Scalable Distributed Agreement from LWE: Byzantine Agreement, Broadcast, and Leader Election

Rex Fernando
Aptos Labs, Palo Alto, CA, USA

Yuval Gelles
The Hebrew University of Jerusalem, Israel

Ilan Komargodski[1][2]
The Hebrew University of Jerusalem, Israel
NTT Research, Sunnyvale, CA, USA

Abstract

Distributed agreement is a general name for the task of ensuring consensus among non-faulty nodes in the presence of faulty or malicious behavior. Well-known instances of agreement tasks are Byzantine Agreement, Broadcast, and Committee or Leader Election. Since agreement tasks lie at the heart of many modern distributed applications, there has been an increased interest in designing *scalable* protocols for these tasks. Specifically, we want protocols where the per-party communication complexity scales sublinearly with the number of parties.

With unconditional security, the state of the art protocols have $\tilde{O}(\sqrt{n})$ per-party communication and $\tilde{O}(1)$ rounds, where n stands for the number of parties, tolerating $1/3-\epsilon$ fraction of corruptions for any $\epsilon > 0$. There are matching lower bounds showing that these protocols are essentially optimal among a large class of protocols. Recently, Boyle-Cohen-Goel (PODC 2021) relaxed the attacker to be computationally bounded and using strong cryptographic assumptions showed a protocol with $\tilde{O}(1)$ per-party communication and rounds (similarly, tolerating $1/3-\epsilon$ fraction of corruptions). The security of their protocol relies on SNARKs for NP with linear-time extraction, a somewhat strong and non-standard assumption. Their protocols further relies on a public-key infrastructure (PKI) and a common-reference-string (CRS).

In this work, we present a new protocol with $\tilde{O}(1)$ per-party communication and rounds but relying only on the standard Learning With Errors (LWE) assumption. Our protocol also relies on a PKI and a CRS, and tolerates $1/3-\epsilon$ fraction of corruptions, similarly to Boyle et al. Technically, we leverage (multi-hop) BARGs for NP directly and in a generic manner which significantly deviate from the framework of Boyle et al.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Theory of computation → Distributed algorithms

Keywords and phrases Byzantine agreement, scalable, learning with errors

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.46

Funding *Rex Fernando*: Algorand Centres of Excellence (ACE) Programme, the Defense Advanced Research Projects Agency under award number HR001120C0086, the Office of Naval Research under award number N000142212064, and the National Science Foundation under award numbers 2128519 and 2044679. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.
Yuval Gelles: ISF Grant No. 1774/20 and BSF-NSF Grant No. 2020643.

[1] Corresponding author.
[2] Incumbent of the Harry & Abe Sherman Senior Lectureship at the School of Computer Science and Engineering at the Hebrew University

Ilan Komargodski: Alon Young Faculty Fellowship, ISF Grant No. 1774/20, and BSF-NSF Grant No. 2020643.

Acknowledgements We thank the anonymous reviewers for instructive suggestions.

1 Introduction

Byzantine Agreement [33, 29] is a classical problem in distributed computing whose aim is to ensure that all non-faulty nodes in a distributed system agree on the same value, even in the presence of faulty or malicious (Byzantine) behavior. This task is tightly related to other well studied distributed agreement tasks such as Broadcast, and Committee or Leader Election. All of the above are fundamental to many distributed systems, such as blockchain protocols (e.g., Algorand [18, 8]) and secure multiparty computation (MPC) protocols [19, 2, 7, 34].

The efficiency of an agreement protocol is measured by the number of rounds of communication required to reach agreement and the amount of bits exchanged in each round. Optimizing the above two is the subject of a rich line of active research. Due to the need to support large numbers of parties, scalability has become an important goal. Ideally, we want to design protocols where the number of rounds and the per-party communication grow (much) slower than the number of parties.

In this work, we consider the synchronous communication model and assume point-to-point channels. We further consider adversaries that corrupt parties statically (after the protocol is specified but before the protocol's execution starts). Our main result is the first protocol which essentially optimal complexity, based on standard falsifiable cryptographic assumptions.

▶ **Theorem 1.1** (Optimal Agreement from LWE). *Assume a public-key infrastructure (PKI), a common reference string (CRS), and the hardness of LWE. Then, there are n-party protocols for Byzantine agreement, broadcast, and committee or leader election, where each of these protocols is secure against a probabilistic polynomial-time adversary that statically corrupts up to $1/3 - \epsilon$ fraction of parties for any constant $\epsilon > 0$. Furthermore, each of these protocols terminates within $\mathsf{poly}(\log n)$ rounds and each party sends $\mathsf{poly}(\lambda, \log n)$ bits overall, where $\lambda \in \mathbb{N}$ is a computational security parameter.*

Our construction, in fact, is obtained using generic usage of a multi-hop non-interactive batch argument (BARG) system for NP.[3] Such BARGs were recently constructed assuming LWE and in the CRS model by [11, 32]. Therefore, any construction of a multi-hop BARG from other assumptions, say group-based, will immediately imply the above result under the other assumptions. (We will explain what BARGs and multi-hop BARGs are below, in Section 2).

Putting the result in context

For convenience, throughout the paper, we use the notation $\tilde{O}(\cdot), \tilde{\Omega}(\cdot)$ to hide poly-logarithmic factors in n, and $\tilde{O}_\lambda(\cdot)$ to additionally hide polynomial factors in λ, the computational security parameter. That is, $\mathsf{poly}(\log n) \equiv \tilde{O}(1)$ and $\mathsf{poly}(\lambda, \log n) \equiv \tilde{O}_\lambda(1)$.

[3] This is an argument system that allows an efficient prover to compute a non-interactive and publicly verifiable "batch proof" of NP instances, with size smaller than the combined witness length. If any of the instances is false, then no polynomial-time cheating prover must be able to produce an accepting proof.

With information theoretic security, the state of the art protocol (in terms of per-party communication complexity) is due to King et al. [28, 26]. They obtained an $\tilde{O}(1)$-round protocol where each party communicates $\tilde{O}(\sqrt{n})$ bits overall (tolerating $1/3-\epsilon$ corruptions). Prior works (e.g., [33, 29, 12, 13, 6]) all had quadratic total communication. Other works that came around the same time or after [28, 26], for example [27, 5, 1], have total quasi-linear communication but they are unbalanced; i.e., few (poly-logarithmically many) parties need to communication with everyone. A recent work [16] gave a computationally-efficient variant of the state of the art protocol of [28, 26]. In another recent work [15], a generic transformation was shown from protocols with worst-case (pessimistic) guarantees to protocols that have optimal guarantees in an honest (optimistic) execution (and same pessimistic guarantees in the worst case).

In terms of lower bounds, few barriers are known. Holtby, Kapron, and King [20] showed that in a large class of Byzantine agreement protocols there is at least one honest party that must send $\tilde{\Omega}(n^{1/3})$ messages. Gelles and Komargodski [16] showed that in the same class of Byzantine agreement protocols, there is at least one honest party that must send *or process* $\tilde{\Omega}(n^{1/2})$ messages.

Allowing cryptography and trusted setup assumptions (that invalidate both of the above-mentioned lower bounds), Boyle, Cohen, and Goel [4] achieve a protocol with essentially best-possible communication: $\tilde{O}(1)$ rounds and per-party $\tilde{O}_\lambda(1)$ bits of communication overall. To do this they introduce a new cryptographic primitive, which they call *succinctly reconstructed distributed signatures* (SRDS), and show that it suffices for distributed agreement. The authors then instantiate SRDS using a PKI and a strong form of succinct non-interactive arguments of knowledge (SNARKs),[4] which are known not to be constructible based on falsifiable assumptions [30, 17]. The authors also give a construction of SRDS using falsifiable assumptions, but in order to achieve this, they modify the PKI model to be much stronger and less meaningful in the context of BA. Specifically, they assume that a single trusted party generates every key pair, where it provides valid secret keys only to a small randomly and secretly chosen subset of the parties; this PKI model can be referred to as a "centrally-generated PKI." The construction of [4] in the centrally-generated PKI model *has the trusted party choose a secret, global committee, and programs knowledge of this committee into each member's secret key.* Thus, their construction can be seen as a way to bootstrap a single BA output (i.e., the choice of a global committee) in this model to achieve more BA outputs. It is unclear, however, what this result means in the context of actually achieving a BA protocol from scratch, since as explained before, it is assumed that a trusted party programs the output of a BA protocol into the trusted setup. Note that this is very different from the standard definition of PKI, where each party publishes a public key that it is allowed to choose arbitrarily. The authors of [4] pose the existence of an efficient BA protocol from falsifiable assumptions and in the non-trusted PKI model as the main open question in this line of work.

In Theorem 1.1 we answer this question in the affirmative, obtaining the same efficiency properties of the resulting protocol as [4] (in the non-trusted PKI model) except that we rely on the standard LWE assumption instead of (strong variants of) SNARKs. Notably, LWE

[4] A SNARK is a proof system that enables a prover holding a witness w to some public NP statement x to convince a verifier that it indeed knows w by sending a single message. The proof string is succinct in the sense that it is much shorter than the witness w, and knowledge is formalized via an efficient extractor that succeeds extracting w from a malicious prover P^* with roughly the same probability that P^* convinces an honest verifier. The work of [4] assumes linear extraction, i.e., where the size of the extractor is linear in the size of the prover.

is both falsifiable and believed to be quantum-secure, which has made it an attractive and widely-used assumption. This gives the first agreement protocols with optimal complexity (up to poly-logarithmic factors) relying on standard falsifiable assumptions. Notably, we do *not* achieve our result by following their template of instantiating SRDSs using LWE; in fact, this is still an open problem. Instead, we suggest a completely different framework for achieving scalable agreement protocols utilizing the power of quorums and multi-hop aggregate signatures. Our approach is arguably much simpler and more direct. We stress that unlike the "centrally-generated PKI model," our protocol does not require the public-key infrastructure to guarantee *any* sort of agreement. Namely, our protocol is a secure BA protocol even if a malicious party sends inconsistent public keys to different other parties. We give a detailed overview of our ideas in Section 2.

On the importance of relying on falsifiable assumptions

Roughly speaking, a falsifiable assumption [30] is one that can be phrased and modeled as an interactive game between an efficient challenger and an adversary, at the conclusion of which the challenger can *efficiently* decide whether the adversary "won" the game. The assumption states that every efficient adversary has at most a negligible winning probability in the game. Most standard cryptographic assumptions are falsifiable, including general notions (e.g. one way functions, trapdoor permutations, oblivious transfer, fully homomorphic encryption, etc.) and concrete assumptions (e.g. hardness of factoring, discrete logarithms, shortest vector problem, RSA, CDH, DDH, LWE etc.). The above notion of falsifiability captures the fact that the challenger gives us an efficient process to test whether an adversarial strategy breaks the assumption. Since the introduction of this notion, falsifiability has been an important criteria of "reasonable" or "standard" assumptions.

Intuitively, assumptions that are not falsifiable are harder to reason about, and therefore we have significantly less confidence in them. SNARKs are a common examples for a primitive that is by-itself *non-falsifiable* because for a challenger to decide whether an adversary produces a proof of a false statement, it needs to decide whether an NP statement is true or false, which could be hard in general. Additionally, it is well known that it is impossible to construct (in a certain black-box manner) SNARKs from any falsifiable assumption [17], making it an assumption that we would prefer to avoid whenever possible.

2 Technical Overview

We consider the following goal that will turn out to suffice for all of our applications. We want to build a communication tree, where every internal node in the tree corresponds to a small set of parties and every one of the n leaf nodes is virtually associated with a unique party. The root node is called the supreme committee. Intuitively, every internal node is a "committee" that certifies communication from the parent of the node to its children and vice versa. At a high level, our goal is to maximize the number of internal nodes that correspond to "good" committees (in particular, containing a majority of honest parties) and also maximize the number of leaf nodes that can communicate with the root (supreme committee) by talking only with other "good" committees (this is explained more precisely below). Obviously, if the members in each node are fixed and known, then an adversary with a sufficient corruption budget can corrupt a majority of parties in many internal nodes and eventually prevent reliable communication. Thus, while the tree structure is fixed and known ahead of time, the members in each node will be selected "on the fly". For efficiency purposes, we want each non-leaf node to contain $\tilde{O}(1)$ members and that each party is a member in $\tilde{O}(1)$ nodes. The tree will have fan-out $\tilde{O}(1)$ and the depth of the tree is $\tilde{O}(1)$.

Let $\mathcal{H}$ be the set of honest parties. A resulting communication tree will induce a set $K \subseteq \mathcal{H}$ such that for every $p \in K$, it holds that

1. The path from the node associated with p to the root is "good", where "good" means that the majority of each node members along the path are from K. That is, for every node N along the path between leaf p to the root (including the root), it holds that $|N \cap K| > |N|/2$.
2. For each node in the tree, each node member from K knows the members of its neighboring nodes (parent and children).

Note that in the above two properties we do not require anything for parties in $\mathcal{H} \setminus K$ and so it is convenient to think of them as behaving adversarially.

If we manage to generate such a tree with $K = \mathcal{H}$, then we are essentially done. The supreme committee members (i.e., the $\tilde{O}(1)$ parties associated with the root node) can perform any polynomial complexity protocol to compute Byzantine Agreement, or Committee/Leader Election, and distribute the result to all the parties using the tree. Indeed, since any honest party has a direct (and short) path to the root, involving only nodes with an honest majority, it is guaranteed that it will learn the right value. The main challenge is therefore to come up with such a communication tree.

Prior work: using the almost everywhere communication tree. Already in the original work of [28], a communication tree as above was constructed. However, instead of achieving it with respect to $K = \mathcal{H}$, they achieved it with respect to some $K' \subset \mathcal{H}$, where $|\mathcal{H} \setminus K'| \in \Theta(n/\log n)$. This is also why they achieved only almost everywhere agreement rather than everywhere agreement.

Boyle et al. [4] worked directly with the tree of [28] and came up with a new primitive that suffices to make sure that the adversary can neither prevent an honest party from learning the right output or cause it to accept a wrong value. This new primitive is called *succinctly reconstructed distributed signatures* (SRDS), and allows the parties to aggregate cryptographic signatures along a tree. SRDS have a "robust threshold" property: if enough (say $2/3 + \epsilon$) of the nodes are honest and agree to sign, then even if some internal nodes of the tree are malicious, these malicious nodes cannot prevent a signature from being generated. On the other hand, if few enough (say $1/3 - \epsilon$) nodes agree to sign, then the malicious nodes cannot produce a signature.

The authors of [4] give two instantiations of SRDS. The first is using a PKI and a strong form of (recursively composable) succinct non-interactive arguments of knowledge (SNARKs). Specifically, since the communication tree is of logarithmic depth they eventually need to perform recursive composition log-many times. Thus, to bound the complexity of the knowledge extractor by a fixed polynomial, they need to assume that a single extraction has only constant multiplicative overhead. The second instantiation only requires one-way functions as a cryptographic assumption, but works in the much stronger and less standard *centrally-generated PKI model*, where it is assumed that a central, trusted party generates *all* key pairs. Crucially, such a setup essentially allows to secretly choose and identify a random committee, almost solving the problem we want to solve in the first place.

(Multi-hop) BARGs to the rescue? We want to work in the standard PKI model, and to also eliminate the need for SNARKs. A natural attempt is to try to use the exciting recent line of work on BARGs. Recall that a non-interactive batch arguments (BARG) [10, 9], given an NP-language, is an argument system that allows an efficient prover to compute a non-interactive and publicly verifiable "batch proof" of the conjunction of k NP instances, with size much smaller than the combined witness length. If any of the k instances is false, then no polynomial-time cheating prover must be able to produce an accepting proof. The multi-hop variant [11] allows to compose proofs, enabling mutually distrustful parties to perform distributed computations in a verifiable manner. See Section 3.2.2 for the precise definition. BARGs are nowadays known from multiple assumptions like LWE [23, 10], bilinear maps [25, 36, 24], decisional Diffie-Hellman (DDH) and quadratic residuosity [9, 22], or obfuscation [14], and the multi-hop variant in particular is known from LWE [11, 32].

Unfortunately, the specific structure of BARGs makes it clear that naively replacing SNARKs with (multi-hop) BARGs in the protocol template of [4] does not work. At a high level, [4]'s certification mechanism consists of a tree of SNARKs, where each non-leaf SNARK has a "running count" of how many valid signatures are present in the corresponding subtree. So a particular node n will have left and right children (c_l, π_l) and (c_r, π_r), where c_l and c_r are counts of the signatures, and π_l and π_r are SNARKs which attest to those signatures (or are signatures themselves, if l and r are leaf nodes). Upon seeing these values, node n will set $c_n = c_l + c_r$, and generate π_n that attests that it saw proofs for its child nodes, *and that they sum up to* c_n. Multi-hop BARGs also allow aggregation of proofs in a tree, but they do not allow this summing-of-counts logic in the internal nodes. With BARGs, the leaves of the tree correspond to NP statements, and the only allowed logic is "all-or-nothing": either all statements are true, and the root-node proof can be generated attesting to this, or if any statement is not true, then the aggregation fails.

Our approach. To alleviate this problem, we completely deviate from the above template and construct a new communication tree which still satisfies the above properties for $K' \subset \mathcal{H}$ but we additionally have the following important guarantee: *All internal (non-leaf) nodes have a majority of parties from* $\mathcal{H}$.

Why does this suffice? Each internal node in the tree will collect signatures on the candidate string from its members. Because each internal node has an honest majority from K', we are guaranteed that the right candidate string will form a majority. Now (because all internal nodes are "honest virtual nodes"), we can use multi-hop BARGs to aggregate the validity of the signatures along the tree. Once all lower level nodes hold an aggregate signature they can send it to their associate parties. The latter are then convinced that this value was held by more than 1/2 of the parties and so it must be correct.

How do we achieve it? One can utilize the communication tree of [28] in the following way: ask the members of the supreme committee to jointly sample a *succinct representation* of a new random tree. Say, this can be done by sampling (at the root node) a key for a pseudo-random function (PRF), disseminating it to everyone (along the tree), and locally expanding it to get a new (pseudo-random) new tree. This will satisfy the above property that all internal nodes will have an honest majority (with all but negligible probability of error).

The above idea almost works. The only problem is that there might exist some "bad" PRF key, where the whole tree generated by this key consists of dishonest-majority nodes. If such a key exists, the adversary can choose this key, generate a BARG-based signature which is signed by all nodes in the tree (since they are all majority-dishonest), and present this to any of the honest parties in $\mathcal{H} \setminus K'$. Any such party in $\mathcal{H} \setminus K'$ will have no way of distinguishing this malicious signature from the honest signature of the true agreed-upon value. Thus, this scenario allows the adversary to prevent full agreement.

We need some way to prevent generating such a "terrible" tree, i.e., in every possible choice of a tree, there must be at least one node that has an honest majority. We achieve this property by using a particular method for getting the tree. Instead of choosing it (pseudo-) randomly, we choose only one node at random. We then "shift" this node members in a particular way so that we are guaranteed that (1) w.h.p all nodes consist of an honest majority and (2) for *every* originally sampled node, there must be at least one node that has an honest majority. See details in Section 4.

The idea of using shifts is somewhat similar to a recent idea due to [16], however, the setting is a bit different and we end up needing new properties. Specifically, in [16], the idea of using shifts was applied to a seed that was sampled differently, and further they use this idea for a different purpose than to generate a tree. In particular, the "no terrible tree" analysis, and in general the entire portion of our work which concerns the communication tree, is novel to this work.

3 Model and Preliminaries

Notation. For any distribution $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the distribution $\mathcal{X}$. For a set X we denote by $x \leftarrow X$ the process of sampling x from the uniform distribution over X. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, .., n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Throughout this paper, all machines are assumed to be non-uniform. We will use λ to denote the computational security parameter. We will use PPT as an acronym for "probabilistic polynomial-time".

Model. We consider n parties in a fully connected network (i.e., clique). Each party has a unique ID and the IDs are common knowledge. We assume the IDs are $1, \ldots, n$. Parties are uniform and efficient algorithms, i.e., they can be represented as Turing machines that run in polynomial time in the (common) security parameter given in unary representation. Parties can perform arbitrary computation and they have a (private) source of randomness. Communication is private and authenticated, that is, whenever a party sends a message directly to another, the identity of the sender is known to the recipient and moreover the content of the message is hidden from the adversary.[5] We assume synchronous communication. That is, communication proceeds in rounds; messages are all sent out at the same time at the start of the round, and then received at the same time at the end of the same round. All parties have synchronized clocks.

We assume that there is a PPT adversary that controls up to t parties and whose goal is to cause the protocol to fail in some way, depending on the context. The adversary is allowed to be *non-uniform* PPT (while the honest parties need only be uniform). The adversary chooses which parties to control non-adaptively, i.e., it chooses the set of corrupted parties at the start of the protocol. The adversary is malicious: corrupted parties can engage in any kind of deviations from the protocol and send arbitrary messages in the name of the corrupted parties. We emphasize that corrupted parties can send arbitrarily long messages to arbitrary other parties. We assume that the adversary is rushing, that is, it can view all messages sent by the honest parties in a round before the corrupted parties send their messages in the same round.

[5] Since our constructions rely on a public-key infrastructure it is easy to "implement" these properties (e.g., by using authenticated encryption).

We further consider protocols in the PKI model, where each party $i \in [n]$ samples for itself a secret key sk_i and a public verification key vk_i from some distribution. The adversary can replace the vk's of corrupted parties arbitrarily and even send inconsistent public keys to different parties. We further assume a common reference string (CRS) which is sampled once at the beginning of time by a trusted party (alternatively, one can think of the CRS as being non-uniformly hardwired into the protocol's description). The CRS is known to all parties as well as to the adversary. We emphasize that the adversary chooses which parties to corrupt after seeing the CRS and after seeing all (honest) vks, but before the protocol's execution begins.

The adversary's advantage, and the relation between n and λ. In this paper, we say that a protocol is secure if a computationally-bounded adversary cannot contradict any of its properties except with negligible probability. By this, we mean that the probability of the adversary succeeding should be bounded by $\mathsf{negl}(n) + \mathsf{negl}(\lambda)$, where n is the number of parties, and λ is a security parameter which is used to instantiate the cryptographic primitives used in the protocol. For this to be meaningful, we can either assume that λ and n are polynomially related (i.e., n is upper-bounded by an unspecified polynomial $p(\cdot)$ in terms of λ), and rely on polynomial-hardness assumptions, or we can rely on sub-exponential hardness assumptions and relax this relation, saying that n can go up to $2^{\lambda^{\Theta(1)}}$. We stress that even in the polynomial-hardness regime, none of the constructions in this paper require knowing or specifying the bounding polynomial p upfront in order to work.

3.1 Definition of Byzantine Agreement and Friends

Byzantine agreement. In this problem, there are n parties, $P_1, \ldots, P_n$. At most t of the parties may be corrupted. Each party P_i begins with an input bit $x_i \in \{0, 1\}$ and outputs a bit y_i. The goal of the protocol is (with high probability) for all honest parties to terminate and, upon termination, agree on a bit held by at least one honest party at the start. This should hold even when the t corrupted parties collude and actively try to prevent it. More precisely, the Byzantine agreement problem is defined as a protocol that satisfies with high probability the following *agreement*, *validity*, and *termination* properties:

- **Agreement:** For every pair of honest parties P_i and P_j it holds that $y_i = y_j$.
- **Validity:** If there exists a bit x such that for every honest party P_i it holds that $x_i = x$, then the common output is x.
- **Termination:** Every honest party eventually outputs a bit.

Committee election and broadcast. In **committee election**, the goal is to bring all honest parties to agree on a small subset of parties with a fraction of corrupted parties close to the fraction for the whole set. In the extreme case, the committee is of size 1 in which case a single leader is chosen (i.e., **leader election**) and the goal is to guarantee that the leader is an honest party with constant probability. In **broadcast**, the goal is to allow an honest party to communicate a message to all other parties so that the message that all other honest parties end up knowing is the same as the one sent.

3.1.1 Quorum to Byzantine Agreement, Broadcast, and Committee Election

It is not at all obvious if one can translate a protocol for one of the above tasks into another one *while preserving scalability.* One naive idea is to choose a small committee and let it do most of the work. That is, elect a small committee and run a "heavy-weight" protocol among the committee members for task X, where X could be (for instance) leader election, Byzantine agreement, or broadcast. If the committee is sufficiently small, then we can afford to execute rather inefficient protocols among the committee members. The problem with this approach is that it results with highly unbalanced protocols: members would need to communication with all other parties.

There is a useful abstraction called a **quorum** that does imply all of the above applications as a special case. A quorum is a collection of n committees where (1) each committee contains roughly the same fraction of bad parties as in the total population, (2) Each party is a member in roughly the same number of committees. Once we have a quorum, it is relatively easy to obtain all other above-mentioned abstractions with essentially no extra complexity (as noted in [28]). For completeness, we sketch the details next.

Consider the Byzantine agreement problem first. Given a quorum, each party sends its bit to its associated committee members. The committees now perform agreement in a tree-like fashion by say computing the majority value of their input bits. Lastly, all committees distribute the agreed upon bit to their respective associated parties. Broadcast is implemented in a similar fashion by performing broadcast in a tree-like fashion between the committees defined via the quorum (again, since these committees are behaving honestly, this is easy). Committee election is a special case of quorum (the first committee in the quorum is a good committee). Leader election can be implemented by running a naive (heavy-weight) protocol among a chosen committee.

3.2 Cryptographic Building Blocks

3.2.1 Digital Signatures

A digital signature allows one party to "convince" another party that a third party indeed "approved" a given message. The notion that suffices for this paper is that of *existential unforgeability*, where it is required to be computationally infeasible to generate a valid message-signature pair even after seeing as many such pairs for different messages as needed. We first describe the syntax and then formalize the security notion via a game between a challenger and an adversary.

A signature scheme consists of three polynomial-time procedures $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to \mathsf{vk}, \mathsf{sk}$: given security parameter 1^λ, the procedure outputs a verification key vk and signing key sk pair.
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$: given signing key sk and a message m, the procedure outputs a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma) \to \{0,1\}$: given verification key vk, message m, and a signature σ, the procedure outputs 0 (for "fail") or 1 (for "succcess").

Correctness. A signature scheme should satisfy that honestly generated signatures verification always succeed. That is, for all $\lambda \in \mathbb{N}$, $\mathsf{vk}, \mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$, and $m \in \{0,1\}^*$, it holds that

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1.$$

PKI existential unforgeability. For any PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that the probability to win the unforgeability game is at most $\mathsf{negl}(\lambda)$. That is,

$$\Pr[\mathsf{ExUnfg}_{\mathcal{S},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{ExUnfg}_{\mathcal{S},\mathcal{A}}(\lambda)$ is defined as:

1. $\mathcal{A}(1^\lambda)$ specifies an integer k by writing it in unary.
2. Initialize $\mathcal{M} = \emptyset$ and run $\{(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda)\}_{i\in[k]}$. Send $\{\mathsf{vk}_i\}_{i\in[k]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ chooses a set $\mathcal{C} \subseteq [k]$ and gets back $\{\mathsf{sk}_i\}_{i\in\mathcal{C}}$.
4. $\mathcal{A}$ gets access to a signing oracle that gets as input pairs of the form (i, m) for $i \in [k] \setminus \mathcal{C}$ and returns $\sigma_{i,m} = \mathsf{Sign}(\mathsf{sk}_i, m)$. $\mathcal{A}$ makes as many queries as it wants (adaptively), and upon every such query, the experiment updates $\mathcal{M} = \mathcal{M} \cup \{(i, m)\}$. When $\mathcal{A}$ queries $(\perp, \perp)$, the experiment proceeds to the next step.
5. $\mathcal{A}$ outputs (i^*, m^*, σ^*) and the experiment outputs 1 (i.e., adversary wins) if (1) i^* is not among the set of corrupted keys $\mathcal{C}$, (2) (i^*, m^*) does not belong to the list of queries $\mathcal{M}$, and (3) the signature σ^* is a valid signature of m^* with respect to vk_{i^*}. That is, $i^* \in [k] \setminus \mathcal{C} \ \wedge \ (i^*, m^*) \notin \mathcal{M} \ \wedge \ \mathsf{Verify}(\mathsf{vk}_{i^*}, m^*, \sigma^*)$.

▶ Remark 1 (Equivalence to standard existential unforgeability). The standard definition of existential unforgeability of signatures is a special case of the above where $k = 1$ and $\mathcal{C} = \emptyset$. By a standard "guessing" argument the two notions are existentially equivalent. That is, to translate an attacker of the above definition to an attacker in the "single-key" variant, we can embed the challenge vk into the list of vk at a random index. If this index ends up being corrupted (i.e., in $\mathcal{C}$), the game aborts. Otherwise, the game proceeds normally. A successful attack to the PKI existential unforgeability game will translate into a successful attack in the single-key existential unforgeability game with probability smaller by $1/n$ factor.

The above definition of digital signatures is standard and satisfied by many known constructions. For instance, the one-way function-based construction satisfies it [31, 35] (which then implies an LWE-based construction, directly).

3.2.2 Multi-Hop BARGs for NP

Let $R(\cdot, \cdot)$ be an NP relation, and let $x_1, \ldots, x_n$ be n public statements. A BARG scheme stands for publicly verifiable non-interactive argument system for batch-NP. Such a scheme allows a prover to prove that it knows n witnesses such that for all $i \in [n], R(x_i, w_i) = 1$, using total communication $o(n)$. A multi-hop BARG scheme extends this notion to a distribute system where there are n provers, each one holding a different witness. A multi-hop BARG allows the provers to jointly and aggregate a proof that they know n witnesses using $o(n)$ communication for each prover. In two concurrent recent works [11, 32], a multi-hop BARG was given. We give the definition of multi-hop BARG from [11], except that we simplify it slightly by fixing the aggregation pattern to be in a full binary tree-like manner.

▶ **Definition 2** (Multi-hop BARG scheme with fixed aggregation pattern). *Let $R(\cdot, \cdot)$ be a NP-relation and let $\lambda, n \in \mathbb{N}$ be two parameters. Let $X = (x_1, \ldots, x_n)$ be n public statements, and let the aggregation pattern be the full binary tree with n labeled nodes (denoted T). A multi-hop BARG scheme with fixed aggregation pattern for R consists of the following (randomized) polynomial time algorithms:*

- $\mathsf{Gen}(1^\lambda, i^*) \rightarrow \mathsf{crs}, \mathsf{td}$. *The setup algorithm receives as input the security parameter λ, and an index of statement i^*. It returns the* crs *(common reference string) and trapdoor* td.
- $\mathcal{P}(\mathsf{crs}, x, w) \rightarrow \pi/\perp$. *The proof generation algorithm receives the common reference string* crs*, an instance x, and a potential witness w. If $R(x, w) = 1$, it outputs a proof π; otherwise, it returns $\perp$.*

- $\mathsf{AggProve}(\mathsf{crs}, X, i, \pi_i, \pi_L, \pi_R) \to \pi$. *The aggregation algorithm receives as input the n instances X, an index i in the aggregation tree, the proof π_i associated with the ith instance x_i, and the aggregated proofs π_L and π_R of the left and right subtrees of index i, respectively. It returns an aggregated proof π associated with the subtree of index i.*
- $\mathcal{V}(\mathsf{crs}, X, i, \pi) \to \{0, 1\}$. *The verification algorithm gets as input the n instances X, an index i, and a proof π. It returns 1 if π is valid proof for the aggregated statements with respect to the subtree; otherwise, it returns 0.*

The scheme need to satisfy the following:

- ***Efficiency.*** *The size of* $\mathsf{crs}, \mathsf{td}$ *is at most $\tilde{O}_\lambda(1)$, and the size of a (combined) proof corresponding to T is at most $\max_{i\in[n]} |w_i| + \tilde{O}_\lambda(1)$.*
- ***Correctness.*** *For any* $\mathsf{crs}, \mathsf{td} \leftarrow \mathsf{Gen}(1^\lambda, i^*)$*, and for any X and valid witnesses:*
  - *For every leaf j of T:*

$$\mathcal{V}(\mathsf{crs}, X, j, \mathcal{P}(\mathsf{crs}, x_j, w_j)) = 1.$$

  - *For every internal node $k \in [n]$ the following holds. Let k_L and k_R be k's left and right children, respectively. Assume that π_L and π_R are proofs associated with k_L and k_R such that $\mathcal{V}(\mathsf{crs}, X, k_L, \pi_L) = 1$ and $\mathcal{V}(\mathsf{crs}, X, k_R, \pi_R) = 1$. Then,*

$$\mathcal{V}(\mathsf{crs}, X, k, \mathsf{AggProve}(\mathsf{crs}, X, k, \mathcal{P}(\mathsf{crs}, x_k, w_k), \pi_L, \pi_R)) = 1.$$

- ***Index hiding.*** *For any* PPT *adversary $\mathcal{A}$, and any $i, j \in [n]$:*

$$\left| \Pr_{\mathsf{crs}\leftarrow\mathsf{Gen}(1^\lambda,i)}[\mathcal{A}(\mathsf{crs}, i, j) = i] - \Pr_{\mathsf{crs}\leftarrow\mathsf{Gen}(1^\lambda,j)}[\mathcal{A}(\mathsf{crs}, i, j) = i] \right| \leq \mathsf{negl}(\lambda).$$

- ***Somewhere argument of knowledge.*** *There exists a* PPT *extractor $\mathcal{E}$, such that for any* PPT *adversary $\mathcal{A}$:*

$$\Pr_{\substack{\mathsf{crs}\leftarrow\mathsf{Gen}(1^\lambda,i^*)\\(i,\pi)=\mathcal{A}(\mathsf{crs})}}[\mathcal{V}(\mathsf{crs}, X, i, \pi) = 1 \wedge R(x_i, \mathcal{E}(X, \mathsf{td}, i^*, i, \pi)) = 0] \leq \mathsf{negl}(\lambda).$$

▶ **Theorem 3.3** (Theorem 1.3 in [11]). *Assume hardness of* LWE *against non-uniform polynomial-time attackers. A multi-hop BARG for all NP satisfying Definition 2 exists.*

Above, we assume that n, the number of instances in the batch, and λ, the computational security parameter, are polynomially related. One can support even n that grows sub-exponentially with λ (or, alternatively, λ that is only poly-logarithmic in n) if we assume sub-exponential hardness of LWE.

Our agreement protocol is eventually proven secure by a reduction to the security of the multi-hop BARG and an underlying signature scheme. We have stated the existence of multi-hop BARGs from LWE because this is the way that [11, 32] state it. However, the latter works actually show a generic construction of multi-hop BARGs from two primitives (1) somewhere-extractable hash functions (SEH) [21, 10], and (2) standard batch arguments (i.e., not necessarily multi-hop). These were first constructed from LWE [21, 10], and more recently using other assumptions (e.g., DLIN or subexponentially-hard DDH) [24]. By relying on these results (and noting that signatures exist too under these assumptions), we can get our results also from those assumptions. Lastly, we mention that our protocols only require multi-hop BARG that supports logarithmically-many hopes.

4 Quorum Agreement Protocol

In this section we provide a scalable protocol used to agree on a "good" quorum. Recall that a quorum is a collection of n committees $C_1, \ldots, C_n \in [n]^d$ (i.e., size d subsets of parties). In our context, each committee will be of size $d = \log^2 n$ and a quorum is good if the following conditions hold. Denote by B the set of parties the adversary controls, and recall that $|B| \leq (1/3 - \epsilon)n$ for $\epsilon > 0$.

1. For each $i \in [n]$, $|C_i \setminus B| > d/2$. That is, there is an honest majority in each committee.
2. For each $j \in [n]$, $\sum_{i \in [n]} |C_i \cap \{j\}| \in O(d)$. That is, each party appears in about d committees.

The protocol that we describe in this section results with a good quorum as described above, and does so using only $\tilde{O}_\lambda(1)$ per-party communication and $\tilde{O}(1)$ rounds. All of our protocols succeed with very high probability, i.e., all but negligible probability of error in the security parameter and in n.

▶ **Theorem 4.1.** *Assume a PKI, a CRS, and the hardness of LWE. Then, there is a protocol for generating a good quorum (with probability at least $1 - \mathsf{negl}(n)$), secure against a* PPT *adversary that statically corrupts up to $1/3 - \epsilon$ fraction of parties for any constant $\epsilon > 0$. Furthermore, the protocol terminates within $\tilde{O}(1)$ rounds and each party sends $\tilde{O}_\lambda(1)$ bits overall.*

As immediate corollaries, we obtain protocols for classical agreement tasks such as byzantine agreement, broadcast, and committee or leader election. The construction that achieves this transformation is immediate and is described in Section 3.1.1.

▶ **Theorem 4.2** (Restatement of Theorem 1.1)**.** *Assume a PKI, a CRS, and the hardness of LWE. Then, there are protocols for byzantine agreement, broadcast, and committee or leader election, secure against a* PPT *adversary that statically corrupts up to $1/3 - \epsilon$ fraction of parties for any constant $\epsilon > 0$. Furthermore, these protocols terminate within $\tilde{O}(1)$ rounds and each party sends $\tilde{O}_\lambda(1)$ bits overall.*

Section organization. In Section 4.1 we present a protocol that makes almost all parties agree on a random string. That is, at the end of this phase, each party i holds a string seed_i. For all but $o(1)$ fraction of parties, there is a string seed such that all seed_i's are equal to it. The $o(n)$ parties that do not hold seed are called unknowledgeable. There is no guarantee on the seed_i of the unknowledgeable parties. Furthermore, seed is distributed uniformly from $\{0,1\}^{\log^3 n}$. It is not guaranteed that a party knows whether it is knowledgeable or not. Then, in Section 4.2 we show that the output of the previous protocol can be used to sample from a strong family of good quorums. That is, we construct a family of good quorums with additional properties that are crucial for the next step in our protocol. Lastly, in Section 4.3 we preset the full protocol that causes all parties to agree on seed and thereby implies a good quorum.

4.1 Almost Everywhere Agreement on a Seed

Recall that [28]'s protocol establishes a $O(\log n)$-degree communication tree (which is essentially a sparse overlay network) in which each node is assigned with a committee of $\tilde{O}(1)$ parties. The guarantees are that the $\tilde{O}(1)$-size root committee has a 2/3 honest majority and almost all of the parties are connected to the supreme committee via the communication tree. Denote *root* the root committee. We run a standard MPC protocol among the parties in the root of the tree to sample fresh seed of length $\log^3 n$.

<u>AlmostEverywhereAgreementOnASeed</u>:

1. All the parties runs the protocol from [28] to generate the communication tree.
2. The parties in the root node perform the MPC protocol of BGW [2] to calculate $\mathsf{seed} = \mathsf{XOR}_{i \in root} \mathsf{seed}_i$, where for each party in the root $\mathsf{seed}_i \leftarrow \{0,1\}^{\log^3 n}$. Recall that BGWs protocol requires private channels and a broadcast channel. The former we assume as part of the model (and this can easily be implemented using a PKI) and the latter we implement via polynomial (but not necessarily scalable) broadcast protocol, say [3] (remember that we run BGW among poly-logarithmically many parties).
3. The root node distributes seed to the leaves using the communication tree from step 1.
4. Each leaf node N_i sends seed to party P_i, where P_i saves only the value that it received mostly.

▶ **Theorem 4.3.** *For any* PPT *adversary $\mathcal{A}$ that statically corrupt up to $1/3 - \epsilon$ parties, for any constant $\epsilon > 0$, with probability at least $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$, at the end of the protocol, $1 - o(1)$ fraction of honest parties agree on a uniform string $\mathsf{seed} \in \{0,1\}^{\log^3 n}$. In addition, the protocol terminates after $\tilde{O}(1)$ rounds, and each party sends $\tilde{O}_\lambda(1)$ bits overall.*

The proof is by direct composition of the results from [28, 2, 3]; see the full version for details.

4.2 Sampling a Strong Quorum

Recall that a quorum is a collection of n committees. Roughly, a quorum is good if each committee has a fraction of corrupt parties similar to the fraction of total corrupt parties (out of n). Also, we shall require a balancedness property meaning that each party appears roughly in the same number of committees. These properties have been considered before and are not new to this work. We introduce and require a new property called *no terrible quorums* meaning that in every possible (adversarially chosen) quorum there is at least one "good" committee.

▶ **Definition 4** (Good family of quorums). *A family of quorums is a family $\mathcal{Q} = \{\mathcal{Q}_n\}_{n \in \mathbb{N}}$. For each $n \in \mathbb{N}$, each member $Q \in \mathcal{Q}_n$ is a set of n committees $Q = (C_1, \dots, C_n) \in ([n]^d)^n$, where the size of each committee is $d = \log^2 n$. Note that each C_i is a multi-set. The family $\mathcal{Q}$ is said to be good if*

1. ***Efficient sampling:*** *There is a deterministic polynomial-time procedure* $\mathsf{QSample}(1^n, \mathsf{seed})$ *that gets as input $n \in \mathbb{N}$ and $\mathsf{seed} \in \{0,1\}^{\log^3 n}$, and outputs a quorum $Q \in \mathcal{Q}_n$.*
2. ***Good committees:*** *For every $n \in \mathbb{N}$, every $H \subseteq [n]$ such that $|H| > n/\log\log n$, with high probability over the choice of the quorum $Q = (C_1, \dots, C_n)$, the density of H-members in C_i is very close to their density in $[n]$. That is,*

$$\Pr_{\substack{\mathsf{seed} \leftarrow \{0,1\}^{\log^3 n},\\ (C_1,\dots,C_n) = \mathsf{QSample}(1^n,\mathsf{seed})}} \left[\forall i \in [n] \colon \frac{1}{d} \cdot \sum_{z \in C_i} \mathbb{1}_{z \in H} \geq \left(1 - \frac{1}{\log\log n}\right) \frac{|H|}{n} \right] \geq 1 - \mathsf{negl}(n).$$

3. ***Balanced committees:*** *For every $n \in \mathbb{N}$, with high probability over the choice of the quorum $Q = (C_1, \dots, C_n)$, every party appears in $O(d)$ committees overall. That is,*

$$\Pr_{\substack{\mathsf{seed} \leftarrow \{0,1\}^{\log^3 n},\\ (C_1,\dots,C_n) = \mathsf{QSample}(1^n,\mathsf{seed})}} \left[\forall j \in [n] \colon \sum_{i=1}^{n} \sum_{z \in C_i} \mathbb{1}_{z = j} \in O(d) \right] \geq 1 - \mathsf{negl}(n).$$

4. ***No terrible quorums:*** *For every $n \in \mathbb{N}$ and every $H \subseteq [n]$, for every possible choice of a quorum $Q = (C_1, \ldots, C_n)$, there exists at least one committee C_i where the density of H-members in C_i is at least $|H|/n$. That is, for every* $\mathsf{seed} \in \{0,1\}^{\log^3 n}$ *such that* $(C_1, \ldots, C_n) = \mathsf{QSample}(1^n, \mathsf{seed})$*, it holds that*

$$\exists i \in [n]\colon \frac{1}{d} \cdot \sum_{z \in C_i} \mathbb{1}_{z \in H} \geq \frac{|H|}{n}.$$

▶ **Theorem 4.5.** *There is an implementation of* $\mathsf{QSample}$ *such that for every $n \in \mathbb{N}$,* $\mathsf{QSample}(1^n, \cdot)$ *results with a good family of quorums as defined in Definition 4.*

Proof. We define the $\mathsf{QSample}$ procedure as follows:[6]

$\mathsf{QSample}(1^n, \mathsf{seed})$, where $|\mathsf{seed}| = \log^3 n$:

1. Parse seed as $d \triangleq \log^2 n$ elements in $[n]$, denoted $C^* = P_1^*, \ldots, P_d^*$.
2. Define $C_i = P_1^* + i, \ldots P_d^* + i$ (all arithmetic is $\mathrm{mod}(n)$ and the nth party is identified by 0).
3. Output $Q = (C_1, \ldots, C_n)$.

The efficiency property of this construction is immediate by description. We argue that we get balanced committees for every seed (i.e., with probability 1). Fix party $j \in [n]$ and $\mathsf{seed} \in \{0,1\}^{\log^3 n}$. Denote $Q = (C_1, \ldots, C_n) = \mathsf{QSample}(1^n, \mathsf{seed})$ and let C^* be as above. It holds that

$$\begin{aligned}\sum_{i=1}^{n} \sum_{y \in C_i} \mathbb{1}_{j=y} &= \sum_{i=1}^{n} \sum_{z \in C^*} \mathbb{1}_{j=z+i \bmod n} \\ &= \sum_{z \in C^*} \sum_{i=1}^{n} \mathbb{1}_{j=z+i \bmod n} = \sum_{z \in C^*} 1 = |C^*| = d.\end{aligned}$$

The fact that the resulting committees are good (with high probability) follows by an application of Chernoff's bound along with a union bound. Specifically, for each $i \in [n]$, by Chernoff's bound,

$$\begin{aligned}\Pr_{\substack{\mathsf{seed} \leftarrow \{0,1\}^{\tilde{O}(1)},\\ (C_1,\ldots,C_n)=\mathsf{QSample}(1^n,\mathsf{seed})}} \left[\frac{1}{d} \cdot \sum_{z \in C_i} \mathbb{1}_{z \in H} \geq \left(1 - \frac{1}{\log\log n}\right) \cdot \frac{|H|}{n} \right] &\geq 1 - n^{-\frac{\log n \cdot |H|}{2n \cdot \log\log^2 n}} \\ &\geq 1 - \mathsf{negl}(n),\end{aligned}$$

where the last inequality follows since $|H| > n/\log\log n$. The second property of a good quorum now follows by union bound over all i's.

For the no terrible quorum property, observe that each party appear in exactly d committees and therefore the total number of honest parties in all quorums together is $d \cdot |H|$. By the pigeonhole principle, there must be one C_i with $(d \cdot |H|)/n$ parties from H. ◀

4.3 Everywhere Agreement on a Quorum

In this section we explain the final protocol, resulting with a good quorum known to all honest parties. At a very high level, we utilize the almost everywhere protocol from Section 4.1 to get a seed that is known to almost everyone. Then, we let each party expand its candidate

[6] As mentioned, the idea to define the quorums via fixed shifts of a single "sufficiently random" committee is due to [16]. However, there the randomness condition on the single committee is weaker and also they did not consider the "no terrible quorum" property.

seed to a quorum (that is agreed upon by almost everyone) using Theorem 4.5. We then use the multi-hop BARG to aggregate signatures along a tree of the quorum committees. Once the committees get an aggregated signature of all other committees they can pass on the candidate string to their associated parties.

Building blocks. Our protocol utilizes the following primitives:

1. A signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, as in Section 3.2.1.
2. A multi-hop BARG for NP $\mathsf{BARG} = (\mathsf{Gen}, \mathcal{P}, \mathsf{AggProve}, \mathcal{V})$, as in Definition 2. The NP relation that we use is defined next.

Committee signature relation. Recall that d is the size of a committee in the quorum. The NP relation we utilize in our BARGs is defined next.

▶ **Definition 6** (NP relation for BARGs). *We define the following NP-relation:*

$$R_{\mathcal{S},f,\vec{\mathsf{vk}},\mathsf{seed}}(i, \{\sigma_j\}_{j\in[d]}) = \begin{cases} 1 & i \in [n] \wedge \sum_{j\in[d]} \mathcal{S}.\mathsf{Verify}(\mathsf{vk}_{f(i,\mathsf{seed},j)}, \mathsf{seed}, \sigma_j) > d/2, \\ 0 & \textit{otherwise}, \end{cases}$$

where $f\colon [n] \times \{0,1\}^* \times [d] \to [n]$ is a fixed polynomial-time computable function, $\vec{\mathsf{vk}}$ is a vector of fixed public verification keys, and seed *is a fixed seed string.*

Above, we view $f(i, \mathsf{seed}, \star)$ as a function that outputs a multi-set of d parties (i.e., a committee). So, the above relation classifies a valid witness for index i as one containing a majority of valid signatures on seed from the multi-set members.

The QuorumAgree protocol. We now present the QuorumAgree protocol. Let T be an abstract binary tree with n nodes. Each of the nodes is virtually associated with a committee (in some arbitrary but fixed manner). Let $Depth(T)$ be T's depth, i.e., the number of nodes in the longest path from the root to the leaves (so roughly $Depth(T) = \log n$). For a node j in the tree, let $Parent(j)$ be its parent, $Left(j)$ and $Right(j)$ be its left and right children, respectively.

Setup phase:

1. A trusted dealer samples $i^* \leftarrow [n]$ uniformly at random and runs $\mathsf{BARG.Gen}(1^\lambda, i^*)$ to obtain crs and td. The dealer then publishes crs.
2. For each $i \in [n]$, P_i runs $\mathcal{S}.\mathsf{Gen}(1^\lambda)$ to obtain vk_i and sk_i. P_i publishes vk_i.

Online phase:

3. All parties run the almost everywhere protocol from Section 4.1. At the end, each party P_i holds seed_i where $|\mathsf{seed}_i| = \log^3 n$. Denote seed the string that is held by $1 - o(1)$ fraction of honest parties.
4. Each party (locally) computes its view of the quorum $C_1^{(i)}, \ldots, C_n^{(i)} = \mathsf{QSample}(1^n, \mathsf{seed}_i)$. Let $C_1, \ldots, C_n = \mathsf{QSample}(1^n, \mathsf{seed})$. Define $f\colon [n] \times \{0,1\}^{\log^3 n} \times [\log^2 n] \to [n]$ such that $f(i, x, j)$ is the j's party in lexicographic order of the ith committee in $\mathsf{QSample}(1^n, x)$.

Terminology. In step 5 below, when we say that a committee does something, e.g., sends a message, computes a value, etc, it means that every party in the committee does this and labels its operation with its identity and its view of the committee index. Importantly, since every party holds its own candidate for seed, the protocol may be different for every party.

5. For each committee C_j (in parallel):
  a. For each committee member P_i in C_j:
    i. P_i computes $\sigma_i = \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_i, \mathsf{seed})$.
    ii. P_i sends σ_i to every party in C_j, and for all $k \in [\log^2 n]$ it learns $\sigma_{f(j,\mathsf{seed},k)}$ (or $\perp$).
  b. Compute $\pi_j = \mathsf{BARG}.\mathcal{P}(j, \{\sigma_{f(j,\mathsf{seed},k)}\}_{k\in[\log^2 n]})$.
  c. Set $\pi_L = \perp$, $\pi_R = \perp$.
  d. If j is a leaf of T, set $\mathsf{State} = (\mathsf{Send}, \pi_j)$, else set $\mathsf{State} = \mathsf{WaitToAggregate}$.
  e. Repeat the following $Depth(T) - 1$ many times:
    i. If $\mathsf{State} = (\mathsf{Send}, \pi)$:
      A. Send π to $C_{Parent(j)}$.
      B. Set $\mathsf{State} = \mathsf{WaitToDistribute}$.
    ii. If $\mathsf{State} = \mathsf{WaitToAggregate}$:
      A. For any "in" message from $C_{Left(j)}$, if $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}, ([n], \vec{\mathsf{vk}}, \mathsf{seed}), Left(j), \mathsf{in}) = 1$, then set $\pi_L = \mathsf{in}$.
      B. For any "in" message from $C_{Right(j)}$, if $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}, ([n], \vec{\mathsf{vk}}, \mathsf{seed}), Right(j), \mathsf{in}) = 1$, then set $\pi_R = \mathsf{in}$.
      C. If $\pi_L \neq \perp$ and $\pi_R \neq \perp$, then set $\mathsf{State} = (\mathsf{Send}, \mathsf{BARG}.\mathsf{AggProve}(\mathsf{crs}, ([n], \vec{\mathsf{vk}}, \mathsf{seed}), j, \pi_j, \pi_L, \pi_R))$.

    * At this point, with high probability for each (honest) party $P_i \in C_{Root(T)}$ with $\mathsf{seed}_i = \mathsf{seed}$, $\mathsf{State} = (\mathsf{Send}, \pi)$, where π is a valid proof for the whole tree.
  f. Repeat the following $Depth(T)$ many times:
    i. If $\mathsf{State} = (\mathsf{Send}, \pi)$:
      A. Send π to $C_{Left(j)}$ and $C_{Right(j)}$ (if they exists).
      B. Set $\mathsf{State} = (\mathsf{Done}, \pi)$.
    ii. If $\mathsf{State} = \mathsf{WaitToDistribute}$:
      A. For any "in" message from $C_{Parent(j)}$, if $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}, ([n], \vec{\mathsf{vk}}, \mathsf{seed}), Root(T), \mathsf{in}) = 1$, set $\mathsf{State} = (\mathsf{Send}, \mathsf{in})$.
  g. If $\mathsf{State} = (\mathsf{Done}, \pi)$, send (seed, π) to party P_j.

6. For each party P_j:
  a. Set $\mathsf{out} = \perp$.
  b. For any message (seed, π), if $\mathsf{BARG}.\mathcal{V}(\mathsf{crs}, ([n], \vec{\mathsf{vk}}, \mathsf{seed}), Root(T), \pi) = 1$, set $\mathsf{out} = \mathsf{seed}$.
  c. Output out.

4.3.1 Efficiency

▶ **Lemma 7.** *The* QuorumAgree *protocol from Section 4.3 terminate after $\tilde{O}(1)$ rounds and each party sends $\tilde{O}_\lambda(1)$ bits overall during the protocol.*

Proof. By Theorem 4.3 it is guaranteed that all parties reached the end of step 3 after $\tilde{O}(1)$ and until then where each party sent $\tilde{O}(1)$ bits. We next analyze the following steps.

Rounds complexity. step 5a takes a single round, steps 5e and 5f consist of $O(Depth(T)) \subset \tilde{O}(1)$ iterations, where each iteration takes a single round. Then, there is additional round to send the aggregated proof to everyone. So, at total there are $\tilde{O}(1)$ rounds, as needed.

Communication complexity. consider a single committee. At step 5a each party sends $|\sigma| \in \tilde{O}_\lambda(1)$ bits to $d \in \tilde{O}(1)$ parties. By Theorem 3.3, during steps 5e and 5f each party sends $\tilde{O}_\lambda(1)$ bits in each step, and $O(Depth(T)) \subset \tilde{O}(1)$. At the last round, each party sends $\tilde{O}_\lambda(1)$ bits. Finally, since each party is part of exactly d committees and $d \in \tilde{O}(1)$, it follows that during the entire protocol each party sends at most $\tilde{O}_\lambda(1)$ bits, as needed. ◀

4.3.2 Security

This section provides the proof of agreement for the above protocol. We show that even if an adversary controls $1/3 - \epsilon$ fraction of parties and behaves arbitrarily in their name, the protocol still results with each honest party agreeing on seed.

We start by stating the result of our transformation from almost everywhere agreement to everywhere agreement. Note that the transformation and the almost everywhere protocol tolerate a different number of corrupted parties ($1/2 - \epsilon$ fraction vs. $1/3 - \epsilon$ fraction, respectively). Thus, the combined protocol will be secure against $1/3 - \epsilon$ fraction of parties. (Still, we decide to state a stronger result below as it might be useful for future works.)

▶ **Theorem 4.8** (From almost everywhere to everywhere agreement). *For any* PPT *adversary $\mathcal{A}$ that statically corrupts up to $1/2 - \epsilon$ fraction of parties, where ϵ is any positive constant, if at the end of step 3 of* QuorumAgree *at least $1 - O(\log^{-1} n)$ fraction of (honest) parties hold the same uniformly sampled value, denoted by* seed*, then with probability at least* $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$*, at the end of the protocol all (honest) parties outputs* seed.

Obviously, Theorem 4.1 follows from Theorem 4.8 together with the almost everywhere agreement protocol from Theorem 4.3. The rest of this section is devoted to the proof of Theorem 4.8.

At a high level, the proof of Theorem 4.8 consists of two main claims: (1) that the adversary can't prevent the honest parties from generating a valid proof for seed, and (2) that the adversary cannot generate valid proof for a maliciously chosen seed seed*. Consider a PPT adversary $\mathcal{A}$ that statically corrupt up to $1/2 - \epsilon$ fraction of parties, where $\epsilon > 0$. Denote by $\mathcal{C} \subseteq [n]$ the set of corrupted parties. Additionally, we assume (as written in the statement) that at the end of Step 3, at least $1 - O(\log^{-1} n)$ fraction of (honest) parties hold the same uniformly sampled value, denoted seed. Let $\mathcal{H} = \{P_i \mid i \in ([n] \setminus \mathcal{C}) \wedge \mathsf{seed}_i = \mathsf{seed}\}$, i.e., $\mathcal{H}$ is the set of honest parties that hold the "right" seed.

The first claim (Claim 10 below) we prove says that if an honest party is part of committee j, then it must hold a valid witness for index j, that is, it knows a w such that $R(j, w) = 1$. We utilize the following claim that says that every committee contains majority of honest parties that hold the right seed. The proof of this claim follows by the construction of the almost everywhere protocol that results with a uniform string which is then fed into our quorum construction.

▷ Claim 9. With probability at least $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$, for all $j \in [n]$, $|\mathcal{H} \cap C_j| > |C_j|/2$.

Proof. Recall that with probability at least $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$, seed, the output of the almost everywhere agreement protocol, is sampled uniformly at random and we use it as a seed to generate a quorum (using Theorem 4.5 with $H = \mathcal{H}$). Thus, with probability at least $(1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda))(1 - \mathsf{negl}(n))$, for each $j \in [n]$,

$$|\mathcal{H} \cap C_j| = \sum_{z \in C_j} \mathbb{1}_{z \in \mathcal{H}} \geq |C_j| \cdot \left(1 - \frac{1}{\log \log n}\right) \cdot \frac{|\mathcal{H}|}{n} \geq$$
$$|C_j| \cdot \frac{(1/2 + \epsilon) \cdot (1 - o(1)) \cdot n}{n} > |C_j|/2.$$ ◁

▷ Claim 10. At the end of Step 5a, with probability at least $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$, for every $i, j \in [n]$, if $P_i \in \mathcal{H} \cap C_j$, then party P_i holds $\{\sigma_{f(j,\mathsf{seed},k)}\}_{k\in[\log^2 n]}$ such that $R(j, \{\sigma_{f(j,\mathsf{seed},k)}\}_{k\in[\log^2 n]}) = 1$ (see Definition 6).

Proof. By description of the protocol, if $P_i \in \mathcal{H} \cap C_j$, then the size of the set

$$\{k \in [\log^2 n] \mid \mathsf{Verify}(\mathsf{vk}_{f(j,\mathsf{seed},k)}, \mathsf{seed}, \sigma_{f(j,\mathsf{seed},k)}) = 1\}$$

is at least $|\mathcal{H} \cap C_j|$. Claim 9 implies that $|\mathcal{H} \cap C_j| > |C_j|/2$ and so party P_i holds a valid witness. ◁

From this point on, it is convenient to assume that for all $j \in [n]$, $|C_j \cap \mathcal{H}| > |C_j|/2$ (which holds w.h.p due to Claim 9). In the next claim, we show that the honest parties aggregate the proof successfully. For simplicity of notation, let $X_{\mathsf{seed}} = ([n], \vec{\mathsf{vk}}, \mathsf{seed})$.

▷ Claim 11. Let T_j be the subtree where j is its root. After ℓ iterations of step 5e, for each $j \in [n]$:
1. If $Depth(T_j) = \ell + 1$, then $\mathsf{State} = (\mathsf{Send}, \pi)$ and $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, j, \pi) = 1$.
2. If $Depth(T_j) < \ell + 1$, then $\mathsf{State} = \mathsf{WaitToDistribute}$.
3. If $Depth(T_j) > \ell + 1$, then $\mathsf{State} = \mathsf{WaitToAggregate}$.

Proof. We prove by induction on ℓ, the number of executed iterations of step 5e.
Base case: For $\ell = 0$ and for all $j \in [n]$, j is a leaf of T. Then,
1. If $Depth(T_j) = 1$: By step 5d, $\mathsf{State} = (\mathsf{Send}, \pi)$, where $\pi = \mathcal{P}(j, \{\sigma_{f(j,\mathsf{seed},k)}\}_{k\in[\log^2 n]})$. By the correctness property of the BARG (Definition 2) and by Claim 10, it follows that $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, j, \pi) = 1$.
2. It is impossible that $Depth(T_j) < 1$, since for every j, $Depth(T_j) \geq 1$.
3. If $Depth(T_j) > 1$: By step 5d, $\mathsf{State} = \mathsf{WaitToAggregate}$.

Inductive step: Assume that the claim is true for $0, \ldots, \ell - 1$, and we prove it for ℓ. For $j \in [n]$:
1. If $Depth(T_j) = \ell + 1$, then before the start of the ℓth iteration of step 5e, $Depth(T_j) > (\ell - 1) + 1$, so by the inductive assumption, $\mathsf{State} = \mathsf{WaitToAggregate}$. W.l.o.g. assume that the left child sub-tree depth is ℓ and the right child sub-tree depth is $\ell' \leq \ell$. Due to the inductive assumption, at the ℓth (resp. ℓ'th) iteration of step 5e, the state of $Left(j)$ (resp. $Right(j)$) was (Send, π), where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Left(j)$ (resp. $Right(j)), \pi) = 1$. So, from the protocol's description, $\mathsf{State} = (\mathsf{Send}, \pi = \mathsf{AggProve}(\mathsf{crs}, X_{\mathsf{seed}}, j, \pi_j, \pi_L, \pi_R))$, such that $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, j, \pi) = 1$.
2. If $Depth(T_j) < \ell + 1$ there are two cases. First, if $Depth(T_j) < \ell$, we can use the inductive assumption directly. Second, if $Depth(T_j) = \ell$, by the inductive assumption, before the ℓth iteration of step 5e, $\mathsf{State} = (\mathsf{Send}, \pi)$. By the protocol's description, in the ℓth iteration of step 5e, the state changes to $\mathsf{WaitToDistribute}$.
3. If $Depth(T_j) > \ell + 1$, by the inductive assumption, before the ℓth iteration of step 5e, $\mathsf{State} = \mathsf{WaitToAggregate}$. W.l.o.g. assume that the left child sub-tree depth is $Depth(T_j) - 1 \geq \ell + 1$. From the inductive assumption, it follows that the left child ends the ℓth iteration of step 5e in state $\mathsf{WaitToAggregate}$ or Send, so j has not yet updated π_L. ◁

A corollary of Claim 11 is that the state of all nodes at the end of the iterations of step 5e is as expected.

► **Corollary 12.** *After the $Depth(T) - 1$ iterations of step 5e, the state of $Root(T)$ is* (Send, π)*, where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$. Additionally, the state of all other nodes is* $\mathsf{WaitToDistribute}$.

Now, similarly, we prove that each committee will learn the right proof attesting to the validity of seed w.r.t T.

▷ Claim 13. Let T_j be the subtree with j at its root. After the ℓ iterations of step 5f, for each $j \in [n]$:

1. If $Depth(T_j) = Depth(T) - \ell$, then $\mathsf{State} = (\mathsf{Send}, \pi)$, where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$.
2. If $Depth(T_j) < Depth(T) - \ell$, then $\mathsf{State} = \mathsf{WaitToDistribute}$.
3. If $Depth(T_j) > Depth(T) - \ell$, then $\mathsf{State} = (\mathsf{Done}, \pi)$, where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$.

Proof. By induction on ℓ:

Base case: For $\ell = 0$, and for all $j \in [n]$:

1. If $Depth(T_j) = Depth(T)$, then since j is the root of T, the claim follows from Corollary 12.
2. If $Depth(T_j) < Depth(T)$, then j is not the root of T and so the claim follows from Corollary 12.
3. It is impossible that $Depth(T_j) > Depth(T)$ since for every j, $Depth(T_j) \leq Depth(T)$.

Inductive step: Assume that the claim is true for $0, \ldots, \ell - 1$, and we will prove it for ℓ. For $j \in [n]$:

1. If $Depth(T_j) = Depth(T) - \ell$, by the inductive assumption, before the ℓth iteration of step 5f, the state of j is $\mathsf{WaitToDistribute}$, and the state of $Parent(j)$ is (Send, π), where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$. Since there is at least one honest party in each committee (actually, there is an honest majority) and by description of step 5f, at the end of the ℓth iteration of step 5f, the state of j will be (Send, π), as needed.
2. If $Depth(T_j) < Depth(T) - \ell$, from the inductive assumption, before the ℓth iteration of step 5f, the state of j and his parent is $\mathsf{WaitToDistribute}$. So, by description of step 5f, at the end of the ℓth iteration of step 5f, the state of j will be $\mathsf{WaitToDistribute}$, as needed.
3. If $Depth(T_j) > Depth(T) - \ell$ there are two cases. First, if $Depth(T_j) > Depth(T) - \ell + 1$, we use the inductive assumption directly. Second, if $Depth(T_j) = Depth(T) - \ell + 1$, by the inductive assumption, before the ℓth iteration of step 5f, $\mathsf{State} = (\mathsf{Send}, \pi)$, where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$. Thus, by the protocol's description, during the ℓth iteration of step 5f, the state changes to (Done, π). ◁

A corollary of Claim 13 is that the state of all nodes at the end of the iterations of step 5f is as expected.

▶ **Corollary 14.** *After the $Depth(T)$ iterations of step 5f the state of all nodes $j \in [n]$ is (Done, π). Additionally, $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$.*

Next, we conclude that with high probability, each honest party will receive *at least once* the right seed along with a valid proof.

▶ **Lemma 15.** *With probability at least $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$, at step 6b, every (honest) party sets $\mathsf{out} = \mathsf{seed}$, where seed is the value that is outputted by the almost-everywhere agreement sub-protocol.*

Proof. By Corollary 14, each committee's state is (Done, π), where $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}}, Root(T), \pi) = 1$. Together with Claim 9, every party receives at least once the right seed and a valid proof. Thus, after step 6b, each party will update out with seed. ◀

The above lemma stands for the protocol's "correctness". In other words, we proved that the adversary cannot prevent from any honest party from receiving the right seed along with a valid proof for it. However, we have not yet proved that the adversary cannot generate another valid proof for a wrong seed, which is also undesirable. For simplicity, assume that the adversary is given free of charge full control over unknowledgeable honest parties, i.e., parties with $\mathsf{seed}_i \neq \mathsf{seed}$ (after step 6). Notice that this group is of size $o(n)$, so this does not add significant power to the attacker.

We show that if there exists an adversary that generates a valid proof for a maliciously chosen seed, then we can build another adversary that wins the unforgeability game of the signature scheme (Section 3.2.1) with only slightly lower probability. Our adversary uses the BARG scheme knowledge extractor from Definition 2 which, to recall, is an efficient procedure that receives valid proofs for some seed' and extracts (w.h.p.) the witness associated with some committee C_{i^*}, where the index i^* is supplied during setup.

▷ Claim 16. Assume a PPT adversary $\mathcal{A}'$ that at step 6b of the protocol can generate a pair (seed', π') where $\mathsf{seed}' \neq \mathsf{seed}$ and $\Pr[\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}'}, Root(T), \pi') = 1] \geq p(n)$. Then, there is another PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{ExUnfg}_{\mathcal{S},\mathcal{A}}(\lambda) = 1] \geq p(n)/2n$, where the game $\mathsf{ExUnfg}_{\mathcal{S},\mathcal{A}}(\lambda)$ is defined in Section 3.2.1.

Proof. Let $\mathcal{E}$ be the knowledge extractor of the BARG scheme (Definition 2). We build $\mathcal{A}$ as follows:

1. $\mathcal{A}$ chooses $k = n$, and receives $\{\mathsf{vk}_i\}_{i\in[n]}$ from the experiment.
2. $\mathcal{A}$ samples $i^* \leftarrow [n]$ and runs $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{BARG.Gen}(1^\lambda, i^*)$.
3. $\mathcal{A}$ sends crs and $\{\mathsf{vk}_i\}_{i\in[n]}$ to $\mathcal{A}'$.
4. $\mathcal{A}'$ chooses the set of corrupted parties $\mathcal{C}$ and sends $\mathcal{C}$ to $\mathcal{A}$. $\mathcal{A}$ sets $\mathcal{C}$ as its corrupted set, receives $\{\mathsf{sk}_i\}_{i\in\mathcal{C}}$ from the experiment, and forwards it to $\mathcal{A}'$.
5. $\mathcal{A}$ and $\mathcal{A}'$ run together the QuorumAgree protocol up to the end of step 6b, where $\mathcal{A}$ simulates $[n] \setminus \mathcal{C}$. When it needs to sign a message m, the experiment signs it for him. At the end, $\mathcal{A}'$ sends (seed', π') to $\mathcal{A}$.
6. $\mathcal{A}$ sends $(X_{\mathsf{seed}'}, \mathsf{td}, i^*, Root(T), \pi')$ to $\mathcal{E}$, and it answers with $\{\sigma_{f(i^*,\mathsf{seed}',j)}\}_{j\in[\log^2 n]}$.
7. For each $j^* \in \{f(i^*, \mathsf{seed}', j)\}_{j\in[\log^2 n]}$, if $j^* \in [n] \setminus \mathcal{C}$ and $\mathsf{Verify}(\mathsf{vk}_{j^*}, \mathsf{seed}', \sigma_{j^*}) = 1$, then $\mathcal{A}$ outputs $(j^*, \mathsf{seed}', \sigma_{j^*})$.
8. $\mathcal{A}$ outputs $(\bot, \bot, \bot)$.

To analyze the above reduction, we define the following events:

- Event $E_{\mathsf{AcceptProof}}$: $\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}'}, Root(T), \pi') = 1$.
- Event $E_{\mathsf{EnoughValidSigs}}$: $R_{\mathcal{S},f,\vec{\mathsf{vk}},\mathsf{seed}'}(i^*, \{\sigma_{f(i^*,\mathsf{seed}',j)}\}_{j\in[\log^2 n]}) = 1$. (See Definition 6.)
- Event $E^{(i)}_{\mathsf{HonestMajority}}$: $|C_i \cap ([n] \setminus \mathcal{C})| > \log^2 n/2$, where C_i the ith committee in the quorum $\mathsf{QSample}(n, \mathsf{seed}')$.
- Event E_1: The experiment outputs 1.

First, recall that every committee size is exactly $\log^2 n$, independently of seed'. By the definition of $R_{\mathcal{S},f,\vec{\mathsf{vk}},\mathsf{seed}'}$ if $E_{\mathsf{AcceptProof}}$ happens, then

$$\sum_{j\in[\log^2 n]} \mathsf{Verify}(\mathsf{vk}_{f(i^*,\mathsf{seed}',j)}, \mathsf{seed}, \sigma_{f(i^*,\mathsf{seed}',j)}) > \log^2 n/2.$$

This means that $\Pr[E_{\mathsf{EnoughValidSigs}} \wedge E^{(i^*)}_{\mathsf{HonestMajority}} \wedge \neg E_1] = 0$. From the "no terrible quorum" property (see Definition 4), we know that $\Pr[\exists i \in [n] \colon E^{(i)}_{\mathsf{HonestMajority}}] = 1$. Since $\mathcal{A}'$ is unaware of i^*, $\Pr[E^{(i^*)}_{\mathsf{HonestMajority}}] \geq 1/n$, and also $E^{(i^*)}_{\mathsf{HonestMajority}}$ and $E_{\mathsf{AcceptProof}}$ are independent events.[7] Next, from the claim requirement it follows that $\Pr[E_{\mathsf{AcceptProof}}] \geq p(n)$, and from the "somewhere argument of knowledge" property of the BARG scheme we conclude that

$$\Pr[E_{\mathsf{EnoughValidSigs}} \mid E_{\mathsf{AcceptProof}} \wedge E^{(i^*)}_{\mathsf{HonestMajority}}] \geq 1/2.$$

Finally,

$$\Pr[E_1] \geq \Pr\left[E_{\mathsf{EnoughValidSigs}} \wedge E^{(i^*)}_{\mathsf{HonestMajority}}\right] \geq$$
$$\Pr\left[E_{\mathsf{AcceptProof}} \wedge E_{\mathsf{EnoughValidSigs}} \wedge E^{(i^*)}_{\mathsf{HonestMajority}}\right] =$$
$$\Pr\left[E^{(i^*)}_{\mathsf{HonestMajority}} \wedge E_{\mathsf{AcceptProof}}\right] \cdot \Pr\left[E_{\mathsf{EnoughValidSigs}} \mid E^{(i^*)}_{\mathsf{HonestMajority}} \wedge E_{\mathsf{AcceptProof}}\right] =$$
$$\Pr\left[E_{\mathsf{AcceptProof}}\right] \cdot \Pr\left[E^{(i^*)}_{\mathsf{HonestMajority}}\right] \cdot \Pr\left[E_{\mathsf{EnoughValidSigs}} \mid E^{(i^*)}_{\mathsf{HonestMajority}} \wedge E_{\mathsf{AcceptProof}}\right] \geq$$
$$p(n)/2n. \qquad \lhd$$

Next, we show how to use the above reduction to prove that the adversary cannot generate a proof for a wrong seed.

▶ **Lemma 17.** *With probability at least* $1 - \mathsf{negl}(n) - \mathsf{negl}(\lambda)$*, at step 6b, if (honest) party sets* $\mathsf{out} = \mathsf{seed}$*, then* seed *is the value that defines* $\mathcal{H}$ *(the almost everywhere agreed value).*

Proof. Assume, by contradiction, that the claim is false. That is, with noticeable probability an honest party sets $\mathsf{out} = \mathsf{seed}'$ for some $\mathsf{seed}' \neq \mathsf{seed}$. Then, by Corollary 14, there exists a PPT adversary $\mathcal{A}$ that can generate a pair (seed', π') such that $\mathsf{seed}' \neq \mathsf{seed}$ and for some noticeable function $f(n)$, $\Pr[\mathcal{V}(\mathsf{crs}, X_{\mathsf{seed}'}, \mathit{Root}(T), \pi') = 1] \geq f(n)$. By Claim 16, there is another PPT adversary $\mathcal{A}'$ that win the PKI unforgeability game of the signature scheme (Section 3.2.1) with noticeable probability. This is a contradiction. ◀

Finally, Theorem 4.8 follows by combining Lemmas 15 and 17.

References

1 Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, pages 317–326, 2019.

2 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–10, 1988.

3 Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. *Computer Science Research*, pages 313–322, 1992.

4 Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $O(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 319–330, 2021.

5 Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 57–64, 2013.

6 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation, (OSDI)*, pages 173–186, 1999.

[7] Recall that for independent events E and E', $\Pr[E \wedge E'] = \Pr[E] \cdot \Pr[E']$.

7 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In *Advances in Cryptology – CRYPTO*, volume 293, page 462, 1987.

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

9 Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *Advances in Cryptology – CRYPTO*, pages 394–423, 2021.

10 Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 68–79, 2021.

11 Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-np and applications. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1057–1068. IEEE, 2022.

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

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

14 Rachit Garg, Kristin Sheridan, Brent Waters, and David J. Wu. Fully succinct batch arguments for NP from indistinguishability obfuscation. In *Theory of Cryptography – TCC*, pages 526–555, 2022.

15 Yuval Gelles and Ilan Komargodski. Brief announcement: Scalable agreement protocols with optimal optimistic efficiency. In *DISC*, pages 42:1–42:6, 2023.

16 Yuval Gelles and Ilan Komargodski. Optimal load-balanced scalable distributed agreement. *IACR Cryptol. ePrint Arch.*, page 1139, 2023.

17 Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 99–108, 2011.

18 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, SOSP*, pages 51–68. ACM, 2017.

19 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC*, pages 218–229, 1987.

20 Dan Holtby, Bruce M. Kapron, and Valerie King. Lower bound for scalable byzantine agreement. *Distributed Comput.*, 21(4):239–248, 2008.

21 Pavel Hubácek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *ITCS*, pages 163–172. ACM, 2015.

22 James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. SNARGs for P from sub-exponential DDH and QR. In *Advances in Cryptology – EUROCRYPT*, pages 520–549, 2022.

23 Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 708–721, 2021.

24 Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Boosting batch arguments and RAM delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC*, pages 1545–1552, 2023.

25 Yael Tauman Kalai, Omer Paneth, and Lisa Yang. Delegation with updatable unambiguous proofs and PPAD-hardness. In *Advances in Cryptology – CRYPTO*, pages 652–673, 2020.

26 Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. In *Distributed Computing and Networking – ICDCN*, pages 203–214, 2011.

27 Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with õ($n^{3/2}$) bits. In *Distributed Computing, 23rd International Symposium, DISC*, pages 464–478, 2009.

28 Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 990–999, 2006.

29 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

30 Moni Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology – CRYPTO*, pages 96–109, 2003.

31 Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing, STOC*, pages 33–43, 1989.

32 Omer Paneth and Rafael Pass. Incrementally verifiable computation via rate-1 batch arguments. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1045–1056, 2022.

33 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

34 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, STOC*, pages 73–85, 1989.

35 John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing, STOC*, pages 387–394, 1990.

36 Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *Advances in Cryptology – CRYPTO*, pages 433–463, 2022.