \left(132 \cdot 2 + 200 \cdot 2 + 200 \cdot \left(\mathbb{E}\left[\text{Alg steps}_{\frac{h}{2}}\right] - 5\right) \cdot h^2 \left(1 + h - h^2\right)\right) \\
&\quad = n \cdot \left(264 + 200 \cdot \left(2 + 6h \left(1 + h - h^2\right)\right)\right) .
\end{aligned}$$

Again, multiplying by ℓ we obtain the total amount of data broadcast in the network:

$$\mathbb{E}[\text{weight}(\text{Alg}_{\ell,h,n,\text{drop}})] \geq \ell \cdot n \cdot \left(264 + 200 \cdot \left(2 + 6h \left(1 + h - h^2\right)\right)\right) .$$

5.3.2 Cob

With Cob, the amount of data broadcast in a single protocol run (which covers all the ℓ components) can be computed as:

$$\mathbb{E}[\text{weight}(\text{Cob}_{\ell,h,n})] = n \left(2(100 + 32\ell) + \left(\mathbb{E}[\text{Cob steps}_{\ell,\frac{h}{2}}] - 2\right) \left(\frac{\ell}{8} + 200\right)\right).$$

To give the idea of how Cob can outperform Algorand, we compute the amount of data broadcast by a network where the percentage of honest users is 80% (i.e. $h = 0.8$), and the expected number of players that must broadcast a message, in each step is $n = 4000$ (recall that we are not considering the first step of Algorand). Figures 2 and 3 show the expected total amount of data broadcast for different values of the parameter ℓ, the number of ambiguous components.

We recall that, for the analysis of Algorand, we considered the two border cases, namely all the malicious leaders drop their components to $\perp$ and no malicious leader makes its component to drop to $\perp$. Therefore, in general, the number of MB broadcast in the network in any Algorand protocol execution will reasonably be between the two corresponding lines.

6 Conclusions

We presented Cob, an extension of the MBA protocol [9] which allows the nodes of a wide gossiping network to reach consensus on a list of arbitrary values, working in parallel on each component.

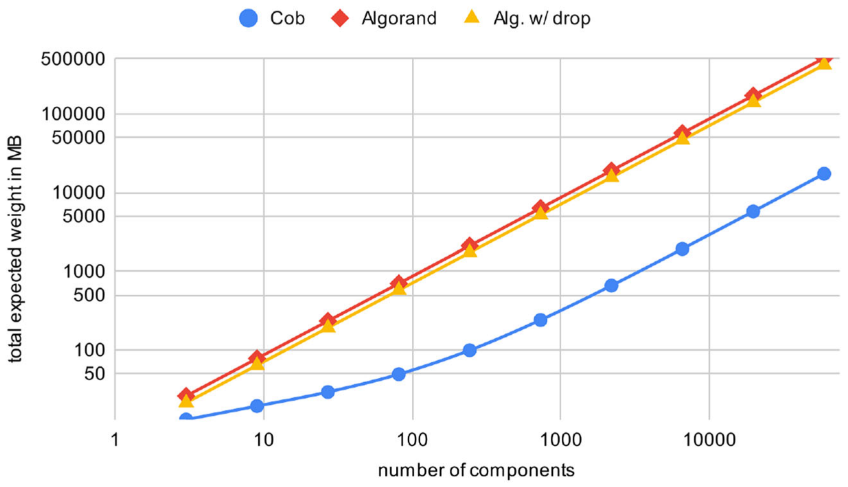

Fig. 2 Amount of data broadcast in the network (in MB) using Algorand or Cob with parameters $h = 0.8$ and $n = 4000$ in terms of the number of components ℓ, logarithmic scale in both axes

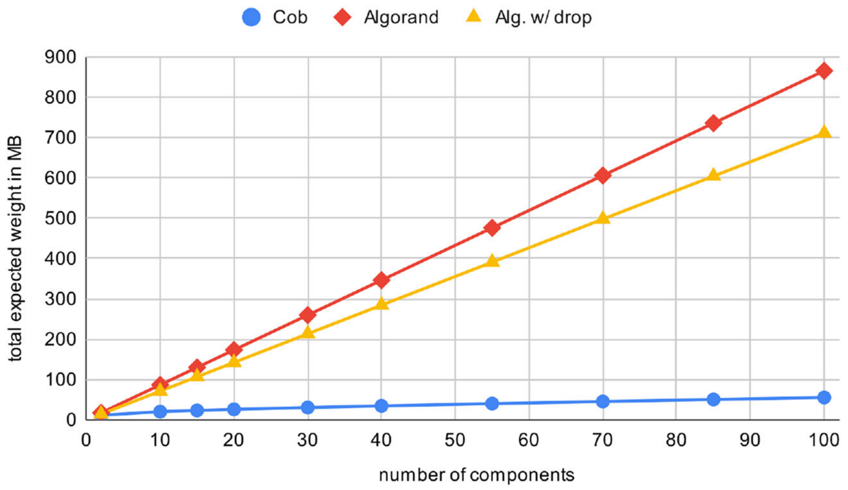

Fig. 3 Amount of data broadcast in the network (in MB) using Algorand or Cob with parameters $h = 0.8$ and $n = 4000$ in terms of the number of components ℓ, linear scale in both axes

This generalization widens the applications of the original protocol thanks to the sortition mechanism that limits the number of messages to be broadcast and processed at each step when there are many players, and the relaxed network assumptions which model real-case scenarios more closely. Notice that the protocol retains the leaderless approach of the MBA protocol, a democratic feature that is valued in permissionless distributed settings and thwarts attacks from malicious leaders. Moreover, it also preserves the parallel approach that enhances efficiency with respect to multiple executions of protocols designed to work in the same environment, such as Algorand.

As we explained in Sect. 1, we believe that one of the most relevant use cases of Cob is as consensus layer to allow the reconciliation of transactions in blockchain platforms implementing sharding. In this context, we proposed a comparison between Cob and the execution of multiple instances of the well-known protocol Algorand, supposing that a network of nodes must reach consensus on which blocks have been legitimately created by different shards. As shown in Figs. 2 and 3, Cob remarkably reduces the amount of data broadcast in the network with respect to multiple executions of Algorand, and this would reasonably speed up the consensus process.

6.1 Future works

Cob guarantees to reach consensus if the assumptions are met, and its leaderless and parallel approach maximizes the number of list components that are finalized on a meaningful value (i.e. $\neq \perp$). However its execution is probabilistic, and although it halts with probability 1, the number of steps necessary to halt have only an upper bound in the form of a Bernoulli-like distribution.

An interesting research direction could focus on extending the protocol by introducing some *termination steps*, in order to have a fixed upper bound on its execution, which would benefit many concrete applications. Specifically, such an extension would see the protocol running normally up to a pre-determined number of steps, then, if the execution has not halted yet, the protocol starts a sequence of termination steps that guarantee to reach a consensus in a fixed number of steps. In this phase it is quite tricky to try to preserve as much meaningful agreement as possible: the trivial solution is to collapse the agreement on $\perp$ if consensus is not reached in time, but avoiding to do so has to account for a wide array of attacks with which malicious players could try to disrupt agreement.

Acknowledgements The core of this work is contained in the first author’s MSC thesis. Part of the results presented here have been carried on within the EU-ESF activities, call PON Ricerca e Innovazione 2014-2020, project Distributed Ledgers for Secure Open Communities. The authors are members of the INdAM Research group GNSAGA. We would like to thank the Quadrans Foundation for their support. The authors would like to thank the anonymous referees.

Funding Open access funding provided by Università degli Studi di Trento within the CRUI-CARE Agreement.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose. The authors have no conflicts of interest to declare that are relevant to the content of this article. The authors have no financial or proprietary interests in any material discussed in this article.

References

1. Meneghetti, A., Parise, T., Sala, M., Taufer, D.: A survey on efficient parallelization of blockchain-based smart contracts. AETiC, 3(5) (2019)
2. Buterin, V.: A next-generation smart contract and decentralized application platform. White Paper, 3(37) (2014)
3. Zilliqa Team.: The zilliqa technical whitepaper. Accessed 16 Sept 2019 (2017)
4. EOS IO.: EOS. IO technical white paper. EOS. IO https://github.com/EOSIO/Documentation. Accessed 18 Dec 2017 (2017)
5. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decent. Bus. Rev. (2008)
6. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)
7. Chen, J., Micali, Silvio: Algorand: a secure and efficient distributed ledger. Theoret. Comput. Sci. **777**, 155–183 (2019)
8. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Annual International Cryptology Conference, pp. 357–388. Springer (2017)
9. Flamini, A., Longo, R., Meneghetti, A.: Multidimensional byzantine agreement in a synchronous setting (2021)
10. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of bft protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–42 (2016)
11. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, pp. 183–192 (1994)
12. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Annual International Cryptology Conference, pp. 524–541. Springer (2001)
13. Cachin, C., Poritz, J.A: Secure intrusion-tolerant replication on the internet. In: Proceedings International Conference on Dependable Systems and Networks, pp. 167–176. IEEE (2002)
14. Mostefaoui, A., Moumen, H., Raynal, M.: Signature-free asynchronous byzantine consensus with t< n/3 and o (n2) messages. In: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing, pp. 2–9 (2014)
15. Duan, S., Reiter, M.K, Zhang, H.: Beat: asynchronous bft made practical. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 2028–2041 (2018)
16. Guo, B., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Dumbo: faster asynchronous bft protocols. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 803–818 (2020)
17. Castro, M., Liskov, B.: Practical byzantine fault tolerance. OSDI **99**, 173–186 (1999)
18. Stathakopoulou, C., David, T., Vukolic, M.: Mir-BFT: high-throughput BFT for blockchains. arXiv:1906.05552 (2019)
19. Gupta, S., Hellings, J., Sadoghi, M.: Rcc: resilient concurrent consensus for high-throughput secure transaction processing. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 1392–1403. IEEE (2021)
20. Rabin, M.O: Randomized byzantine generals. In: 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), pp. 403–409. IEEE (1983)
21. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68 (2017)
22. Cai, S.: Analysis of committee selection mechanism in blockchain. arXiv:1905.05079 (2019)
23. Micali, S.: Byzantine agreement, made trivial (2016)
24. Feldman, P.: Micali, Silvio: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)