Sublinear-Round Broadcast without Trusted Setup

Andreea B. Alexandru*[1], Julian Loss†[2], Charalampos Papamanthou‡[3], Giorgos Tsimos§[4], and Benedikt Wagner¶[5]

[1]Duality Technologies
[2]CISPA Helmholtz Center for Information Security
[3]Yale University
[4]University of Maryland
[5]Ethereum Foundation

Abstract

Byzantine broadcast is one of the fundamental problems in distributed computing. Many of its practical applications, from multiparty computation to consensus mechanisms for blockchains, require increasingly weaker trust assumptions, as well as scalability for an ever-growing number of users n. This rules out existing solutions which run in a linear number of rounds in n or rely on trusted setup requirements. In this paper, we propose the first *sublinear-round* and *trustless* Byzantine broadcast protocol for the dishonest majority setting. Unlike previous sublinear-round protocols, our protocol assumes neither the existence of a trusted dealer who honestly issues keys and correlated random strings to the parties nor random oracles. Instead, we present a solution whose setup is limited to an unstructured uniform reference string and a plain public key infrastructure (a.k.a. bulletin-board PKI).

Our broadcast protocol builds on top of a *moderated gradecast* protocol which parties can use to reach weak agreement on shared random strings. Using these strings, we can then run in an unbiased fashion a committee-based Byzantine protocol, similar to that of Chan et al. (PKC 2020), which terminates in a sublinear number of rounds. To this end, we propose a novel construction for committee election, which does not rely either on random oracles or on a trusted dealer, and uses NIZKs and time-lock puzzles. Our protocol is resilient against an adaptive adversary who corrupts any constant fraction of parties.

*`aalexandru@dualitytech.com`

†`loss@cispa.de`

‡`charalampos.papamanthou@yale.edu`

§`tsimos@umd.edu`; Work partially done while the author was visiting at Yale.

¶`benedikt.wagner@ethereum.org`; Work done while the author was at CISPA.

Contents

1 Introduction

In the problem of Byzantine broadcast, a sender distributes its input v to n parties. A protocol for broadcast is deemed secure if it satisfies two properties in the presence of any $t < n$ corruptions: (1) *consistency*: all honest parties output the same value, and (2) *validity*: if the sender honestly follows the protocol, then all honest parties output v. Broadcast is essential in ensuring a consistent view between parties, and has important applications in multiparty computation, verifiable secret sharing and state machine replication. To reduce the overhead of such applications, a long line of works has focused on minimizing the *round-efficiency* of broadcast, i.e., how many synchronous rounds the protocol runs for. By the famous work of Dolev and Strong [14], having $t+1$ synchronous rounds is both necessary and sufficient to achieve broadcast *deterministically*. This severely limits the practicality of such protocols as n grows large.

Fortunately, randomized protocols are known to bypass this lower bound. Thus, an active area of research has studied robust and efficient randomized broadcast protocols for the most challenging setting with a *dishonest majority* of $n/2 < t < n$ corrupted parties. In this corruption regime, the lower bound of Lamport, Shostak, and Pease [33, 29] asserts that some form of cryptographic setup is necessary in order to achieve broadcast. With respect to such setup assumptions, existing round-efficient protocols fall into one of two unsatisfactory categories.

Protocols of the first category [10, 40, 41] achieve $o(n)$ rounds for any *constant fraction* of corrupted parties. However, they rely on a trusted dealer who securely generates and distributes cryptographic parameters during a setup phase (e.g., signing key pairs). Unfortunately, assuming a trusted dealer is unacceptable for high-stake scenarios as it lies at odds with the primary goal of distributed systems: to eliminate single points of failure.

Protocols in the second category work in the *plain public key model.* Rather than relying on a trusted dealer, this model allows parties to generate their own secret and public keys. In this manner, trusted setup is minimized to a *public bulletin-board* to which parties can post their public keys and correctly read the other parties' public keys before commencing protocol execution. The only known protocols in this category are due to Garay *et al.* [21] and Fitzi and Nielsen [18]. However, both of these protocols achieve security only for a very small margin where the number t of corrupted parties must satisfy $t - n/2 \leq o(n)$.

1.1 Our Contribution

In this work, we significantly improve over the state of the art. Concretely, we design the first sublinear round broadcast protocol secure against an adaptive adversary controlling a majority of $t \leq (1-\epsilon)n$ of the parties, for constant $\epsilon \in (0,1)$, with setup assumptions of only plain public key model and a uniform common reference string. Our technical approach can be summarized as follows.

- **Committee election without trusted setup.** We follow the approach of [10] of electing a small unpredictable committee which is responsible for voting on the sender's bit. We devise algorithms for a committee election scheme based on non-interactive proof systems and time-lock puzzles, but without trusted setup, which allows parties to verify the membership of both themselves and of other parties. We guarantee that at the time of election, this committee contains at least one honest party and cannot be larger than $O(\lambda)$ members, for a statistical security parameter λ. This initial scheme describes algorithms, not protocols, and thus does not deal with the consistency of views between honest parties, a limitation that we must overcome in the next step toward our final goal.

- **Verifiable graded committee election.** We augment our committee election scheme with a flavor of *graded broadcast* channels [17, 21]. This, in turn, yields a graded committee election with the same properties as above that has a round complexity sublinear in the number of parties n. The committee election is graded in the sense that the confidence of the honest parties over the members of the committee is quantified by a grade they assign to each member. Our grading ensures that honest parties always assign honest parties the highest possible grades, whereas grades for dishonest parties may differ by at most one among honest parties.
- **Sublinear-round Broadcast.** Building on top of our graded committee election, we provide a broadcast protocol with $O(\lambda) = o(n)$ round complexity. Our protocol compares favorably to the Dolev-Strong protocol [14], as it reduces the round complexity from $O(n)$ without introducing trusted dealers.

Our solution relies on time-lock puzzles, but we believe this to be a minor restriction. Indeed, many works have considered setup-free protocols under resource-restricted cryptography assumptions, for instance, Andrychowicz and Dziembowski [3] and Garay *et al.* [22, 23].

1.2 Technical Overview

We now provide a more detailed overview of our techniques.

Chan *et al.*'s Protocol. Our starting point is the protocol of Chan *et al.* [10] which achieves sublinear round complexity against $(1-\epsilon)n$ adaptive corruptions, for constant $\epsilon > 0$, relying on a trusted dealer. At a high level, it delegates the message signing steps in the Dolev-Strong [14] protocol to an ad-hoc committee of roughly $O(\lambda)$ parties.[1] If at least one party in the committee is honest, their protocol achieves $O(\lambda)$ round complexity. For committee election, a *verifiable random function* (VRF) is used, which allows a party P to evaluate a pseudorandom function F on a point v privately, as $y := F(\mathsf{sk}, v)$, where sk is P's private key. The VRF also produces a proof π that can be used to verify y against P's public key pk. Thus, each party can be independently elected a committee member with λ/n probability if its VRF output y evaluated on, e.g., a fixed identifier id, falls below a certain threshold τ, i.e., $F(\mathsf{sk}, \mathsf{id}) < \tau$.

In [10], a trusted dealer generates the key pairs on behalf of all parties and securely distributes them before the protocol execution. The approach as sketched above fails if parties generate their VRF keys and register them to a public bulletin-board. Namely, a malicious party P can choose its key to minimize its output when evaluated on the fixed identifier, thus becoming a committee member with high probability. As such, an adversary controlling any constant fraction of the parties can degrade Chan *et al.*'s protocol to $O(n)$ rounds by electing all of them to the committee (as consistency would be broken with fewer rounds).

Challenges of Removing the Trusted Dealer. One way of addressing the above issue is for parties to agree on an unpredictable identifier id, but *after* registering their keys to the public bulletin-board. Intuitively, this should thwart any attempt of generating VRF keys in a way that minimizes the output when evaluated on id. For our purposes, this turns out to be rather simple in the random oracle model. Assuming VRFs which themselves act as random oracles, parties can set, for instance, $\mathsf{id} := H(\mathsf{pk}_1, ..., \mathsf{pk}_n)$ to obtain a suitably unbiased random string to be input to the VRF.[2] Here, H is a hash function modeled as a public and truly random function.

[1] To counter adaptive corruptions, there is one committee per bit that could be input by the sender.

[2] A proper subset of the authors have authored a previous version where they achieved similar results but on the random oracle model. The effort to remove the ROM assumption led the paper's structure to change, through several major changes and novel necessary intermediate results. While the previous version acted as a building block, an

Obviously, this approach fails in the plain model. As there seems to be no way of locally computing id without a random oracle, moving to the standard model introduces a circular issue: how can parties agree on an unpredictable string id if their ultimate goal is to agree on the sender's output? To add to the difficulty, the round complexity of the resulting broadcast protocol now has to account for the process of agreeing on id, in addition to other steps. Thus, agreement on id is only helpful if it can be achieved in a round-efficient manner.

As an additional challenge, we also need to replace the VRFs in the above approach by a plain-model election function which outputs uniform values when called on uniform inputs. This is especially challenging given that corrupted parties can select their keys *after* honest parties keys.

Unbiased Randomness from VSS: A Recap. To break the circularity described above, our approach follows the template of boosting a weak form of agreement—graded agreement—on an unpredictable string (for committee election) to a strong form of agreement on an arbitrary sender input (see [16, 25, 3, 23]). Compared to prior work, however, the setting of dishonest majority with only a constant fraction of honest parties turns out to be a major challenge. In particular, this disqualifies typical approaches such as verifiable secret sharing (VSS) [16, 25], which requires an honest majority, or reducing the number of Byzantine parties to run an honest-majority broadcast protocol [21, 18], which requires either $t - n/2 \leq o(n)$ or trusted setup.

Nevertheless, toward illustrating our own approach, it is helpful to recall VSS-based protocols for the honest majority setting. To understand the difficulty of agreeing on an unbiased random string id, consider first a naive protocol where parties locally generate random seeds and share them. Then, parties combine them into an output. Setting aside for the moment the issue of reaching agreement on these outputs, there is an obvious problem with *bias*: dishonest parties could observe the honest parties' strings and accordingly adjust or simply withhold their own strings, thus biasing the outcome. If we were in an honest majority scenario, these issues could be overcome using, e.g., parallel VSS, which each instance of VSS commits a party P to a value in such a way that it can be forcibly revealed even if P does not publish it itself. To build intuition, we briefly recall how VSS works: the dealer sends a share of its secret to each other party, along with information for verification; if this information is malformed, honest parties distribute it to show the dealer is malicious. Then, when reconstructing the secret of the dealer, only the valid shares submitted by parties are used. Going back to our goal of distributed unpredictable string generation, each party would act as a VSS dealer in the first round of the protocol. Then, the aggregation of all the reconstructed secrets, happening in a second round (and based solely on the valid shares received in the first round), would serve as the common unpredictable string id. Importantly, VSS guarantees that the adversary is not capable of reconstructing the secrets of the honest parties before having to commit its own secret. Moreover, the adversary cannot obtain valid shares that would bias the result. Finally, honest majority ensures that the adversary's committed secrets are always reconstructed and are included in the aggregated value even without the adversary's participation. This prevents the adversary from biasing id by selectively aborting certain instances of VSS.

From VSS to Time-Lock Puzzles. Unfortunately, VSS with guaranteed output for honest parties does not exist in the dishonest majority setting. Thus, our idea is to try to mirror a weaker property of VSS-based approaches in a dishonest majority setting as follows. The adversary is allowed to reconstruct honest parties' secrets *eventually*, (which would not be allowed in VSS without a honest majority or the dealer's participation), but the adversary is still not capable to commit to a secret

even simpler solution can be built in the ROM. Therefore, the authors considered it more natural to build the current construction from scratch, especially since it involved new and distinct tools. Still, the authors consider this work to be a natural continuation of the past version, and thus do not cite it as a separate work. The past version (in the ROM) can be accessed at `https://eprint.iacr.org/2022/1383` that depends on the honest parties' secrets. Intuitively, this should prevent the adversary from biasing id, so long as honest parties agree on its final value before the adversary learns their secrets.

To this end, we use the idea of resource-restricted cryptography, where the restricted resource is sequential time. Concretely, we assume the existence of a *time-lock puzzle* (TLP). Critically, a TLP cannot be solved faster than within a certain predetermined time Δ, even when given access to parallel computation. This suggests the following approach: first, have parties locally sample random strings, one for each other party, time-lock encrypt these strings, and agree on the resulting TLPs instead of agreeing directly on the strings. Then, solve a batch of n puzzles (one from each party) and compute from its solution the final identifier id that will be used as an input to the election function. Intuitively, if Δ is larger than the time it takes to agree on the puzzle, it should be impossible for Byzantine parties to pick their contributions to the election function in a manner that biases id. Moreover, honest parties can forcibly open *any* of the corrupted parties' puzzles without further interaction using the batch solve algorithm. This resembles the step of forcible reconstruction in VSS-based approaches and eliminates any remaining avenues for the adversary to bias the distribution of id.

Converting this approach into a working solution with a formal security proof, however, turns out to be very challenging. While there are prior works [37, 35, 13] on random beacons using time-lock puzzles or verifiable delay functions, they either require a trusted setup and/or additional assumptions on the computational resources of Byzantine parties, an honest majority, broadcast, or do not run in $o(n)$ rounds.

As a result, we have to develop novel techniques in order to reach our objective, both for achieving randomness and for achieving agreement.

Towards Committee Elections in the Plain Model. To illustrate the main ideas behind our solution, let us assume broadcast channels, ignoring for now that we want to construct such a channel in the first place.[3] We also assume a predicate Pred that takes as input a value $\mathsf{id}_j \in \mathbb{Z}_p$ associated to a party P_j and outputs whether P_j is in the committee, such that Pred yields a secure committee election if all id_j's are uniform, independent, and cannot be biased. It turns out that such a predicate can easily be constructed and reduces our task to generating the values id_j.

In this scenario with broadcast channels, we can have parties broadcast special purpose one-time public keys at the beginning of each committee election. It is acceptable for parties to reveal their secret key to convince others they were elected. With this in mind, our first strawman approach is that each party P_j defines its public key as $X_j = g^{x_j}$ for a random secret key x_j and sets $\mathsf{id}_j := x_j$. Of course, such id_j is neither independent nor necessarily uniform for dishonest parties. To make them uniform, we take inspiration from distributed coin flips and set

$$\mathsf{id}_j := x_j + \sum_{i \in [n]} T_{i,j} \quad \text{for every party } P_j$$

where $T_{i,j} \in \mathbb{Z}_p$ is a tag that P_i prepares for party P_j. In this way, each party contributes to the other party's id_j. However, we still need to ensure that the tags themselves are independent. As insinuated above, we solve this by having each party P_i broadcast a TLP of its tags $T_{i,j}$. Then, a party P_j (batch) opens all received TLPs to determine their values id_j.

Proving Security of Our Committee Election. One of the main technical challenges that we overcome in this work is how to properly use the security of TLP to prove the security of our committee election. Concretely, we want to show two properties: (i) the adversary cannot corrupt all elected parties, and (ii) the committee does not grow too large.

[3] In the main body of the paper, we do not assume broadcast channels when building the solution; instead, we first describe algorithms that parties run locally for the committee election, and then connect them through protocols.

To prove (i), we can avoid using the security of TLPs in the first place by letting the public key of P_i be a perfectly hiding commitment for x_i: $X_i = g^{x_i} h^{r_i}$. We can now argue that even when the adversary corrupts t parties, with high probability one of the remaining $n - t$ parties is elected. Notably, this holds even if the adversary chooses *all* tags. Because commitments are perfectly hiding, it means that from the view of the adversary, all uncorrupted x_i are uniformly random, and so is id_i. We instantiate the predicate such that, with overwhelming probability, at least one among $n - t$ uniform values satisfies it.

Proving (ii) requires, in essence, to argue that the adversary cannot significantly increase its probability of winning. Intuitively, this should hold since if at least one honest tag $T_{i^*,j}$ contributes to a corrupted id_j, then value id_j is uniform and cannot be biased by the adversary. This intuition crucially exploits the fact that the adversary has to fix its keys and tags *before* learning $T_{i^*,j}$. Clearly, to prove this formally, we have to rely on the security of the TLP.

This turns out to be highly non-trivial. More concretely, we want to have a hybrid step in our proof which replaces the puzzle of $T_{i^*,j}$ with a puzzle of 0. The security definition of a TLP states that the adversary can take arbitrary polynomial time to come up with two messages m_0 and m_1, then gets a puzzle containing m_b for a random bit b, and finally has to find b in time bounded by a parameter Δ, which models the hardness of the puzzle. With this in mind, note that the reduction interpolating the subsequent hybrids (one with a puzzle of $T_{i^*,j}$ and one with a puzzle of 0) is the adversary in the TLP game. The reduction therefore gets the puzzle (of $T_{i^*,j}$ or of 0) and has to determine the bit b. To do so, it would have to evaluate the winning condition and output the final output of the game. This naive approach, however, is flawed: as the game runs much longer than Δ, the reduction would do so as well, meaning it is not a valid adversary in the TLP game. Indeed, this is a general issue when composing timed cryptography in larger protocols.

To address this, we make several changes to our election protocol in order to enable the reduction to evaluate the winning condition quickly. For that, the reduction needs to know the corrupted party's secret key x_j and all tags $T_{i,j}$. We establish this by adding proofs of knowledge to (i) each party's public key, and (ii) all time-lock puzzles,[4] such that the reduction can extract x_j and all $T_{i,j}$. Relying on the computational binding property of our public keys (which are commitments to x_j) and on the correctness of TLPs, we can argue that the extracted values are as good as the ones obtained by finishing the game. Notably, the full formal proof involves two subsequent hybrids that are not indistinguishable but for which we can show that the adversary's winning probability does not increase. Such distinguishable hybrids have been used before [40].

Removing the Broadcast Channels. Our sketch above on how to use a TLP as a "substitution" for VSS in the dishonest majority setting implicitly assumed that parties could broadcast their individual puzzles and keys X_i. As our goal is to construct a broadcast protocol, we need to find a means of replacing any broadcast with a suitable (weaker) primitive.

Following classical works in the area of round-efficient agreement, we consider a weaker consensus primitive commonly referred to as *gradecast* (GC). In GC, parties output a grade together with the output. Intuitively, the grade indicates a party's confidence in its output as being equal to the sender's value. Thus, if the sender is honest, every party should output the grade expressing the highest possible confidence. If the sender is dishonest, parties' grades can be lower and some parties might not learn the output at all. In this case, we would still like to ensure that (i) even low grades correspond to the same output across honest parties (if any), and (ii) grades of honest parties differ by at most one. The solution, however, is not as straightforward as simply replacing all broadcast channels with gradecasts. While for the distribution of their public keys parties can use gradecast, in order to prevent bias on the ids, we need all Byzantine parties to compute their id on puzzle

[4]This intuitively makes the puzzles non-malleable. Indeed, malleability would allow to bias the result.

solutions that include the contribution of at least one honest party. To this end, we propose to use a generalized notion of *moderated gradecast* (Mod-GC), introduced in [25].

In GC, the sender's behaviour fully determines the parties' confidence in the output. Mod-GC designates a different party M as a *moderator* responsible for relaying the GC output of the sender to all parties. Parties now output a grade that reflects their confidence in M rather than in the sender directly. While the honest grades' properties closely mirror those of GC, Mod-GC enables a more refined strategy against dishonest parties. The main idea is for each party M to moderate each other party P's GC instance. Parties then assign M the minimum grade over all the instances it moderated. This way, a party M who incorrectly forwards an honest party's contribution is punished with a lower grade. To ensure that honest parties do not punish each other for moderating GCs received initially with low grades, our grading scheme takes into account both the grades of the original senders P as well as the efforts of M in relaying all the GCs it moderated. The outcome is a scenario in which, for all parties M, the honest parties attain graded agreement on a set of TLPs associated with M. Now, they can derive from these puzzles (and M's claimed secret key x_M, validated using the graded public key X_M) a way to verify the identifier id_M attributed to M along with a grade g_M, regardless of the senders' behaviour in the GCs which M moderated. Unpredictability follows because a dishonest party M can only obtain an id_M that holds a positive grade in the view of any honest party if id_M was computed using the contributions of *all honest parties*. In addition, all honest parties will have high grades in the view of the other honest parties (the maximum grade or the maximum grade minus one). As mentioned above, soundness of proof systems is a requirement to obtain consistency between honest parties' views. Finally, the step where honest parties batch solve the puzzles to compute their own id resembles the step of reconstruction from VSS. Honest parties only use the puzzles with non-zero grades to compute id, which loosely corresponds to using only the valid shares in a VSS-based approach.

Wrapping up. Now that all parties have unpredictable ids and associated grades in the view of honest parties, we can use them in Chan *et al.*'s construction [10]. We set the maximum grade equal to the number of stages in their protocol. There, if a party receives r signatures on a message $b \in \{0, 1\}$ in Stage r (consisting of two rounds), it accepts it as a possible output. If that party is on the committee, it signs this message and forwards it to all parties so they can accept it in the next stage. In our protocol, we additionally require that for a party P to accept a bit b in Stage r, all the accompanying signatures must come from committee members who have an id with an associated grade that depends on r. More concretely, for each additional stage of the protocol, parties accept one additional (lower) grade than in the previous stage. By the consistency in the view of honest parties on all grades, any signature that is accepted by one party P in Stage r will be accepted by any other honest party P' in Stage $r + 1$ and thus can safely be forwarded by P before accepting b. The result is a sublinear-round broadcast protocol.

Open questions. While we significantly weaken the setup assumptions and obtain a trustless sublinear-round broadcast protocol, the message communication complexity takes a hit compared to the bulletin-PKI linear-round protocol of Dolev and Strong (which has $O(n^3)$ communication complexity) or compared to the sublinear-round protocol with trusted setup of Chan *et al.* [10] (which has $O(n^2)$ communication complexity). Using the gossiping techniques from Tsimos *et al.* [38], we can bring down the communication to $\tilde{O}(n^4)$ at the cost of having the number of rounds polylogarithmic in n. Reducing the communication complexity further is a valuable future research problem.

1.3 Related Work

Our work, like most of the ones described here, considers an atomic send model (the adversary is weakly adaptive, meaning it cannot perform after-the-fact removals) and uses property-based definitions. In the case of a stronger adversarial model of *non-atomic sends*, where an adaptive adversary can corrupt a party and revert or modify its send action before it reaches all the other parties, Cohen *et al.* [11] investigate the feasibility of property-based and simulation-based adaptive broadcast (also assuming TLPs).

We start with the related literature for broadcast protocols in the setup-free case apart from a bulletin-board public key infrastructure (bulletin-PKI). The result of Pease, Shostak and Lamport [33] disallows any broadcast protocol in the asynchronous setting tolerating $n/3$ or more corruptions, so we focus on the synchronous setting. Dolev and Strong [14] give a protocol against an adaptive dishonest majority $t < n$ with a linear number of rounds and $O(n^3)$ communication complexity. The line of work [27, 26, 8] proposed protocols with reduced communication complexity of down to $\tilde{O}(n^2)$ but for honest majority and with a linear number of rounds. Abraham *et al.* [2] achieve expected constant rounds for Parallel Broadcast against a static adversary in the honest majority setting with bulletin-PKI. Other works in the adaptive dishonest majority case [38, 39] and in the bulletin-PKI model studied the amortized communication complexity of protocols over a number of broadcast instances, but achieve a linear number of rounds.

We now survey the literature on synchronous broadcast protocols for a dishonest majority that achieve sublinear round complexity. The first works obtained a sublinear number of rounds only in a narrow case $t/n - 1/2 \leq o(n)$: Garay *et al.* [21] achieved $O((2t - n)^2)$ rounds, and Fitzi and Nielsen [18] achieved $O(2t - n)$ rounds, against a strongly adaptive adversary.

Chan *et al.* [10] was the first result achieving broadcast with sublinear rounds $O(\frac{n}{n-t})$ and $\tilde{O}(n^2)$ communication in a dishonest majority $t/n - 1/2 \geq \omega(1)$. It requires a trusted setup for the common random strings and keys against a weakly adaptive adversary. Wan *et al.* [41] further improved this result by presenting a protocol for synchronous broadcast that achieves expected constant rounds $O((\frac{n}{n-t})^2)$ and $\tilde{O}(n^4)$ communication complexity, but still with trusted setup (in particular, this avoids having maliciously generated keys for the VRF, which we treat in this work). The solution requires building a trust graph which allows honest parties to identify the corrupted parties.

Wan *et al.* [40] tolerates stronger adversaries that can also erase messages, still with a sublinear number of rounds. In [40], parties distribute during each round their real or dummy votes through time-lock puzzles as a means of encryption against the adaptive adversary. In order to not have each honest party solve a linear number of puzzles, parties probabilistically sample which puzzle to solve based on the puzzles' age and then multicast the solution and the validity proof. This guarantees that after a logarithmic number of rounds, all honest puzzles are solved and their solutions are received by all honest nodes. However, this solution does not guarantee that corrupt puzzles are also solved or that honest parties have a consistent view of puzzles originating from the adversary. This is the main reason this solution cannot be used in our case of emulating random beacons, where an adversary can bias the result by observing the intermediate opened puzzles and deciding to not allow some corrupt puzzles to be opened.

Hou *et al.* [24] describe a blockchain that tolerates dishonest majority, loosely based on the Chan *et al.* broadcast protocol [10], proof of stake, proof of work in the random oracle model (ROM).

Graded broadcast, graded consensus, graded verifiable secret sharing, have been proposed in various settings as a stepping stone to stronger primitives of Byzantine broadcast or Byzantine agreement, see [16, 12, 25, 21] for instance. Recently, in the honest majority case, [19] generalized graded consensus to the multi-dimensional case which deals with a vector of inputs.

Time-lock puzzles (TLP) [34], timed commitments [7] and verifiable delay functions (VDF) [6] are cryptographic tools relying on time assumptions, which involve "slow functions" that can be opened or evaluated only after an a priori chosen amount of time passes. Several constructions of TLPs have been proposed [5, 32, 37, 1, 31, 15].

In the context of timing assumptions and broadcast, Das *et al.* [13] describe a Byzantine agreement protocol with VDF in ROM, which achieves an expected constant round complexity without trusted setup, tolerating $t < n/2$ adaptive corruptions. Their construction differs conceptually from ours: Das *et al.* [13] generate a graded PKI with only two grades, then use VDFs both in this construction and to elect a leader on which honest parties agree with high probability, in order to augment a graded byzantine agreement into a full byzantine agreement. However, their constructions heavily rely on $t < n/2$ and on the ROM. Srinivasan *et al.* [36] give a compiler based on TLPs (using indistinguishability obfuscation) that achieves expected constant-round Byzantine broadcast protocol against a strongly adaptive adversary for the dishonest majority setting, but without removing the trusted setup.

The use of timed functions for multi-party unbiased randomness generation was first exemplified by Lenstra *et al.* [30]. Constructing randomness from VDF/TLP with transparent setup in ROM, but assuming broadcast, is also addressed by Bhat *et al.* [4], who tolerate $t < n$ only if the adversary is covert, and by Thyagarajan *et al.* [37] who tolerate $t < n$. Freitag *et al.* [20] propose a fair coin flipping protocol that assumes either a public bulletin-PKI and a partially trusted setup called all-but-one model (non-interactive but where parties need to solve all TLPs) or a trusted setup and ROM (interactive but publicly verifiable).

1.4 Paper Organization

After recalling necessary preliminaries in Section 2, we construct our broadcast protocol in three steps. First, in Section 3 we describe the cryptographic core of our protocol, namely, algorithms to securely elect a committee without random oracles. We show two properties for these algorithms: (1) the adversary cannot corrupt all committee members, and (2) the committee is never too large. Roughly speaking, these properties hold from the perspective of a single party. In the second step, in Section 4, we use (moderated) gradecast to design a distributed protocol for committee election and we analyze the properties of these algorithms within the context of our distributed protocol. Then, in Section 5, we present and analyze our full broadcast protocol. We conclude in Section 6.

2 Preliminaries

In this section, we fix notation, define necessary cryptographic primitives, and recall some mathematical background.

Notation. We write $x \leftarrow\!\!\$\, X$ to denote that x is sampled uniformly at random from a finite set X. We write $y \leftarrow \mathcal{A}(x)$ when y is the output of a probabilistic algorithm $\mathcal{A}$ run on input x with uniform coins. To make the coins explicit, we write $y := \mathcal{A}(x; \rho)$, in which case the algorithm becomes deterministic. We denote the running time of an algorithm $\mathcal{A}$ by $\mathbf{T}(\mathcal{A})$. PPT stands for probabilistic polynomial time. Throughout, we use negl to denote a negligible function, which is a function that goes to zero faster than every inverse polynomial. Similarly, we use poly and $\mathsf{polylog}$ to denote functions that are polynomial or polylogarithmic. Throughout the paper, log denotes the natural logarithm.

Network. We consider n parties $P_1, \ldots, P_n$ that have access to a bulletin-board public key infrastructure (bulletin-PKI). Every party generates a pair of keys for signing (the signature function can be any signature function which does not require a trusted setup or a random oracle) and posts the public key to the public bulletin-board before the protocol starts. The posted keys are not guaranteed to have been generated correctly, but they are the same in the view of all parties. The bulletin-board also has public parameters (not requiring a trusted setup to generate), for instance common reference strings. The public bulletin-board is assumed to implement a one-time secure read and write for the parties.

We consider a *synchronous* network, i.e., messages between parties are delivered with a finite, known delay Δ_r, and the local clocks of the parties are synchronized. Our protocols execute in rounds: every round r of the protocol has length Δ_r and parties start executing round r at time $(r-1)\cdot\Delta_r$. We assume *atomic send operations*, i.e., parties can send a message to multiple parties simultaneously such that the adversary cannot corrupt them in between individual sends. The number of rounds that a protocol takes to terminate is denoted as its *round complexity.*

Security Parameters. Throughout the paper we denote with κ the computational security parameter. We let λ be a statistical security parameter, which we assume to be smaller than κ. Naturally, the number of parties n is polynomial, i.e., it satisfies $n \leq \mathsf{poly}(\kappa)$ and $n \leq \mathsf{poly}(\lambda)$. This also means that we have $n \cdot \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$ and $n \cdot \mathsf{negl}(\kappa) = \mathsf{negl}(\kappa)$. Similarly, we assume $n \geq \omega(\log \lambda)$. For smaller n, the goal of sublinear-round broadcast is not interesting.

Threat model. We consider a *Byzantine fault* model, in which some fraction of the parties $t \leq (1-\epsilon)n$, may be corrupted by an adversary, for a constant $\epsilon \in (0,1)$. The adversary controls the messages and current state of any corrupted party, and can coordinate the actions of all corrupted parties. The adversary is *adaptive and rushing*, i.e., it can adaptively corrupt parties over the course of a protocol execution and wait until all honest parties have sent their messages before making a decision. Uncorrupted parties are called *honest.* The adversary cannot perform *after the fact removals*, i.e., it cannot indefinitely prevent a message from being delivered once it is sent by an honest party, even if the adversary corrupts it at some point after the send action.

Idealized Signatures. We use the notation $\mathsf{Sig}_i(m)$ to denote a signature of party P_i using sk_i on message m and $\mathsf{SigVer}_i(s,m)$ to denote the verification of signature s on message m using public key pk_i. Signatures should achieve *correctness*: for any message m, it holds that $\mathsf{SigVer}_i(\mathsf{Sig}_i(m),m)=1$, and *unforgeability under chosen-message attack*: for a pair of honestly generated keys $(\mathsf{sk}_i,\mathsf{pk}_i)$, a party that does not have access to sk_i, cannot generate a signature s for a new message m such that $\mathsf{SigVer}_i(s,m)=1$. As standard in the consensus literature, we assume idealized signatures that satisfy correctness and unforgeability perfectly.

Assumption and Inequalities. We recall the discrete logarithm assumption and two useful inequalities.

Definition 1 (Discrete Logarithm Assumption)**.** *Let* GGen *be an algorithm that outputs the description of a cyclic group* $\mathbb{G}$ *with generator* g *of prime order* p*. We say that the discrete logarithm assumption holds relative to* GGen *if for any PPT algorithm* $\mathcal{A}$ *the following probability is negligible:*

$$\Pr\left[\mathcal{A}(\mathbb{G},g,p,g^x)=x \mid (\mathbb{G},g,p)\leftarrow \mathsf{GGen}(1^\kappa),\ x \leftarrow\!\!\$\, \mathbb{Z}_p\right].$$

Lemma 1 (Bernoulli's inequality)**.** *For every* $x\in\mathbb{R}$ *and any positive exponent* $r>0$*, it holds:*

$$(1+x)^r \leq e^{xr}. \tag{1}$$

Lemma 2 (Chernoff Bound)**.** *Assume that* $X_1,\dots,X_n$ *are independent* $\{0,1\}$ *random variables, let* $X := \sum_{i=1}^n X_i$ *and* $\mu := \mathbb{E}[X]$*. Then, for any* $\zeta \geq 0$*, we have*

$$\Pr\left[X \geq (1+\zeta)\mu\right] \leq e^{-\frac{\zeta^2\mu}{\zeta+2}}.$$

Cryptographic Primitives. An **NP** relation is a relation $\mathcal{R}$ containing pairs (x, w) of statement and witnesses that satisfies the following properties: (1) there is a polynomial q with $|w| \leq q(|x|)$ for all $(x, w) \in \mathcal{R}$, and (2) the relation is decidable deterministically in polynomial time. In addition, we allow $\mathcal{R}$ to be parameterized implicitly by the security parameter and assume that $|x| \leq \mathsf{poly}(\kappa)$ for all $(x, w) \in \mathcal{R}$. We define non-interactive proof systems.

Definition 2 (Non-Interactive Proof System)**.** *Let $\mathcal{R}$ be an* **NP** *relation. A non-interactive proof system for $\mathcal{R}$ is defined to be a triple* $\mathsf{PS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$ *of PPT algorithms with the following syntax:*

- $\mathsf{Setup}(1^\kappa) \to \mathsf{crs}$ *takes as input the security parameter and outputs a common reference string* crs.
- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$ *takes as input a common reference string* crs*, a statement x and a witness w, and outputs a proof π.*
- $\mathsf{Ver}(\mathsf{crs}, x, \pi) \to b$ *is deterministic, takes as input a common reference string* crs*, a statement x and a proof π, and outputs a bit $b \in \{0, 1\}$.*

Further, we require that the following completeness property holds: For any pair $(x, w) \in \mathcal{R}$, we have

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, x, \pi) = 1 \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\right] = 1.$$

Most importantly, proof systems need to be sound. For our purpose, computational soundness is enough. Such systems are often referred to as argument systems, but we use the term proof system throughout.

Definition 3 (Soundness)**.** *Consider an* **NP** *relation $\mathcal{R}$ and a non-interactive proof system* $\mathsf{PS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$ *for $\mathcal{R}$. We say that* PS *is (computationally) sound, if for every PPT algorithm $\mathcal{A}$, the probability that the following game outputs* 1 *is negligible:*

1. *Run* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$.
2. *Run $\mathcal{A}$ on input* crs *and obtain a pair (x, π).*
3. *Output* 1 *and terminate if the following two conditions hold. Otherwise, output* 0*:*
   (a) *We have* $\mathsf{Ver}(\mathsf{crs}, x, \pi) = 1$.
   (b) *There is no witness w such that $(x, w) \in \mathcal{R}$.*

We require a proof system that is zero-knowledge and simulation-extractable, according to the following definitions. Note that the definitions are weak in a sense that the adversary is non-adaptive.

Definition 4 (Zero-Knowledge)**.** *Consider an* **NP** *relation $\mathcal{R}$ and a non-interactive proof system* $\mathsf{PS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$ *for $\mathcal{R}$. We say that* PS *is zero-knowledge, if there are PPT algorithms* $\mathsf{TrapSetup}$, Sim *such that for every PPT algorithm $\mathcal{A}$, the probability that the following game outputs* 1 *is negligibly close to* $1/2$*:*

1. *Sample* $b \leftarrow_{\$} \{0, 1\}$.
2. *Generate a common reference string* crs *as follows:*
   (a) *If $b = 0$, run* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa)$.

   (b) *If* $b = 1$, *run* $(\mathsf{crs}, \mathsf{trap}) \leftarrow \mathsf{TrapSetup}(1^\kappa)$.

3. *Run* $\mathcal{A}$ *on input* crs *and obtain a state* st *and pairs* $(x_1, w_1), \ldots, (x_L, w_L)$ *of statements and witnesses.*

4. *If there is a* $j \in [L]$ *with* $(x_j, w_j) \notin \mathcal{R}$, *terminate and output* 0.

5. *Otherwise, for each* $j \in [L]$, *compute* π_j *as follows:*

   (a) *If* $b = 0$, *run* $\pi_j \leftarrow \mathsf{Prove}(\mathsf{crs}, x_j, w_j)$.

   (b) *If* $b = 1$, *run* $\pi_j \leftarrow \mathsf{Sim}(\mathsf{trap}, x_j)$.

6. *Run* $\mathcal{A}$ *on input* st *and* $\pi_1, \ldots, \pi_L$ *and obtain a bit* b'.

7. *Output* 1 *if* $b = b'$. *Otherwise, output* 0.

In this case, we say that $\mathsf{TrapSetup}$ *is the trapdoor setup algorithm and* Sim *is the zero-knowledge simulator for* PS.

Definition 5 (Simulation-Extractability)**.** *Let* $\mathcal{R}$ *be an* **NP** *relation and* $\mathsf{PS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$ *be a non-interactive proof system for* $\mathcal{R}$. *Assume that* PS *is zero-knowledge with trapdoor setup algorithm* $\mathsf{TrapSetup}$ *and zero-knowledge simulator* Sim. *We say that* PS *is simulation-extractable, if there is a PPT algorithm* Ext *such that for every PPT algorithm* $\mathcal{A}$, *the probability that the following game outputs* 1 *is negligible:*

1. *Generate* $(\mathsf{crs}, \mathsf{trap}) \leftarrow \mathsf{TrapSetup}(1^\kappa)$.

2. *Run* $\mathcal{A}$ *on input* crs *and obtain a state* st *and statements* $x_1, \ldots, x_L$.

3. *For each* $j \in [L]$, *run* $\pi_j \leftarrow \mathsf{Sim}(\mathsf{trap}, x_j)$.

4. *Run* $\mathcal{A}$ *on input* st *and* $\pi_1, \ldots, \pi_L$ *and obtain pairs* $(x_1^*, \pi_1^*), \ldots, (x_T^*, \pi_T^*)$ *of statements and proofs.*

5. *If there are* $i \in [T]$ *and* $j \in [L]$ *with* $x_i^* = x_j$, *then terminate and output* 0.

6. *For each* $i \in [T]$, *run* $w_i^* := \mathsf{Ext}(\mathsf{trap}, x_i^*, \pi_i^*)$.

7. *If there is an* $i \in [T]$ *such that* $\mathsf{Ver}(\mathsf{crs}, x_i^*, \pi_i^*) = 1$ *and* $(x_i^*, w_i^*) \notin \mathcal{R}$, *terminate and output* 1.

8. *Otherwise, output* 0.

In this case, we refer to Ext *as the knowledge extractor of* PS.

There are constructions of non-interactive proof systems satisfying our notion with a common *random* string, e.g., [28].

We now introduce time-lock puzzles with batch solving following [36, 15].

Definition 6 (Time-Lock Puzzles with Batch Solving)**.** *A time-lock puzzle with batch solving is a tuple of PPT algorithms* $\mathsf{TLP} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve})$ *with the following syntax:*

- $\mathsf{Setup}(1^\kappa) \rightarrow \mathsf{tlpar}$ *takes as input the security parameter and outputs parameters* tlpar.

- $\mathsf{Gen}(\mathsf{tlpar}, s) \rightarrow Z$ *takes as input parameters* tlpar *and a solution* s *and outputs a puzzle* Z.

- $\mathsf{Solve}(\mathsf{tlpar}, (Z_i)_{i=1}^L) \to (s_i)_{i=1}^L$ *is deterministic, takes as input parameters* tlpar *and a list of puzzles* Z_i *and outputs a list of solutions* s_i.

Further, we require that the following completeness property holds: for every $\mathsf{tlpar} \in \mathsf{Setup}(1^\kappa)$*, every* $L \in \mathbb{N}$*, and every list of solutions* $s_1, \ldots, s_L$*, and puzzles* $Z_i \in \mathsf{Gen}(\mathsf{tlpar}, s_i)$ *for all* $i \in [L]$*, we have* $\mathsf{Solve}(\mathsf{tlpar}, (Z_i)_{i=1}^L) = (s_i)_{i=1}^L$.

Definition 7 (CPA Security of TLPs)**.** *Let* $\mathsf{TLP} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Solve})$ *be a time-lock puzzle with batch solving and* Δ *be a parameter. We say that* TLP *is* Δ*-CPA secure, if for every PPT algorithm* $\mathcal{A}_{pre}$ *and every PPT algorithm* $\mathcal{A}_{on}$ *such that* $\mathcal{A}_{on}$ *runs in time at most* Δ*, the probability that the following game outputs* 1 *is negligibly close to* $1/2$*:*

1. *Sample* $b \leftarrow\$ \{0, 1\}$ *and generate* $\mathsf{tlpar} \leftarrow \mathsf{Setup}(1^\kappa)$.
2. *Run* $\mathcal{A}_{pre}$ *on input* tlpar *to obtain a state* st *and two solutions* s_0, s_1.
3. *Generate* $Z \leftarrow \mathsf{Gen}(\mathsf{tlpar}, s_b)$ *and run* $\mathcal{A}_{on}$ *on input* (st, Z).
4. *Obtain a bit* $b' \in \{0, 1\}$ *from* $\mathcal{A}_{on}$ *and output* 1 *if* $b = b'$*. Else, output* 0.

We assume that Setup and Gen run in time substantially faster than Δ, and that honestly running Solve runs in time $p_1(\kappa, \Delta) + p_2(\kappa, L)$ for polynomials p_1, p_2. Note that the running time of Solve does not scale with $\Delta \cdot L$. We can obtain time-lock puzzles with batch opening from any homomorphic time-lock puzzles [32]. There exists a construction of CPA secure homomorphic time-lock puzzles with transparent setup [37], without random oracles,[5] based on the hardness of computing the order of class groups of imaginary order [9]. More communication efficient constructions based on [36] and [15] are imaginable.

Protocols. In the following, we implicitly assume the definitions are for protocols tolerating t malicious parties, i.e., the conditions hold whenever there are at most t corrupted parties.

Introduced in the seminal work by Lamport *et al.* [29], *broadcast* ensures agreement of honest parties on a sender's message. We focus our exposition on binary broadcast, where input values are bits, but it can be extended to multi-bit values.

Definition 8 (Broadcast)**.** *Consider a protocol executed by parties* $P_1, \ldots, P_n$*, where a designated sender* $P^* \in \{P_1, \ldots, P_n\}$ *initially holds an input* x^**. We say that such a protocol is a* broadcast *protocol if the following properties hold:*

- ***Validity.*** *If* P^* *is honest, then every honest party outputs* x^*.
- ***Consistency.*** *Every honest party outputs the same value* x.
- ***Termination.*** *Each honest party* P_i *outputs* (x_i) *and terminates.*

Gradecast, introduced by Feldman and Micali in [17] and generalized for an arbitrary grade by Garay *et al.* [21], is a relaxation of broadcast, where honest parties are allowed to disagree by a "small amount".

Definition 9 (Gradecast)**.** *Consider a protocol executed by parties* $P_1, \ldots, P_n$*, where a designated sender* $P^* \in \{P_1, \ldots, P_n\}$ *initally holds an input* x^**. We say that such a protocol is a* g^*-gradecast *protocol if the following properties hold:*

[5] Only the CCA secure construction in [37] requires ROM for concrete efficiency.

- ***Validity.*** *If P^* is honest, then every honest party outputs (x^*, g^*).*
- ***Soundness.*** *Let P_i, P_j be two honest parties outputting (x_i, g_i) and (x_j, g_j), respectively. If $g_i \geq 2$, then $x_i = x_j$ and $|g_i - g_j| \leq 1$. If $g_i = 1$, then either $x_i = x_j$ or $g_j = 0$.*
- ***Termination.*** *Each honest party P_i outputs (x_i, g_i) where $g \in \{0, \ldots, g^*\}$ and terminates.*

In this work, we will use the gradecast construction from Garay *et al.* [21]. We also define *moderated gradecast* where a moderator M re-gradecasts the value it received from the sender in gradecast P^*. The goal is for the honest parties to use the two pieces of information coming from the two gradecasts with different senders, to obtain "similar" outputs and to grade the moderator.

Definition 10 (Moderated Gradecast)**.** *A protocol executed by parties $P_1, \ldots, P_n$, where a sender $P^* \in \{P_1, \ldots, P_n\}$ begins holding input x^*, and a moderator $M \in \{P_1, \ldots, P_n\}$ moderates for the sender P^*, is a g^*-moderated gradecast protocol if the following notions hold:*

- ***Validity.*** *If P^* is honest and moderator M is also honest, every honest party P_i outputs (x^*, g^*).*
- ***M-Validity.*** *If moderator M is honest, then every honest party P_i outputs (x, g_i) for some x and for $g_i \in \{g^* - 1, g^*\}$.*
- ***Soundness.*** *Let P_i, P_j be two honest parties outputting (x_i, g_i) and (x_j, g_j), respectively. If $g_i \geq 2$, then $x_i = x_j$ and $|g_i - g_j| \leq 1$. If $g_i = 1$, then either $x_i = x_j$ or $g_j = 0$.*
- ***Termination.*** *Each honest party P_i outputs (x_i, g_i) where $g_i \in \{0, \ldots, g^*\}$ and terminates.*

3 Algorithms for Committee Election

In this section, we construct the cryptographic core of our broadcast protocol, namely, algorithms to run an unbiased committee election. We then formally specify two security games and prove that no efficient adversary can win these games. Intuitively, they model that—under the assumption that the outputs of the algorithms are distributed consistently—the resulting committee contains at least one honest party and is not too large. Later, we will use various flavors of gradecast to turn our committee election scheme into a full protocol.

3.1 Informal Description

Our committee election scheme CES consists of five algorithms. Before we formally specify these algorithms, we informally introduce them and summarize how we envision using them, thereby also introducing all relevant notation.

Setup. We assume parties have access to a common set of system parameters par generated by an algorithm $\mathsf{CES.Setup}$. These include the description of a cyclic group $\mathbb{G}$ with generator g and prime order p, namely, $(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\kappa)$. In addition, the parameters contain a group element $h \leftarrow\!\!\$\, \mathbb{G}$, time-lock parameters $\mathsf{tlpar} \leftarrow \mathsf{TLP.Setup}(1^\kappa)$, as well as common reference strings $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\kappa)$ and $\tilde{\mathsf{crs}} \leftarrow \tilde{\mathsf{PS}}\mathsf{.Setup}(1^\kappa)$ as public parameters. Here, PS is a proof system for relation

$$\mathcal{R} := \{(X, (x, r)) \mid X = g^x h^r\},$$

and $\tilde{\mathsf{PS}}$ is a proof system for the relation

$$\tilde{\mathcal{R}} := \left\{ ((Z, (c_j)_{j=1}^n), ((T_j, \gamma_j)_{j=1}^n, \rho)) \;\middle|\; \begin{array}{c} Z = \mathsf{TLP.Gen}(\mathsf{tlpar}, (T_j, \gamma_j)_{j=1}^n; \rho) \\ \wedge \quad \forall j \in [n] : c_j = g^{T_j} h^{\gamma_j} \end{array} \right\}.$$

Looking ahead, with PS parties will prove that their keys are well-formed, and with $\tilde{\mathsf{PS}}$ parties prove that their time-lock puzzles are well-formed. We will elaborate on the rationale for defining the relation $\tilde{\mathcal{R}}$ like this below.

Predicates. We make use of a predicate $\mathsf{Pred}\colon \mathbb{Z}_p \to \{0,1\}$. For now, we treat this predicate abstractly and postpone a concrete instantiation to the very end of this section. For convenience, we define the *bias* of the predicate and a bit $b \in \{0,1\}$ as

$$\mathsf{bias}(\mathsf{Pred}, b) := \Pr\left[\mathsf{Pred}(\nu) = b \mid \nu \leftarrow\!\!\$\, \mathbb{Z}_p\right].$$

We also define the *tail* of the predicate as

$$\mathsf{tail}(\mathsf{Pred}, n, B) := \Pr\left[|\{j \in [n] \mid \mathsf{Pred}(\nu_j) = 1\}| > B \mid \nu_1, \ldots, \nu_n \leftarrow\!\!\$\, \mathbb{Z}_p\right].$$

That is, the tail denotes the probability that among n uniform and independent values, more than B of them satisfy the predicate.

Key Generation. Before running the committee election, each party P_i generates a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ via a key generation algorithm $\mathsf{CES.Gen}$. This is done as follows: the secret key sk_i contains two random exponents $x_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$. The public key pk_i consists of a commitment $X_i := g^{x_i} h^{r_i}$ and a proof of well-formedness π_i. Namely, this proof is computed as $\pi_i \leftarrow \mathsf{PS.Prove}(\mathsf{crs}, X_i, (x_i, r_i))$ and shows that the party computed X_i honestly. Looking ahead, parties with invalid proofs will never be considered elected. For now, the reader may think of the public keys as being posted on a public bulletin-board, i.e., all parties have a consistent view of these keys. When we use our committee election scheme to construct a protocol, we will avoid making this assumption by using a form of weak agreement.

Sampling Tags. To prevent the election from being biased by the adversary, all parties jointly generate randomness. This is done using what we call tags and pretags. In more detail, we assume that each party P_i generate $\mathsf{pretags}_i$ via an algorithm $\mathsf{CES.Prepare}$. The set of all these $\mathsf{pretags}_i$ will later help determine who is in the committee. Let us now explain how $\mathsf{pretags}_i$ is generated: party P_i locally samples a list of tags $T_{i,j} \leftarrow\!\!\$\, \mathbb{Z}_p$, one tag for each other party P_j, $j \in [n]$. Then, P_i commits to these tags by sampling $\gamma_{i,j} \leftarrow\!\!\$\, \mathbb{Z}_p$ for all $j \in [n]$ and setting $c_{i,j} := g^{T_{i,j}} h^{\gamma_{i,j}}$, and sets $\mathsf{pretags}_i$ to consist of the commitments and time-lock puzzles of the tags. More concretely, P_i samples random coins ρ_i for the algorithm $\mathsf{TLP.Gen}$. It then computes a puzzle $Z_i := \mathsf{TLP.Gen}(\mathsf{tlpar}, (T_{i,j}, \gamma_{i,j})_{j=1}^n; \rho_i)$ and a proof $\tilde{\pi}_i \leftarrow \tilde{\mathsf{PS}}.\mathsf{Prove}(\tilde{\mathsf{crs}}, (Z_i, (c_{i,j})_{j=1}^n), ((T_{i,j}, \gamma_{i,j})_{j=1}^n, \rho_i))$. Recall that $\tilde{\pi}_i$ proves that Z_i is a valid puzzle for the tags $T_{i,j}$ that are committed in the commitments $c_{i,j}$. That is, the proof links the puzzles to the commitments. Intuitively, to ensure that malicious parties cannot make their tags depend on other tags, we will make sure that all parties output their $\mathsf{pretags}_i$ within a certain time interval that is related to the hardness of the puzzles.[6]

Winning the election. Informally, after the pretags are distributed, each party P_k will be associated with a random value

$$\mathsf{id}_k = x_k + \sum_i T_{i,k}.$$

Notably, we let i range *only* over the indices for which $\mathsf{pretags}_i$ contains a proof $\tilde{\pi}_i$ that verifies. Phrased differently, if a party provides an invalid proof, all tags provided by that party are treated as zero. Therefore, for all tags that contribute to the sum, we can be sure that the commitments and the puzzles were consistent (by the soundness of $\tilde{\mathsf{PS}}$). A party P_k having access to the list $(\mathsf{pretags}_i)_{i=1}^n$ can therefore check if it has been elected into the committee as follows, modeled by

[6]Importantly, this relies on the synchronous model of communication.

algorithm $\mathsf{CES.TryElect}$: first, it batch solves all puzzles with valid proofs $\tilde{\pi}_i$, obtaining the tags $T_{i,k}$ and the corresponding $\gamma_{i,k}$; second, it computes the value id_k as above; it concludes that it has won the election if and only if $\mathsf{Pred}(\mathsf{id}_k) = 1$.

Convincing Others of Committee Membership. Say that a party P_k with public key $\mathsf{pk}_j = (X_k, \pi_k)$ holding the list $(\mathsf{pretags}_i)_{i=1}^n$ wants to convince a second party P_l that it has been elected to the committee. First of all, if π_k is invalid, P_l will reject immediately and not consider P_k as a committee member. Otherwise, P_l expects P_k to provide its secret key $\mathsf{sk}_k = (x_k, r_k)$. Party P_l then checks that (x_k, r_k) is indeed consistent with the commitment X_k, and that $\mathsf{Pred}(\mathsf{id}_k) = 1$. Needless to say, to compute id_k, party P_l also needs the tags $T_{i,k}$ (for all i for which $\tilde{\pi}_i$ is valid).

Note that, if we assume that parties have a consistent view of the $(\mathsf{pretags}_i)_{i=1}^n$ and of pk_k, then by solving the puzzles for itself, P_l learns the tags $T_{i,k}$ and does not need P_k to provide its solved tags as proof of committee membership. However, since we need to avoid this strong assumption when constructing our protocol in the subsequent sections, we also weaken the setup here. In particular, we allow the possibility that corrupted parties provide different pretags to different honest parties. We will go into details about how to solve this in Section 4, but for the moment, this translates into the following. In order to verify that a party is in the committee, the verifying party P_l needs to know the tags (and the secret key) for each such party P_k, and so P_l would need to resolve the puzzles contained in the pretags for every party P_k that is claiming to be in the committee. This naive solution would require each verifying party to solve $\Theta(n)$ batch puzzles, which is too costly. Therefore, our goal is to avoid this. To do so, we rely on the soundness of $\widetilde{\mathsf{PS}}$: we know that the output of a puzzle Z_i is consistent with the commitments $(c_{i,j})_{i=1}^n$. Therefore, we let P_k provide to P_l the tags $T_{i,k}$ along with their openings $\gamma_{i,k}$. Party P_l can then check the correctness of the tags by comparing its local view on $c_{i,k}$ with the provided $T_{i,k}$ and $\gamma_{i,k}$. To summarize, party P_k convinces party P_l of its membership in the committee by providing ticket_k, which contains x_k, r_k and $T_{i,k}, \gamma_{i,k}$ for all indices i which have valid proofs $\tilde{\pi}_i$. We model verifying such a claim and ticket_k by algorithm $\mathsf{CES.VerElect}$.

3.2 Formal Definition

With our previous explanations in mind, we now define our committee election scheme CES by the following algorithms:

- $\mathsf{CES.Setup}(1^\kappa) \to \mathsf{par}$:
  1. $(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\kappa)$, $\quad h \leftarrow_\$ \mathbb{G}$
  2. $\mathsf{tlpar} \leftarrow \mathsf{TLP.Setup}(1^\kappa)$
  3. $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\kappa)$, $\quad \tilde{\mathsf{crs}} \leftarrow \widetilde{\mathsf{PS}}.\mathsf{Setup}(1^\kappa)$
  4. $\mathsf{par} := (\mathbb{G}, g, p, h, \mathsf{tlpar}, \mathsf{crs}, \tilde{\mathsf{crs}})$
- $\mathsf{CES.Gen}(\mathsf{par}) \to (\mathsf{pk}_i, \mathsf{sk}_i)$:
  1. $x_i \leftarrow_\$ \mathbb{Z}_p,\ r_i \leftarrow_\$ \mathbb{Z}_p,\ X_i := g^{x_i} h^{r_i}$
  2. $\pi_i \leftarrow \mathsf{PS.Prove}(\mathsf{crs}, X_i, (x_i, r_i))$
  3. $\mathsf{pk}_i := (X_i, \pi_i)$, $\quad \mathsf{sk}_i := (x_i, r_i)$
- $\mathsf{CES.Prepare}(\mathsf{par}) \to \mathsf{pretags}_i$:
  1. For all $j \in [n]$: $\ T_{i,j} \leftarrow_\$ \mathbb{Z}_p,\ \gamma_{i,j} \leftarrow_\$ \mathbb{Z}_p,\ c_{i,j} := g^{T_{i,j}} h^{\gamma_{i,j}}$

2. Sample random coins ρ_i for algorithm $\mathsf{TLP.Gen}$
3. $Z_i := \mathsf{TLP.Gen}(\mathsf{tlpar}, (T_{i,j}, \gamma_{i,j})_{j=1}^n; \rho_i)$
4. $\tilde{\pi}_i \leftarrow \tilde{\mathsf{PS}}.\mathsf{Prove}(\tilde{\mathsf{crs}}, (Z_i, (c_{i,j})_{j=1}^n), ((T_{i,j}, \gamma_{i,j})_{j=1}^n, \rho_i))$
5. $\mathsf{pretags}_i := (Z_i, (c_{i,j})_{j=1}^n, \tilde{\pi}_i)$

- $\mathsf{CES.TryElect}(\mathsf{par}, \mathsf{sk}_k, (\mathsf{pretags}_i)_{i=1}^n) \rightarrow (\mathsf{res}, \mathsf{ticket}_k)$:

```
1. Parse $\mathsf{sk}_k := (x_k, r_k)$
2. For all $i \in [n]$: Parse $\mathsf{pretags}_i = (Z_i, (c_{i,j})_{j=1}^n, \tilde{\pi}_i)$
3. $\mathsf{ValidPre} := \{i \in [n] \mid \tilde{\mathsf{PS}}.\mathsf{Ver}(\tilde{\mathsf{crs}}, (Z_i, (c_{i,j})_{j=1}^n), \tilde{\pi}_i) = 1\}$
4. $((T_{i,j}, \gamma_{i,j})_{j=1}^n)_{i \in \mathsf{ValidPre}} := \mathsf{TLP.Solve}(\mathsf{tlpar}, (Z_i)_{i \in \mathsf{ValidPre}})$
5. $\mathsf{id}_k := x_k + \sum_{i \in \mathsf{ValidPre}} T_{i,k}$
6. $\mathsf{ticket}_k := (x_k, r_k, (T_{i,k}, \gamma_{i,k})_{i \in \mathsf{ValidPre}})$
7. If $\mathsf{Pred}(\mathsf{id}_k) = 0$: return $(0, \bot)$
8. Else: return $(1, \mathsf{ticket}_k)$
```

- $\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k, (\mathsf{pretags}_i)_{i=1}^n, \mathsf{ticket}_k) \rightarrow \mathsf{res}$:

```
1. Parse $\mathsf{pk}_k = (X_k, \pi_k)$. If $\mathsf{PS.Ver}(\mathsf{crs}, X_k, \pi_k) = 0$: return 0
2. For all $i \in [n]$: Parse $\mathsf{pretags}_i = (Z_i, (c_{i,j})_{j=1}^n, \tilde{\pi}_i)$
3. $\mathsf{ValidPre} := \{i \in [n] \mid \tilde{\mathsf{PS}}.\mathsf{Ver}(\tilde{\mathsf{crs}}, (Z_i, (c_{i,j})_{j=1}^n), \tilde{\pi}_i) = 1\}$
4. Parse $\mathsf{ticket}_k = (x_k, r_k, (T_{i,k}, \gamma_{i,k})_{i \in \mathsf{ValidPre}})$
5. If $g^{x_k} h^{r_k} \neq X_k$: return 0
6. If there is an $i \in \mathsf{ValidPre}$ such that $g^{T_{i,k}} h^{\gamma_{i,k}} \neq c_{i,k}$: return 0
7. $\mathsf{id}_k := x_k + \sum_{i \in \mathsf{ValidPre}} T_{i,k}$
8. Return $\mathsf{res} := \mathsf{Pred}(\mathsf{id}_k)$
```

3.3 Committee Corruption Game

The first security property that we show informally states that no adversary can corrupt all members of the committee. The game is strong in a sense that we allow the adversary to select *all* tags. Also, for each honest party j, we allow the adversary to choose an entirely different set of pretags $(\mathsf{pretags}_i^{(j)})_i$. These are then used (1) by honest party j to check if it is in the committee, and (2) by other honest parties to verify if party j is in the committee. To avoid clutter, the state of $\mathcal{A}$ is kept implicit in the game. Formally, the *committee corruption game* for a number of parties n and a corruption threshold t is defined as follows:

1. **Setup.** The game runs $\mathsf{par} \leftarrow \mathsf{CES.Setup}(1^\kappa)$.

2. **Initialization of Parties.** In the initialization phase, the adversary can declare its initial set of corrupted parties and learns the public keys of honest parties. More precisely:

   (a) The game runs $\mathcal{A}$ on input par. Then, $\mathcal{A}$ declares a set $\mathsf{Corr}_0 \subseteq [n]$. If $|\mathsf{Corr}_0| > t$, the game terminates and outputs 0.

   (b) The game sets $\mathsf{Corr} := \mathsf{Corr}_0$ and $\mathsf{Hon}_0 := [n] \setminus \mathsf{Corr}_0$.

(c) For each $i \in \mathsf{Hon}_0$, the game runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{CES.Gen}(\mathsf{par})$.

(d) The game gives all pk_i for $i \in \mathsf{Hon}_0$ to $\mathcal{A}$.

(e) At any time from this point on, $\mathcal{A}$ gets access to a corruption oracle. On input $i \in [n]$, if $i \in \mathsf{Corr}$ or if $|\mathsf{Corr}| = t$, then the oracle outputs $\perp$. Otherwise, the oracle adds i to Corr, and outputs sk_i.

3. **Tag Phase.** The game obtains $\mathsf{pretags}_i^{(k)}$ for all $i \in [n]$ and $k \in [n]$ from $\mathcal{A}$.

4. **Winning Condition.** The game ends as soon as $|\mathsf{Corr}| = t$. The adversary wins if none of the remaining honest parties is elected:

(a) Let $\mathsf{Hon}^* := [n] \setminus \mathsf{Corr}$ be the set of $n - t$ remaining honest parties.

(b) For every $k \in \mathsf{Hon}^*$, the game runs $(\mathsf{res}_k, \mathsf{ticket}_k) \leftarrow \mathsf{CES.TryElect}(\mathsf{par}, \mathsf{sk}_k, (\mathsf{pretags}_i^{(k)})_{i=1}^n)$.

(c) The game defines the set of honest committee members as

$$\mathsf{HonComm} := \left\{k \in \mathsf{Hon}^* \mid \mathsf{res}_k = 1 \land \mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k, (\mathsf{pretags}_i^{(k)})_{i=1}^n, \mathsf{ticket}_k) = 1\right\}.$$

(d) The game outputs 1 if $\mathsf{HonComm} = \emptyset$. Otherwise, it outputs 0.

Next, we show that no efficient adversary can win the committee corruption game for the protocol described above. That is, no efficient algorithm can corrupt all parties of the committee. We do not need any assumptions on the time-lock puzzles here.

Lemma 3. *Assume that* PS *is zero-knowledge (see Definition 4), that* $\tilde{\mathsf{PS}}$ *is sound (see Definition 3), and that* $p > \omega(\log \kappa)$. *Let* $\mathcal{A}$ *be a PPT algorithm in the committee corruption game for a number of parties* n *and a corruption threshold* t. *Then, the probability that the committee corruption game outputs* 1 *is at most* $\mathsf{negl}(\kappa) + \mathsf{bias}(P, 0)^{n-t}$.

Proof. Fix a PPT adversary $\mathcal{A}$ and let $\varepsilon^{\mathcal{A}}$ be the probability that the committee corruption game outputs 1. Our goal is to upper bound $\varepsilon^{\mathcal{A}}$. We do so by providing a sequence of hybrid games $\mathbf{H}_0, \ldots, \mathbf{H}_4$. For each hybrid $\mathbf{H}_i$, we denote the probability that it outputs 1 when run with adversary $\mathcal{A}$ by $\varepsilon_i^{\mathcal{A}}$.

Hybrid $\mathbf{H}_0$. This is the committee corruption game. To fix notation, we recall it here. The game first generates parameters par via algorithm $\mathsf{CES.Setup}$. Recall that par include the group $\mathbb{G}$ with generator g, a group element $h \leftarrow_\$ \mathbb{G}$, time-lock parameters tlpar and common reference strings crs and $\tilde{\mathsf{crs}}$ for PS and $\tilde{\mathsf{PS}}$, respectively. Next, the adversary gets par as input and declares its initial set $\mathsf{Corr}_0 \subseteq [n]$ of corrupted parties (with $|\mathsf{Corr}_0| \leq t$. The set of initially honest parties is $\mathsf{Hon}_0 := [n] \setminus \mathsf{Corr}_0$. For each such party $i \in \mathsf{Hon}_0$, the game computes a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ via algorithm $\mathsf{CES.Gen}$. To recall, we have $\mathsf{pk}_i = (X_i, \pi_i)$ and $\mathsf{sk}_i = (x_i, r_i)$, where x_i, r_i are uniformly chosen in $\mathbb{Z}_p$, $X_i = g^{x_i} h^{r_i}$, and π_i is a proof for this equality computed using $\mathsf{PS.Prove}$. At any time during the game, $\mathcal{A}$ can corrupt a party i, thereby learning $\mathsf{sk}_i = (x_i, r_i)$. The set Corr gets updated accordingly by inserting i. In the *tag phase*, $\mathcal{A}$ outputs $\mathsf{pretags}_i^{(k)} = (Z_i^{(k)}, (c_{i,j}^{(k)})_{j=1}^n, \tilde{\pi}_i^{(k)})$ for all $i \in [n]$ and all $k \in [n]$. The game continues until $\mathcal{A}$ has corrupted exactly t parties. Let $\mathsf{Hon}^* := [n] \setminus \mathsf{Corr}$ be the set of $n - t$ remaining honest parties. For each $k \in \mathsf{Hon}^*$, the game runs algorithm $\mathsf{CES.TryElect}$. As in this algorithm, let

$$\mathsf{ValidPre}^{(k)} := \{i \in [n] \mid \tilde{\mathsf{PS}}.\mathsf{Ver}(\tilde{\mathsf{crs}}, (Z_i^{(k)}, (c_{i,j}^{(k)})_{j=1}^n), \tilde{\pi}_i^{(k)}) = 1\}$$

and

$$((T_{i,j}^{(k)},\gamma_{i,j}^{(k)})_{j=1}^{n})_{i\in\mathsf{ValidPre}^{(k)}} := \mathsf{TLP.Solve}(\mathsf{tlpar},(Z_i^{(k)})_{i\in\mathsf{ValidPre}^{(k)}}).$$

Then, to check the *winning condition*, the game computes

$$\mathsf{id}_k := x_k + \sum_{i\in\mathsf{ValidPre}^{(k)}} T_{i,k}^{(k)}$$

for each $k \in \mathsf{Hon}^*$ as in $\mathsf{CES.TryElect}$. Then, it defines the honest committee members as the remaining honest parties k for which $\mathsf{Pred}(\mathsf{id}_k) = 1$ and $\mathsf{CES.VerElect}$ outputs 1, concretely

$$\begin{aligned}\mathsf{HonComm} &= \left\{k \in \mathsf{Hon}^* \;\middle|\; \mathsf{Pred}(\mathsf{id}_k) = 1 \wedge X_k = g^{x_k}h^{r_k} \wedge \forall i \in \mathsf{ValidPre}^{(k)} :\; g^{T_{i,k}^{(k)}}h^{\gamma_{i,k}^{(k)}} = c_{i,k}^{(k)}\right\}\\ &= \left\{k \in \mathsf{Hon}^* \;\middle|\; \mathsf{Pred}(\mathsf{id}_k) = 1 \wedge \forall i \in \mathsf{ValidPre}^{(k)} :\; g^{T_{i,k}^{(k)}}h^{\gamma_{i,k}^{(k)}} = c_{i,k}^{(k)}\right\}.\end{aligned}$$

Here, we have used the definition of $\mathsf{CES.VerElect}$, that $\mathsf{CES.VerElect}$ and $\mathsf{CES.TryElect}$ define the same set $\mathsf{ValidPre}^{(k)}$ when they get the same list $(\mathsf{pretags}_i^{(k)})_{i=1}^{n}$ as input, and the fact that the X_k have been computed honestly by the game. Finally, the game outputs 1 if $\mathsf{HonComm} = \emptyset$. By definition, we get

$$\varepsilon^{\mathcal{A}} = \varepsilon_0^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_1$. We change how $\mathsf{HonComm}$ is defined. Namely, we now define it as

$$\mathsf{HonComm} = \{k \in \mathsf{Hon}^* \mid \mathsf{Pred}(\mathsf{id}_k) = 1\}.$$

Observe that if the definition differs from the definition in the previous hybrid, then there must be an $i \in [n]$ and a $k \in \mathsf{Hon}^*$ such that $i \in \mathsf{ValidPre}^{(k)} \wedge g^{T_{i,k}^{(k)}}h^{\gamma_{i,k}^{(k)}} \neq c_{i,k}^{(k)}$. In particular, for at least one $i^* \in \mathsf{ValidPre}^{(k)}$, it must hold that

$$\exists j \in [n] : g^{\hat{T}_{i^*,j}}h^{\hat{\gamma}_{i^*,j}} \neq c_{i^*,j}^{(j)}$$

where $\hat{T}_{i^*,j}$ and $\hat{\gamma}_{i^*,j}$ are such that $Z_{i^*}^{(k)} \in \mathsf{TLP.Gen}(\mathsf{tlpar},(\hat{T}_{i^*,j},\hat{\gamma}_{i^*,j})_{j=1}^{n})$. Indeed, if there was no such i^*, then by completeness of TLP there could never have been such an i. Hence, we can build a PPT reduction that breaks soundness of $\tilde{\mathsf{PS}}$ by outputting the statement $(Z_{i^*}^{(k)},(c_{i^*,j}^{(k)})_{j=1}^{n})$ and the proof $\tilde{\pi}_{i^*}^{(k)}$ it received from $\mathcal{A}$ in the tag phase. The proof verifies because $i^* \in \mathsf{ValidPre}^{(k)}$. By our assumption that $\tilde{\mathsf{PS}}$ is sound, we get

$$\left|\varepsilon_0^{\mathcal{A}} - \varepsilon_1^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_2$. This hybrid is the same as $\mathbf{H}_1$, but we change the way the common reference string crs and the proofs π_i are generated. Namely, recall that they have been generated in $\mathbf{H}_1$ as

$$\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\kappa),\quad \forall i \in \mathsf{Hon}_0 :\; \pi_i \leftarrow \mathsf{PS.Prove}(\mathsf{crs},X_i,(x_i,r_i)).$$

Now, in $\mathbf{H}_2$, we generate them as

$$(\mathsf{crs},\mathsf{trap}) \leftarrow \mathsf{PS.TrapSetup}(1^\kappa),\quad \forall i \in \mathsf{Hon}_0 :\; \pi_i \leftarrow \mathsf{PS.Sim}(\mathsf{trap},X_i),$$

where $\mathsf{TrapSetup}$ and Sim are the trapdoor setup algorithm and zero-knowledge simulator guaranteed to exist by the zero-knowledge property. With this, we no longer need the witness (x_i,r_i) to generate the proofs. We can easily bound the difference between using a reduction to the zero-knowledge property of PS. The reduction gets crs from the game as input and simulates $\mathbf{H}_1$ until the proofs π_i have to be generated. Instead of generating them itself, the reduction outputs the statement-witness pairs $(X_i, (x_i, r_i))$ for $i \in \mathsf{Hon}_0$ to the zero-knowledge game and obtains the proofs π_i in return. It continues simulating $\mathbf{H}_1$ and outputs whatever the hybrid would output. Clearly, the reduction is PPT. Further, if $b = 0$ in the zero-knowledge game, then the reduction perfectly simulates $\mathbf{H}_1$, and if $b = 1$ it perfectly simulates $\mathbf{H}_2$. By our assumption that PS is zero-knowledge, we get

$$\left|\varepsilon_1^{\mathcal{A}} - \varepsilon_2^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_3$. We change how to generate element $h \in \mathbb{G}$. Namely, from now on the game samples $\vartheta \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and sets $h := g^{\vartheta}$. Note that $\mathbf{H}_2$ and $\mathbf{H}_3$ only differ if $h = g^0$ in $\mathbf{H}_2$. So we get

$$\left|\varepsilon_2^{\mathcal{A}} - \varepsilon_3^{\mathcal{A}}\right| \leq \frac{1}{p} \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_4$. In this hybrid, we change how the elements X_i are computed. Recall that before, they were computed as $X_i = g^{x_i} h^{r_i}$ for all $i \in \mathsf{Hon}_0$. Also, observe that by the change in $\mathbf{H}_2$, the game only needs (x_i, r_i) when party i is corrupted or to check the winning condition. In this hybrid, we sample $d_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and set $X_i := g^{d_i}$ for all $i \in \mathsf{Hon}_0$. When $\mathcal{A}$ calls the corruption oracle on input i and the has to return (x_i, r_i), it samples $x_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and computes $r_i := (d_i - x_i)/\vartheta$. Note that $\vartheta \neq 0$. Further, let Hon^* be the final set of honest parties when the winning condition is checked, i.e., $\mathsf{Hon}^* := [n] \setminus \mathsf{Corr}$ at the end of the game. For each $i \in \mathsf{Hon}^*$, sample $x_i \leftarrow\!\!\$\, \mathbb{Z}_p$ before checking the winning condition. Note that the joint distribution of the (X_i, x_i, r_i) does not change and so we get

$$\varepsilon_3^{\mathcal{A}} = \varepsilon_4^{\mathcal{A}}.$$

Let us make explicit what we have achieved so far: for every party $k \in \mathsf{Hon}^*$ that is not corrupted when the winning condition is checked, the game first samples $x_k \leftarrow\!\!\$\, \mathbb{Z}_p$, then defines

$$\mathsf{id}_k := x_k + \sum_{i \in \mathsf{ValidPre}^{(k)}} T_{i,k}^{(k)}$$

and then checks if $\mathsf{Pred}(\mathsf{id}_k) = 1$ to see if the party is in the committee. The game outputs 1 if no such party is in the committee. Note that all id_k for $k \in \mathsf{Hon}^*$ are uniform over $\mathbb{Z}_p$ and independent, because the x_k are. Further, we know that $|\mathsf{Hon}^*| = n - t$. We can therefore finish the proof with

$$\varepsilon_4^{\mathcal{A}} \leq \Pr[\mathsf{HonComm} = \emptyset] = \Pr\left[\forall k \in \mathsf{Hon}^* : \mathsf{Pred}(\mathsf{id}_k) = 0\right] = \Pr_{\nu}[\mathsf{Pred}(\nu) = 0]^{n-t} = \mathsf{bias}(\mathsf{Pred}, 0)^{n-t}.$$

□

3.4 Large Committee Game

The second property we prove is that the adversary cannot make the committee too large. This holds even if the adversary controls all parties and can choose tags for each party independently. Importantly, however, one of the tags for each party is chosen honestly by the game. We can for now think of this tag as being the tag output by the party that later verifies committee membership. Again, the state of $\mathcal{A}$ is kept implicit in the description of the game. More formally, the *large committee game* for a number of parties n and a committee bound B is defined as follows:

1. **Setup.** The game runs $\mathsf{par} \leftarrow \mathsf{CES.Setup}(1^{\kappa})$.

2. **Initialization of Parties.** The adversary declares all keys. More precisely:
   (a) The game runs $\mathcal{A}$ on input par.
   (b) Then, $\mathcal{A}$ outputs pk_k for all $k \in [n]$.
3. **Tag Phase.** In the tag phase, the tags are chosen:
   (a) The game runs $\mathsf{pretags}_1 \leftarrow \mathsf{CES.Prepare}(\mathsf{par})$ and sets $\mathsf{pretags}_{1,k} := \mathsf{pretags}_1$ for all $k \in [n]$.
   (b) The game continues running $\mathcal{A}$ on input $\mathsf{pretags}_1$.
   (c) The game obtains $\mathsf{pretags}_{i,k}$ for all $i \in [n] \setminus \{1\}$ and all $k \in [n]$ from $\mathcal{A}$. If the time between this output and the previous step is larger than Δ', the game terminates and outputs 0.
4. **Winning Condition.** The adversary wins if too many parties are elected. More precisely:
   (a) The game runs $\mathcal{A}$ and gets a set $\mathsf{Committee} \subseteq [n]$ and ticket_k for all $k \in \mathsf{Committee}$.
   (b) The game outputs 1 if the following two conditions hold. Otherwise, it outputs 0:
      i. We have $|\mathsf{Committee}| > B$.
      ii. For all $k \in \mathsf{Committee}$, we have $\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k, (\mathsf{pretags}_{i,k})_{i=1}^n, \mathsf{ticket}_k) = 1$.

We now show that no efficient adversary can win the large committee game, i.e., the committee is never too large.

Lemma 4. *Assume that* PS *and* $\tilde{\mathsf{PS}}$ *are simulation-extractable (see Definition 5), assume that the discrete logarithm assumption holds relative to* GGen*, and assume that* TLP *is* Δ*-CPA secure, where* Δ *is sufficiently larger than* $\mathbf{T}(\tilde{\mathsf{PS}}.\mathsf{Sim}) + \Delta' + n \cdot \mathbf{T}(\tilde{\mathsf{PS}}.\mathsf{Ext})$*. Let* $\mathcal{A}$ *be a PPT algorithm in the large committee game for a number of parties* $n < p$ *and a committee bound* B*. Then, the probability that the large committee game outputs* 1 *is at most* $\mathsf{negl}(\kappa) + \mathsf{tail}(P, n, B)$*.*

Proof. Consider a PPT adversary $\mathcal{A}$ running in the large committee game. Let $\varepsilon^{\mathcal{A}}$ be the probability that the large committee game outputs 1. We want to upper bound $\varepsilon^{\mathcal{A}}$. To do so, we provide a sequence of hybrid games $\mathbf{H}_0, \ldots, \mathbf{H}_{10}$. For each hybrid $\mathbf{H}_i$, we denote the probability that it outputs 1 when run with adversary $\mathcal{A}$ by $\varepsilon_i^{\mathcal{A}}$.

Hybrid $\mathbf{H}_0$. This is the large committee game. We recall the game to fix notation. Initially, the game sets up parameters par via algorithm $\mathsf{CES.Setup}$. To recall, par include the group $\mathbb{G}$ with generator g, an element $h \leftarrow\$ \mathbb{G}$, time-lock parameters tlpar and common reference strings crs and $\tilde{\mathsf{crs}}$ for PS and $\tilde{\mathsf{PS}}$, respectively. In the *initialization of parties phase*, the adversary gets par as input and declares all public keys $\mathsf{pk}_k = (X_k, \pi_k)$ for all $k \in [n]$, where $X_k \in \mathbb{G}$ and π_k is a proof. In the *tag phase*, the game generates a pretag $\mathsf{pretags}_1 = (Z_1, (c_{1,j})_{j=1}^n, \tilde{\pi}_1)$ via algorithm $\mathsf{CES.Prepare}$. Precisely, by definition of $\mathsf{CES.Prepare}$, these values are computed by sampling $T_{1,j} \leftarrow\$ \mathbb{Z}_p$, $\gamma_{1,j} \leftarrow\$ \mathbb{Z}_p$, setting $c_{1,j} := g^{T_{i,j}} h^{\gamma_{i,j}}$ for every $j \in [n]$, and computing the puzzle $Z_1 := \mathsf{TLP.Gen}(\mathsf{tlpar}, (T_{i,j}, \gamma_{i,j})_{j=1}^n; \rho_i)$ and the proof $\tilde{\pi}_i \leftarrow \tilde{\mathsf{PS}}.\mathsf{Prove}(\tilde{\mathsf{crs}}, (Z_1, (c_{1,j})_{j=1}^n), ((T_{1,j}, \gamma_{1,j})_{j=1}^n, \rho_1))$ for some random coins ρ_1. The game also sets $\mathsf{pretags}_{1,k} := \mathsf{pretags}_1$ for all $k \in [n]$, and we set $(Z_{1,k}, (c_{1,j,k})_{j=1}^n, \tilde{\pi}_{1,k}) := (Z_1, (c_{1,j})_{j=1}^n, \tilde{\pi}_1)$ accordingly. The adversary $\mathcal{A}$ then gets $\mathsf{pretags}_1$ and within time Δ', it has to output $\mathsf{pretags}_{i,k}$ for all $i \in [n] \setminus \{1\}$ and all $k \in [n]$, where we write $\mathsf{pretags}_{i,k} = (Z_{i,k}, (c_{i,j,k})_{j=1}^n, \tilde{\pi}_{i,k})$. To evaluate the *winning condition*, the game continues running $\mathcal{A}$ until it outputs a set $\mathsf{Committee} \subseteq [n]$ and tickets ticket_k for all $k \in \mathsf{Committee}$. The game then outputs 1 if and only if $|\mathsf{Committee}| > B$ and $\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k, (\mathsf{pretags}_{i,k})_{i=1}^n, \mathsf{ticket}_k) = 1$ for all $k \in \mathsf{Committee}$. Let us make explicit how $\mathsf{CES.VerElect}$ works here for such a $k \in \mathsf{Committee}$: The algorithm defines the set

$$\mathsf{ValidPre}_k := \left\{ i \in [n] \;\middle|\; \tilde{\mathsf{PS}}.\mathsf{Ver}(\tilde{\mathsf{crs}}, (Z_{i,k}, (c_{i,j,k})_{j=1}^n), \tilde{\pi}_{i,k}) = 1 \right\}.$$

It parses $\mathsf{ticket}_k = (x_k, r_k, (T_{i,k}, \gamma_{i,k})_{i \in \mathsf{ValidPre}_k})$. Then, it outputs 1 if and only if the following four conditions hold:

1. We have $\mathsf{PS.Ver}(\mathsf{crs}, X_k, \pi_k) = 1$.
2. We have $g^{x_k} h^{r_k} = X_k$.
3. For every $i \in \mathsf{ValidPre}_k$, we have $g^{T_{i,k}} h^{\gamma_{i,k}} = c_{i,k,k}$.
4. We have $\mathsf{Pred}(\mathsf{id}_k) = 1$ for $\mathsf{id}_k = x_k + \sum_{i \in \mathsf{ValidPre}_k} T_{i,k}$.

By definition, we get

$$\varepsilon^{\mathcal{A}} = \varepsilon_0^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_1$. We change how the game generates crs. Namely, while it has been generated as $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\kappa)$ in the previous hybrid, from now on it is generated with a trapdoor as $(\mathsf{crs}, \mathsf{trap}) \leftarrow \mathsf{PS.TrapSetup}(1^\kappa)$. We can easily bound the difference between $\mathbf{H}_0$ and $\mathbf{H}_1$ using the zero-knowledge property of PS, see Definition 4. We omit giving the reduction formally. We get

$$\left|\varepsilon_0^{\mathcal{A}} - \varepsilon_1^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_2$. We add a step to the game after the *initialization of parties phase*. To understand this additional step, first recall that in the *initialization of parties phase*, the adversary $\mathcal{A}$ outputs keys $\mathsf{pk}_k = (X_k, \pi_k)$ where $X_k \in \mathbb{G}$ and π_k is a proof for all $k \in [n]$. Here is the additional step that we add in $\mathbf{H}_2$: the game first defines the set[7]

$$\mathsf{Active} := \{k \in [n] \mid \mathsf{PS.Ver}(\mathsf{crs}, X_k, \pi_k) = 1\}.$$

It then extracts witnesses from the proofs π_k for $k \in \mathsf{Active}$, i.e.,

$$(\hat{x}_k, \hat{r}_k) \leftarrow \mathsf{PS.Ext}(\mathsf{trap}, X_k, \pi_k) \text{ for all } k \in \mathsf{Active},$$

where trap is the trapdoor introduced in $\mathbf{H}_1$. Further, the game aborts if $X_k \neq g^{\hat{x}_k} h^{\hat{r}_k}$ for some $k \in \mathsf{Active}$. The rest of the game remains unchanged. Note that the two hybrids only differ if the game aborts, and if the game aborts a straightforward reduction can break simulation-extractability (see Definition 5) of PS. Therefore, we get

$$\left|\varepsilon_1^{\mathcal{A}} - \varepsilon_2^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_3$. We change how $\tilde{\mathsf{crs}}$ and the proof $\tilde{\pi}_1$ contained in $\mathsf{pretags}_1 = (Z_1, (c_{1,j})_{j=1}^n, \tilde{\pi}_1)$ (as computed in the *tag phase*) are generated. Recall that until now they have been generated as

$$\tilde{\mathsf{crs}} \leftarrow \tilde{\mathsf{PS}}.\mathsf{Setup}(1^\kappa), \quad \tilde{\pi}_1 \leftarrow \tilde{\mathsf{PS}}.\mathsf{Prove}(\tilde{\mathsf{crs}}, (Z_1, (c_{1,j})_{j=1}^n), ((T_{1,j}, \gamma_{1,j})_{j=1}^n, \rho_1)).$$

[7] Note that every party that is in the committee has to be in Active, by definition of algorithm $\mathsf{CES.VerElect}$.

From now on, we generate them using the trapdoor setup algorithm $\tilde{\mathsf{PS}}.\mathsf{TrapSetup}$ and the zero-knowledge simulator $\tilde{\mathsf{PS}}.\mathsf{Sim}$ as

$$(\tilde{\mathsf{crs}}, \tilde{\mathsf{trap}}) \leftarrow \tilde{\mathsf{PS}}.\mathsf{TrapSetup}(1^\kappa), \quad \tilde{\pi}_1 \leftarrow \tilde{\mathsf{PS}}.\mathsf{Sim}(\tilde{\mathsf{trap}}, (Z_1, (c_{1,j})_{j=1}^n)).$$

We can bound the distinguishing advantage of $\mathcal{A}$ between this hybrid and the previous hybrid using the zero-knowledge property of $\tilde{\mathsf{PS}}$, see Definition 4. The formal reduction is trivial and omitted. We get

$$\left|\varepsilon_2^{\mathcal{A}} - \varepsilon_3^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_4$. In this hybrid, we use the knowledge extractor $\tilde{\mathsf{PS}}.\mathsf{Ext}$ of $\tilde{\mathsf{PS}}$ to extract the tags provided by the adversary. We now make this change more precise: recall that in the *tag phase*, the game obtains pretags $\mathsf{pretags}_{i,k} = (Z_{i,k}, (c_{i,j,k})_{j=1}^n, \tilde{\pi}_{i,k})$ for all $i \in [n] \setminus \{1\}$ and all $k \in [n]$. Here are the additional step that we add in $\mathbf{H}_4$, after receiving these pretags:

For each $k \in [n]$, do the following:

1. Define the set $\mathsf{ValidPre}_k := \{i \in [n] \mid \tilde{\mathsf{PS}}.\mathsf{Ver}(\tilde{\mathsf{crs}}, (Z_{i,k}, (c_{i,j,k})_{j=1}^n), \tilde{\pi}_{i,k}) = 1\}$. Note that this is exactly as algorithm $\mathsf{CES.VerElect}$ will define this set when the game will evaluate the winning condition, see $\mathbf{H}_0$.
2. Define the set $\mathsf{Copied}_k := \{i \in \mathsf{ValidPre}_k \mid (Z_{i,k}, (c_{i,j,k})_{j=1}^n) = (Z_1, (c_{1,j})_{j=1}^n)\}$ and note that $1 \in \mathsf{Copied}_k$.
3. For all $i \in \mathsf{Copied}_k$, set $\hat{T}_{i,k,k} = T_{1,k}$.
4. For all $i \in \mathsf{ValidPre}_k \setminus \mathsf{Copied}_k$, run
$$((\hat{T}_{i,j,k}, \hat{\gamma}_{i,j,k})_{j=1}^n, \hat{\rho}_{i,k}) \leftarrow \mathsf{PS.Ext}(\tilde{\mathsf{trap}}, (Z_{i,k}, (c_{i,j,k})_{j=1}^n), \tilde{\pi}_{i,k}).$$
5. If there is an $i \in \mathsf{ValidPre}_k$ such that $g^{\hat{T}_{i,k,k}} h^{\hat{\gamma}_{i,k,k}} \neq c_{i,k,k}$, abort the game.

Intuitively, this means the game extracts all commitment preimages (i.e., tags) for pretags that have valid proofs and have not been copied by the adversary. The rest of the game remains unchanged for now. Clearly, the output of this hybrid only differs from the previous one if the abort happens. For each fixed $k^* \in [n]$ and $i^* \in [n] \setminus \{1\}$, we bound the probability that $i^* \in \mathsf{ValidPre}_{k^*} \setminus \mathsf{Copied}_{k^*}$ and the abort happens for this pair (k^*, i^*). To this end, we using the simulation-extractability of $\tilde{\mathsf{PS}}$ via the following reduction:

1. The reduction gets $\tilde{\mathsf{crs}}$ as input. It runs the game in $\mathbf{H}_3$ until it has to provide the proof $\tilde{\pi}_1$ to the adversary.

2. To provide $\tilde{\pi}_1$, it outputs the statement $(Z_1, (c_{1,j})_{j=1}^n)$ to the simulation-extractability game. It obtains a simulated proof $\tilde{\pi}_1$ in return.

3. The reduction continues simulating $\mathbf{H}_3$ until in the *tag phase* $\mathcal{A}$ outputs $\mathsf{pretags}_{i,k}$ for all $i \in [n] \setminus \{1\}$ and all $k \in [n]$. It defines $\mathsf{ValidPre}_{k^*}$ and Copied_{k^*} as above and aborts if $i^* \notin \mathsf{ValidPre}_{k^*}$ or $i^* \in \mathsf{Copied}_{k^*}$. Otherwise, it outputs $(Z_{i^*,k^*}, (c_{i^*,j,k^*})_{j=1}^n)$ and $\tilde{\pi}_{i^*,k^*}$ and terminates.

If the abort happens for (k^*, i^*), the reduction breaks simulation-extractability. Further, the reduction is PPT. With a union bound over all pairs pair (k^*, i^*), we get

$$\left|\varepsilon_3^{\mathcal{A}} - \varepsilon_4^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_5$. We change how the winning condition is evaluated: concretely, recall that to evaluate the winning condition, the game obtains a set $\mathsf{Committee} \subseteq [n]$ and tickets of the form $\mathsf{ticket}_k = (x_k, r_k, (T_{i,k}, \gamma_{i,k})_{i \in \mathsf{ValidPre}_k})$ for all $k \in \mathsf{Committee}$ from the adversary. Then, it checks the winning condition as explained in $\mathbf{H}_0$. Especially, it defines

$$\mathsf{id}_k = x_k + \sum_{i \in \mathsf{ValidPre}_k} T_{i,k} \quad \text{for every } k \in \mathsf{Committee}.$$

In $\mathbf{H}_5$, we instead define them as

$$\mathsf{id}_k = \hat{x}_k + \sum_{i \in \mathsf{ValidPre}_k} \hat{T}_{i,k,k} \quad \text{for every } k \in \mathsf{Committee}.$$

To recall, the $\hat{x}_k$ have been extracted in hybrid $\mathbf{H}_2$ and the $\hat{T}_{i,k,k}$ have been extracted in hybrid $\mathbf{H}_4$. We argue that if the outputs of $\mathbf{H}_4$ and $\mathbf{H}_5$ differ, then we can break the discrete logarithm assumption via an efficient reduction. To see this, note that if the outputs differ, then there has to be some $k \in \mathsf{Committee}$ such that:

1. $\hat{x}_k \neq x_k$ or $\hat{T}_{i,k,k} \neq T_{i,k}$ for some $i \in \mathsf{ValidPre}_k$, because the outputs differ.
2. $g^{x_k} h^{r_k} = X_k$ and $g^{T_{i,k}} h^{\gamma_{i,k}} = c_{i,k,k}$ for all $i \in \mathsf{ValidPre}_k$, due to the winning condition, see $\mathbf{H}_0$.
3. $g^{\hat{x}_k} h^{\hat{r}_k} = X_k$ and $g^{\hat{T}_{i,k,k}} h^{\hat{\gamma}_{i,k,k}} = c_{i,k,k}$ for all $i \in \mathsf{ValidPre}_k$, due to the aborts in $\mathbf{H}_2$ and $\mathbf{H}_4$.

From this, it is easy to see that an efficient reduction can find the discrete logarithm of h with respect to basis g. By the discrete logarithm assumption, we get

$$\left|\varepsilon_4^{\mathcal{A}} - \varepsilon_5^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_6$. We change the winning condition making it easier for $\mathcal{A}$ to win. Concretely, as part of algorithm $\mathsf{CES.VerElect}$, the game now no longer checks that $g^{x_k} h^{r_k} = X_k$ and $g^{T_{i,k}} h^{\gamma_{i,k}} = c_{i,k,k}$ for every $i \in \mathsf{ValidPre}_k$. What remains are the checks that $k \in \mathsf{Active}$ and that $\mathsf{Pred}(\mathsf{id}_k) = 1$, where id_k is defined as in the previous hybrid. Note that $\mathbf{H}_5$ and $\mathbf{H}_6$ are *not indistinguishable*, but one can easily see that if $\mathcal{A}$ wins in $\mathbf{H}_5$, then it also wins in $\mathbf{H}_6$ with at least the same probability. Thus, we get

$$\varepsilon_5^{\mathcal{A}} \leq \varepsilon_6^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_7$. We change the winning condition again. Recall that until now, game outputs one if $|\mathsf{Committee}| > B$, $\mathsf{Committee} \subseteq \mathsf{Active}$, and $\mathsf{Pred}(\mathsf{id}_k) = 1$ for all $k \in \mathsf{Committee}$. We now make it easier for the adversary: the game outputs 1 if $|\mathsf{Committee}^*| > B$ for

$$\mathsf{Committee}^* := \{k \in \mathsf{Active} \mid \mathsf{Pred}(\mathsf{id}_k) = 1\},$$

Again, these two hybrids are *not indistinguishable*, but it is easy to see that

$$\varepsilon_6^{\mathcal{A}} \leq \varepsilon_7^{\mathcal{A}}.$$

Note what we have now achieved: the winning condition of the game can be evaluated as soon as the adversary has sent its puzzles and the game extracted the tags $\hat{T}_{i,j,k}$ as described in hybrid $\mathbf{H}_4$. This is because the winning condition does not depend on the final output of the adversary any more. Also, recall that the adversary has to send its puzzles in time Δ' after receiving puzzle Z_1. Thus, hybrid $\mathbf{H}_7$ can be run in time

$$\mathbf{T}(\tilde{\mathsf{PS}}.\mathsf{Sim}) + \Delta' + n \cdot \mathbf{T}(\tilde{\mathsf{PS}}.\mathsf{Ext}) + t \leq \Delta$$

after puzzle Z_1 is defined, where t is the negligible amount of time it takes to define the sets $\mathsf{ValidPre}_k$ and Copied_k, and to compute the size of $\mathsf{Committee}^*$ given all relevant $\hat{x}_k$ and $\hat{T}_{i,k,k}$. Our next goal is to remove all information the adversary gets about the tags $T_{1,j}$.

Hybrid $\mathbf{H}_8$. We change how the game computes the puzzle Z_1. While so far the game has generated

$$Z_1 := \mathsf{TLP.Gen}(\mathsf{tlpar}, (T_{1,j}, \gamma_{1,j})_{j=1}^n; \rho_1),$$

we now generate it as

$$Z_1 \leftarrow \mathsf{TLP.Gen}(\mathsf{tlpar}, \mathbf{0}),$$

were $\mathbf{0}$ is an all-zero string of appropriate length. We can bound the difference between hybrids $\mathbf{H}_7$ and $\mathbf{H}_8$ using the Δ-CPA security of TLP: a reduction would get tlpar from the Δ-CPA game and simulate $\mathbf{H}_7$ for the adversary. To define Z_1, the reduction would output $(T_{1,j}, \gamma_{1,j})_{j=1}^n$ and $\mathbf{0}$ to the Δ-CPA game and get Z_1 in return. Then, it would evaluate the winning condition and return the result to the Δ-CPA game. This works because (1) we do not use ρ_1 elsewhere in $\mathbf{H}_7$, due to the change introduced in hybrid $\mathbf{H}_3$, and (2) the online phase of the reduction, i.e., the time between receiving Z_1 and outputting the output of the game takes time at most Δ, as explained above. By the Δ-CPA security of TLP, we get

$$\left|\varepsilon_7^{\mathcal{A}} - \varepsilon_8^{\mathcal{A}}\right| \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_9$. We change the commitments $c_{1,j}$ for $j \in [n]$ that the game sends in the *tag phase*. So far, these commitments have been generated as

$$\gamma_{1,j} \leftarrow_\$ \mathbb{Z}_p,\ c_{1,j} := g^{T_{1,j}} h^{\gamma_{1,j}} \text{ for all } j \in [n].$$

From now on, the game samples them at random, i.e., $c_{1,j} \leftarrow_\$ \mathbb{G}$ for all $j \in [n]$. Note that due to the change in $\mathbf{H}_3$ and the previous hybrid, the values $\gamma_{1,j}$ are only used to generate these commitments and not elsewhere. Hence, assuming h is a generator of $\mathbb{G}$, the distribution does not change. We get

$$\left|\varepsilon_8^{\mathcal{A}} - \varepsilon_9^{\mathcal{A}}\right| \leq \frac{1}{p} \leq \mathsf{negl}(\kappa).$$

Hybrid $\mathbf{H}_{10}$. We change id_k to random. Namely, while id_k has been computed as $\mathsf{id}_k = \hat{x}_k + \sum_{i \in \mathsf{ValidPre}_k} \hat{T}_{i,k,k}$ for every $k \in \mathsf{Active}$ until now, this hybrid instead samples $\mathsf{id}_k \leftarrow_\$ \mathbb{Z}_p$ for all $k \in \mathsf{Active}$. We claim that this is a purely conceptual change, i.e., the values id_k are already uniform and independent in $\mathbf{H}_9$. To see this, recall the definition of the sets $\mathsf{Copied}_k \subseteq \mathsf{ValidPre}_k$ from $\mathbf{H}_4$ and let $\Gamma_k := |\mathsf{Copied}_k|$ for every $k \in \mathsf{Active}$. Then, in $\mathbf{H}_9$, we get

$$\mathsf{id}_k = \hat{x}_k + \sum_{i \in \mathsf{ValidPre}_k} \hat{T}_{i,k,k} = \hat{x}_k + \Gamma_k \cdot T_{1,k} + \sum_{i \in \mathsf{ValidPre}_k \setminus \mathsf{Copied}_k} \hat{T}_{i,k,k}.$$

By the changes we have made in hybrids $\mathbf{H}_3, \mathbf{H}_8$, and $\mathbf{H}_9$, the adversary obtains no information about the $T_{1,k}$'s and therefore $\hat{x}_k$, Γ_k, and all $\hat{T}_{i,k,k}$ for $i \in \mathsf{ValidPre}_k \setminus \mathsf{Copied}_k$ are independent of $T_{1,k}$. In $\mathbb{Z}_p$, $\Gamma^{(k)}$ is invertible as $1 \leq \Gamma_k < p$, so all id_k's are uniform and independent.

Finally, we bound the probability $\varepsilon_{10}^{\mathcal{A}}$: observe that the game in hybrid $\mathbf{H}_{10}$ outputs 1 if $|\mathsf{Committee}^*| > B$ (see $\mathbf{H}_7$), i.e., there are more than B parties k in Active such that $\mathsf{Pred}(\mathsf{id}_k) = 1$. Due to the changes in the previous hybrids, all id_k are uniform and independent. Hence, we can conclude the proof with

$$\varepsilon_{10}^{\mathcal{A}} \leq \mathsf{tail}(P, n, B).$$

$\square$

Remark 1. *For conciseness, from now on we hide the terms depending on the proof system and on computations independent of the time-lock puzzle by using the notation*

$$\Delta(\Delta') := \mathbf{T}(\widetilde{\mathsf{PS}}.\mathsf{Sim}) + \Delta' + n \cdot \mathbf{T}(\widetilde{\mathsf{PS}}.\mathsf{Ext}) + t, \tag{2}$$

where t is the negligible amount of time it takes to define the sets $\mathsf{ValidPre}_k$ and Copied_k, and to compute the size of $\mathsf{Committee}^*$ given all relevant $\hat{x}_k$ and $\hat{T}_{i,k,k}$, as in the proof of Lemma 4. In particular, this shifts the focus on needing to define Δ', which is the time by which the adversary needs to submit all its pretags.

We also note that $\mathbf{T}(\widetilde{\mathsf{PS}}.\mathsf{Sim})$ and $\mathbf{T}(\widetilde{\mathsf{PS}}.\mathsf{Ext})$ are independent of a round duration Δ_r. The same holds for the additional time t above. As such, we have that $(\Delta(\Delta') - \Delta')/\Delta_r = O(1)$, i.e., the extra time apart from Δ' in (2) takes a constant number of rounds. Looking ahead, we will select Δ' to take a number of rounds that is sublinear in n, such that an honest party solving a TLP with difficulty $\Delta(\Delta')$ will be able to obtain the solution in a sublinear number of rounds.

3.5 Predicate Instantiation

We now instantiate the predicate $\mathsf{Pred}\colon \mathbb{Z}_p \to \{0,1\}$ and the committee bound B. To use the previous two lemmata in a meaningful way, we need that $\mathsf{tail}(\mathsf{Pred}, n, B)$ and $\mathsf{bias}(\mathsf{Pred}, 0)^{n-t}$ are negligible in λ. We let $\mathsf{Pred}(\nu)$ output 1 if and only if $\nu < \mathsf{bound}_{\epsilon,\delta}$, where

$$p_{\mathrm{mine}} := \min\left\{1, \frac{1}{\epsilon n}\log\left(\frac{1}{\delta}\right)\right\}, \quad \mathsf{bound}_{\epsilon,\delta} := p_{\mathrm{mine}} \cdot p.$$

Here, p denotes the size of the field $\mathbb{Z}_p$, p_{mine} roughly corresponds to the probability of a party being elected, ϵ is a constant in $(0,1)$ denoting the fraction of honest parties (i.e. $t = (1-\epsilon)n$), and δ is a failure probability. Below, we will focus on the case where $\delta = \exp(-\omega(\log\lambda))$ and $\log(\delta^{-1}) \leq \epsilon n$.[8] Further, we define the committee bound B as

$$B := \left\lceil \frac{3}{\epsilon}\log\left(\frac{1}{\delta}\right)\right\rceil = O(\lambda/\epsilon).$$

For this choice of parameters, we show that $\mathsf{tail}(P, n, B)$ and $\mathsf{bias}(P, 0)^{n-t}$ are both negligible in the security parameter.

Lemma 5. *Consider the predicate* Pred *defined as above. Then, it holds that*

$$\mathsf{bias}(\mathsf{Pred}, 0)^{n-t} \leq \delta \leq \mathsf{negl}(\lambda).$$

Proof. By definition, we have $\mathsf{bias}(\mathsf{Pred}, 0) = 1 - p_{\mathrm{mine}}$. If $p_{\mathrm{mine}} = 1$, then $\mathsf{bias}(\mathsf{Pred}, 0) = 0$, so the statement holds trivially. We now focus on the other case. From Bernoulli's inequality (Lemma 1), we get

$$\mathsf{bias}(\mathsf{Pred}, 0)^{n-t} = (1 - p_{\mathrm{mine}})^{\epsilon\cdot n} \leq \exp(-\epsilon n \cdot p_{\mathrm{mine}}) = \exp(-\log(\delta^{-1})) = \delta,$$

where the first inequality follows from Bernoulli's inequality. $\square$

[8] I.e., pick δ such that $\delta \geq \exp(-\epsilon n)$ and $\delta = \exp(-\omega(\log\lambda))$, which can be done if $\epsilon n > \log\lambda$.

Lemma 6. *Consider the predicate* Pred *and the committee bound* B *as above. Then, it holds that*

$$\mathsf{tail}(\mathsf{Pred}, n, B) \leq \delta \leq \mathsf{negl}(\lambda).$$

Proof. Let $X_i, i \in [n]$ be Bernoulli random variables, denoting the event that $\mathsf{Pred}(\mathsf{id}_i) = 1$. Since id_i are chosen independently and uniformly, the X_i's are independent and identically distributed with $\Pr[X_i = 1] = p_{\text{mine}}$. In that notation, $\mathsf{tail}(\mathsf{Pred}, n, B) = \Pr[\sum_{i\in[n]} X_i > B]$. The expected value of $X := \sum_{i\in[n]} X_i$ is $\mathbb{E}[X] = n \cdot p_{\text{mine}}$.

$$\begin{aligned}\mathbb{E}[X] &= n \cdot p_{\text{mine}} = n \cdot \min\left\{1, \frac{1}{\epsilon n} \log\left(\delta^{-1}\right)\right\} \\ &= \begin{cases} n & \text{if } \log\left(\delta^{-1}\right) > \epsilon n \\ \frac{1}{\epsilon}\log\left(\delta^{-1}\right) & \text{if } \log\left(\delta^{-1}\right) \leq \epsilon n \end{cases}.\end{aligned}$$

The first case is not interesting, since it corresponds to the case where the committee is as large as the total number of parties. This motivates our assumption of δ above, namely $\log(\delta^{-1}) \leq \epsilon n$, so we are in the second case. We continue with Chernoff's inequality (Lemma 2).

$$\begin{aligned}\Pr[X > (1+\zeta)\cdot\mathbb{E}[X]] \leq \exp\left(-\frac{\zeta^2\cdot\mathbb{E}[X]}{2+\zeta}\right) &= \exp\left(-\frac{\zeta^2}{2+\zeta}\cdot\frac{1}{\epsilon}\cdot\log\left(\frac{1}{\delta}\right)\right) \\ &\overset{*}{\leq} \exp\left(-\log\left(\delta^{-1}\right)\right) = \delta,\end{aligned}$$

where $*$ holds if

$$\frac{\zeta^2}{2+\zeta}\cdot\frac{1}{\epsilon} \geq 1. \tag{3}$$

We want to pick a ζ that satisfies (3) for any value of $\epsilon \in (0,1)$ and from it, define $B_0 := (1+\zeta)\cdot\mathbb{E}[X]$, such that $\Pr[X \geq B_0] \leq \delta$.

The roots of (3) are $(\epsilon - \sqrt{\epsilon^2+8\epsilon})/2$ and $(\epsilon + \sqrt{\epsilon^2+8\epsilon})/2$ and inequality holds outside of the roots. Since $\zeta \geq 0$, we only need $\zeta \geq (\epsilon + \sqrt{\epsilon^2+8\epsilon})/2$. To account for any value of $\epsilon \in (0,1)$, we choose to set

$$\begin{aligned}B_0 &= \left(1 + \frac{\epsilon+\sqrt{\epsilon^2+8\epsilon}}{2}\right)\cdot\mathbb{E}[X] = \frac{2+\epsilon+\sqrt{\epsilon^2+8\epsilon}}{2}\cdot\frac{1}{\epsilon}\cdot\log\left(\frac{1}{\delta}\right) \\ &= \frac{2+\epsilon+\sqrt{\epsilon^2+8\epsilon}}{2\epsilon}\cdot\log\left(\frac{1}{\delta}\right) \leq \frac{6}{2\epsilon}\cdot\log\left(\frac{1}{\delta}\right) = \frac{3}{\epsilon}\cdot\log\left(\frac{1}{\delta}\right).\end{aligned}$$

Since it holds that for $B \geq B_0$, $\Pr[X > B_0] \leq \Pr[X > B] \leq \delta$, we can choose an integer B from the maximum value of B_0. Thus, we get $B = \lceil 3\log\left(\delta^{-1}\right)/\epsilon\rceil$, obtaining the value of B in the statement. □

4 Graded Committee Election

In this section, we introduce a committee election scheme that also accounts for inconsistent views between parties. This committee election scheme will be composed of both algorithms and protocols. The main difficulty of adapting the algorithms from Section 3 into protocols stems from the inconsistent views of the committee of different honest parties. In more detail, the reader may recall from the previous section that a party can decide if it is in the committee based on an "identity", composed of its secret key and a set of so-called pretags. Other parties can then be convinced of the committee membership using a ticket provided by the party claiming membership. However, in Section 3, we assumed that the winning party and the verifying party agree on the set of pretags and on the winning party's public key. In this section, we show how to achieve this. The challenge in doing so is to deal with inconsistencies and to ensure that the process of distributing keys and pretags terminates.

Informally, in our proposed solution, parties will share their pretags and keys via several applications of a gradecast protocol. Recall that at the end of a gradecast protocol, parties hold an output they think corresponds to the sender's input as well as a grade associated to this output. Importantly, the grades that honest parties hold at the end of gradecast will quantify how much confidence they have that their outputs coincide. In particular, grades greater than 1 certify that honest parties have the same view on the input pretags and hence, on the identities used for election.

Throughout, we call our solution a graded committee election scheme (GCES). It consists of the GCES.Gen protocol in Figure 1, which is a gradecast on the public keys, and the GCES.Toss protocol in Figure 2, which can be seen as a parallel moderated gradecast on the pretags. The final part of the graded committee election consists parties locally running CES.TryElect and CES.VerElect on appropriate inputs.

4.1 Graded Consensus on Pretags and Keys

We give a step-by-step description of the protocols, beginning with a strawman construction.

Strawman Construction. In this strawman construction, parties engage in a parallel gradecast protocol where each party inputs their pretags. At the end of the gradecast protocol, each party P_i outputs for each party P_j a pretag and a grade $(\mathsf{pretags}_j^{(i)}, g_j^{(i)})$. Here, we use the superscript to denote the receiving party and the subscript to denote the sending party. This is sufficient for party P_i to be able to run CES.TryElect. As a result, P_i may hold an evidence of election in the form of ticket_i. Let us now explore how other parties can verify that P_i is part of the committee. Party P_i will multicast its ticket. A party P_j will retrieve from memory its result of the parallel gradecasts and check if the pretags it has corresponding to P_i are also consistent with the ticket multicast by P_i. In other words, it would run CES.VerElect, ignoring for now the input corresponding to P_i's public key. To see why this strawman solution fails, assume that both P_i and P_j are honest and there is a malicious party P_k not following the protocol. If there is any pretag from P_k for which, at the end of the gradecast, P_i outputs grade 1 and P_j outputs grade 0, then P_j will not agree with the committee election claim of P_i. In other words, when using a single instance of parallel gradecasts, the adversary has full power in causing honest parties to disagree in their views of the committee membership.

Distributing Pretags using Moderated Gradecast. To address this, we update the strawman construction with a second step of parallel gradecasts where parties *moderate* the values they received in the first step. The final set of pretags of a moderator P_j from the point of view of a party P_i will be thus made up of pretags that P_j gradecast itself in the first step, but also of pretags P_j gradecasts from other parties in the second step. Moreover, moderators will have their final grade penalized by the difference in the outputs of their moderated gradecast and the initial gradecast. This ensures that a moderator who honestly gradecasts a value sent by a malicious initial sender will not be penalized by more than 1 in its final grade. Thus, malicious parties with positive grades cannot arbitrarily cause two honest parties to disagree on their views of the committee membership. We call this protocol GCES.Toss and detail it in Figure 2.

Distributing Keys using Gradecast. The verification of election outcome does not only depend on the pretags and on the ticket of a party claiming membership, but also on a public key shared by that party in advance of sharing the pretags, as specified in $\mathsf{CES.VerElect}$. Therefore, honest parties need to also account for inconsistencies in their views of the adversary's public keys, which are critical in validating the result of the election. To this end, we specify that parties have to also run a gradecast protocol for their public keys as an initial step, that we call $\mathsf{GCES.Gen}$, see Figure 1. Note that here, a single gradecast is sufficient for sharing the public keys, since each public key describes a quantity originating from a single party and is used only in the election (and verification) of *that* party. In contrast, the pretags from each party are used to determine the election of each *other* party, i.e., parties are elected based on an aggregated string that depends on quantities originating from all parties, and, as described previously, require more steps to build trust in the output. For both $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, parties first read the system parameters $\mathsf{par} \leftarrow \mathsf{CES.Setup}(1^\kappa)$. In our instantiation, these parameters are transparent.

Graded Election and Verification. After a party P_i runs $\mathsf{GCES.Gen}$ and (the first two steps of) $\mathsf{GCES.Toss}$, it can determine if it is part of the committee in the following manner. In particular, P_i will have the secret key sk_i and $(\mathsf{pretags}^{(i)}_{j,i})_{j\in[n]} =: (\mathsf{pretags}^{(i)}_{j})_{j\in[n]}$, where for $\mathsf{pretags}$ the notation is as follows: the superscript denotes the receiving party P_i, and the subscripts denote the original sender P_j and the moderator P_i in this order (or only the original sender P_j when there is a single subscript). Then, each party P_i will run $\mathsf{CES.TryElect}$ from Section 3 on input $(\mathsf{par}, \mathsf{sk}_i, (\mathsf{pretags}^{(i)}_{j})_{j\in[n]})$ and will output $(\mathsf{res}, \mathsf{ticket})$, where if $\mathsf{res} = 0$, then $\mathsf{ticket} = \bot$, else $\mathsf{res} = 1$ and ticket contains the evidence of election.

Note that the results from $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$ are not necessarily identical for honest parties with respect to the party they want to verify membership for. To address this, a ticket containing the secret key and the tags obtained from solving the puzzle need to be multicast by each party claiming election.[9] Here, multicasting the ticket is sufficient (in contrast to requiring gradecast), because honest parties already have the necessary commitments and proofs which they can use to validate the received ticket. Therefore, parties use $\mathsf{CES.VerElect}$ from Section 3 to verify the election result claimed by another party. Honest party P_i will have the graded outputs of $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, and will be able to check the validity of P_j's claimed membership upon receiving $\mathsf{ticket}^{(i)}_j$ from P_j. Concretely, P_i will run $\mathsf{CES.VerElect}$ on input $(\mathsf{par}, \mathsf{pk}^{(i)}_j, (\mathsf{pretags}^{(i)}_{k,j})^n_{k=1}, \mathsf{ticket}^{(i)}_j)$ and output a bit $\mathsf{res} \in \{0, 1\}$, where $\mathsf{res} = 1$ if P_j is a member of the committee from P_i's perspective and $\mathsf{res} = 0$ otherwise.[10]

Details on Gradecast. In our constructions, we use the GC protocol for gradecast, described in [21], with a maximum grade g^*. This GC protocol takes $2g^* + 1$ rounds and has $O(g^*(\kappa + \ell)n^2)$ communication complexity for input messages of size ℓ. In the gradecast protocol instantiation, each message sent by a party in a round also carries its signature. In $\mathsf{GCES.Toss}$, we want to ensure that when a party moderates a puzzle of a different party, it does not moderate a different puzzle than the one it truly received. To this end, each puzzle gradecast by party P_i also carries the signature of P_i. In other words, in Step 1 in $\mathsf{GCES.Toss}$, P_i gradecasts the message $(\mathsf{pretags}_i, \mathsf{Sig}_i(\mathsf{pretags}_i))$, and in Step 2, P_i gradecasts the message $(m^{(i)}_j, \mathsf{Sig}_j(m^{(i)}_j)) := ((\mathsf{pretags}^{(i)}_j, \mathsf{Sig}_j(\mathsf{pretags}^{(i)}_j)), \mathsf{Sig}_i((\mathsf{pretags}^{(i)}_j, \mathsf{Sig}_j(\mathsf{pretags}^{(i)}_j)))$. To not burden the notation, we do not explicitly write all these signatures everywhere.

[9] As mentioned in Section 3, we aim for each party to only have to solve a single batch puzzle, namely, its own (done in $\mathsf{CES.TryElect}$). Otherwise, given that solving a batch puzzle is a sequential action, evaluating n distinct batch puzzles would automatically refute the claim of a sublinear number of rounds. Multicasting the ticket which contains the solved puzzle thus resolves this issue.

[10] Note that P_i does not need to check here again whether $\mathsf{pretags}^{(i)}_{i,j} = \mathsf{pretags}_i$; we will only be interested in verifying membership claims coming from parties with a positive grade, which means this check at P_i already happened.

$\mathsf{GCES.Gen}(\mathsf{par})$

Step 0: Each party samples their key pair.

P_i runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{CES.Gen}(\mathsf{par})$.

Step 1: Each party gradecasts their key.

1. P_i calls $\mathsf{GC}(\mathsf{pk}_i, g^*)$.
2. P_i receives $(\mathsf{pk}_j^{(i)}, \tilde{g}_j^{(i)})$ from the gradecast with sender P_j, where $\mathsf{pk}_j^{(i)}$ is the public key of P_j from the view of P_i and $\tilde{g}_j^{(i)}$ is the associated grade.

Step 3: Each party outputs the public parameters, its local key pair and public keys and associated grades for all other parties.

P_i outputs $\Big((\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j^{(i)}, \tilde{g}_j^{(i)})_{j\in[n]\setminus\{i\}}\Big)$.

Figure 1: The key generation and distribution protocol.

$\mathsf{GCES.Toss}(\mathsf{par})$

Step 0: Each party generates their pretags.

$\mathsf{pretags}_i \leftarrow \mathsf{CES.Prepare}(\mathsf{par})$.

Step 1: Each party gradecasts their pretag.

1. P_i calls $\mathsf{GC}(\mathsf{pretags}_i, g^*)$.
2. P_i receives from gradecast from sender P_j $(\mathsf{pretags}_j^{(i)}, g_j^{(i)})$, where $g_j^{(i)}$ is the grade.

Step 2: Each party acts as moderator.

1. For each $j \in [n] \setminus \{i\}$ for which P_i participated in a gradecast with sender P_j, let $m_j^{(i)} := (\mathsf{pretags}_j^{(i)}, \mathsf{Sig}_j(\mathsf{pretags}_j^{(i)}))$.
2. P_i calls $\mathsf{GC}(m_j^{(i)}, g^*)$, for all $j \in [n] \setminus i$.
3. P_i outputs $(m_{k,j}^{(i)}, g_{k,j}^{(i)})$, for each moderator P_j and for each initial sender P_k, $k \in [n] \setminus \{j\}$ where $m_{k,j}^{(i)}$ contains the puzzle, commitments and proof gradecasted by P_j for P_k, and $g_{k,j}^{(i)}$ is the respective grade.[a]

Step 3: Each party decides its output value and grade for each moderator.

1. P_i sets for each moderator P_j and initial sender P_k the pretag: $\mathsf{pretags}_{k,j}^{(i)} = (Z_{k,j}^{(i)}, C_{k,j}^{(i)}, {\pi'}_{k,j}^{(i)}) \in m_{k,j}^{(i)}$.[b]
2. P_i sets for each moderator P_j and initial sender P_k the grade:

$$G_{k,j}^{(i)} = \begin{cases} g_{k,j}^{(i)} & \text{, if } \mathsf{pretags}_{k,j}^{(i)} = \mathsf{pretags}_k^{(i)}, \\ \min\{g_{k,j}^{(i)}, g^* - g_k^{(i)}\} & \text{, otherwise.} \end{cases}$$

3. P_i sets its general grade for each moderator P_j as $G_j^{(i)} := \min_{k\in[n]}\{G_{k,j}^{(i)}\}$ and outputs for each $j \in [n]$: $((\mathsf{pretags}_{k,j}^{(i)})_{k=1}^n, G_j^{(i)})$.

[a]The notation is as follows: the superscript denotes which party receives/observes the value, in this case P_i. The subscripts denotes the order of relays. In this case it is k, j, since moderator P_j relays a value sent by party P_k in the first step.

[b]We use the membership operator here to obscure the signatures that were part of $m_{k,j}^{(i)}$.

Figure 2: $\mathsf{GCES.Toss}$ protocol (or Parallel Moderated Gradecast).

Round Complexity. The number of rounds of $\mathsf{GCES.Gen}$ is determined by running n-parallel instances of GC, which is $2g^* + 1$ rounds. $\mathsf{GCES.Toss}$ takes $4g^* + 2$ rounds, determined by running n-parallel instances of GC followed by n^2-parallel instances of GC (termination is proved in Section 4.2).

Communication Complexity. Let $\mathsf{CC}_{\mathsf{GC}}(\ell, \kappa, g^*)$ denote the total communication complexity of the gradecast protocol for a message size ℓ. The total communication complexity of $\mathsf{GCES.Gen}$ is $\mathsf{CC}_{\mathsf{GC}}(\kappa, \kappa, g^*) = O(g^* \cdot \kappa \cdot n^2)$, assuming the size of a signature, a public key X_i and a proof π_i is $O(\kappa)$. Assuming the size of a signature and a commitment (there are n of them) is $O(\kappa)$ and the size of a puzzle on n tags and of the corresponding proof is $O(\kappa n)$, the total communication complexity of $\mathsf{GCES.Toss}$ for n parties with inputs of length $O(\kappa n)$ is $n^2 \cdot \mathsf{CC}_{\mathsf{GC}}(\kappa n, \kappa, g^*) = O(g^* \kappa n^5)$. Using gossiping for the parallel moderated gradecast as in [38], one can reduce the total communication complexity of $\mathsf{GCES.Toss}$ to $\tilde{O}(g^* \kappa n^4)$.

In the remainder of this section, we analyze the agreement and security properties of GCES. In particular, Section 4.2 proves that the two protocols, $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, terminate with outputs for honest parties satisfying a graded form of consensus over the quantities of interest, the public keys and the pretags. These properties are then used in Section 4.3 and Section 4.4 to prove that an adversary has only limited influence on the resulting committee in terms of size and composition.

4.2 Agreement Properties

Note that $\mathsf{GCES.Gen}$ is essentially a parallel gradecast protocol, so for each of the gradecast instances, the properties from Definition 9 are trivially satisfied. We now show that $\mathsf{GCES.Toss}$ satisfies some generalized properties of the moderated gradecast protocols, namely *Graded Validity*, *Graded Agreement* and *Termination*, defined in the subsequent lemmata.

We first start with some helper results. Consider one moderator P_j and one sender P_i running the subprotocol in Figure 2 starting from Step 1. We show that this is an instance of a moderated gradecast protocol, which we call $\mathsf{Mod\text{-}GC}(\mathsf{pretags}_i)$, which satisfies *Validity*, *M-validity*, *Soundness* and *Termination* against an adaptive adversary controlling $t \leq (1 - \epsilon)n$ parties.

Proposition 1. *Let a_1, b_1, a_2, b_2, g such that $|a_1 - a_2| \leq 1$, $|b_1 - b_2| \leq 1$ and $g \geq \max\{a_1, a_2, b_1, b_2\}$. Let $G_i = \min\{a_i, g - b_i\}$, for $i = 1, 2$. Then, $|G_1 - G_2| \leq 1$.*

Proof. We have the following cases:
Case 1. $a_1 \leq g - b_1$ and $a_2 \leq g - b_2$. Then $|G_1 - G_2| = |a_1 - a_2| \leq 1$.
Case 2. $g - b_1 \leq a_1$ and $g - b_2 \leq a_2$. Then $|G_1 - G_2| = |g - b_1 - (g - b_2)| = |b_2 - b_1| \leq 1$.
Case 3. $g - b_1 \leq a_1$ and $a_2 \leq g - b_2$. Then $|G_1 - G_2| = |g - b_1 - a_2|$. Notice that $a_2 + b_2 \leq g \leq a_1 + b_1$, so $b_2 - b_1 \leq g - b_1 - a_2 \leq a_1 - a_2$. Both the lower and upper bound can take values in $[-1, 1]$, which constrains $|G_1 - G_2| = |g - b_1 - a_2| \leq 1$.
Case 4. $a_1 \leq g - b_1$ and $g - b_2 \leq a_2$. This mirrors case 3. □

Lemma 7. *(Moderated Gradecast) Let $\mathsf{Mod\text{-}GC}$ be the subprotocol in Figure 2 from Step 1 to Step 3.2, for a single pair of sender P_i and moderator P_j. Then, $\mathsf{Mod\text{-}GC}$ is a moderated gradecast protocol as in Definition 10.*

Proof. We show that $\mathsf{Mod\text{-}GC}$ satisfies validity, M-validity, soundness, and termination.

***Validity and M-validity*:** Let P_m be the honest moderator and let P_j be an honest party. P_m will gradecast the value obtained from sender P_i correctly, thus $g_{i,m}^{(j)} = g^*$. If P_i is also honest, then $\mathsf{pretags}_i = \mathsf{pretags}_i^{(j)} = \mathsf{pretags}_{i,m}^{(j)}$, which implies $G_{i,m}^{(j)} = g^*$. Thus, all honest parties P_j

output the same tuple $(\mathsf{pretags}_i, g^*)$, and validity holds. However, if P_i is dishonest, then it could be that $\mathsf{pretags}_i^{(j)} \neq \mathsf{pretags}_{i,m}^{(j)}$. If that is the case, then P_i gradecasts different values to P_m and P_j in Step 1, so from the consistency of GC we have $g_i^{(j)} \leq 1$. Therefore, honest party P_j has $G_{i,m}^{(j)} = \min\{g_{i,m}^{(j)}, g^* - g_i^{(j)}\} \geq g^* - 1$. Finally, since P_m is honest, all honest parties receive (and output) the same message from P_m, $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_i^{(m)}$, and M-validity holds.

***Soundness*:** For any moderator P_m and a sender P_i, let P_j, P_k be two honest parties obtaining $(\mathsf{pretags}_{i,m}^{(j)}, G_{i,m}^{(j)})$ and $(\mathsf{pretags}_{i,m}^{(k)}, G_{i,m}^{(k)})$ respectively. We have the following cases:

Case 1. $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_i^{(j)}$ and $\mathsf{pretags}_{i,m}^{(k)} = \mathsf{pretags}_i^{(k)}$. Then, $G_{i,m}^{(j)} = g_{i,m}^{(j)}$ and $G_{i,m}^{(k)} = g_{i,m}^{(k)}$, which originate both from the same gradecast and thus, by the soundness of GC, $|G_{i,m}^{(j)} - G_{i,m}^{(k)}| \leq 1$. If both $G_{i,m}^{(j)}, G_{i,m}^{(k)}$ are greater or equal to 1, then by GC soundness $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_{i,m}^{(k)}$.

Case 2. $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_i^{(j)}$ and $\mathsf{pretags}_{i,m}^{(k)} \neq \mathsf{pretags}_i^{(k)}$. Then, $G_{i,m}^{(j)} = g_{i,m}^{(j)}$ and $G_{i,m}^{(k)} = \min\{g_{i,m}^{(k)}, g^* - g_i^{(k)}\}$. Also, either $g_{i,m}^{(k)} \leq 1$ or $g_i^{(m)} \leq 1$, since one of the two gradecasts has two honest parties outputting different messages. Therefore,

- if $g_{i,m}^{(k)} \leq 1$, then $G_{i,m}^{(k)} \leq 1$ and by GC soundness, $|g_{i,m}^{(j)} - g_{i,m}^{(k)}| \leq 1$. There are two subcases. Subcase 1): $g_{i,m}^{(k)} = G_{i,m}^{(k)}$, which immediately implies $|G_{i,m}^{(j)} - G_{i,m}^{(k)}| \leq 1$. If both $G_{i,m}^{(j)}, G_{i,m}^{(k)}$ are greater or equal to 1, then by the moderator's GC soundness $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_{i,m}^{(k)}$. Subcase 2): $g_{i,m}^{(k)} = 1$ and $G_{i,m}^{(k)} = 0$, meaning that $g_i^{(k)} = g^*$. By the initial GC soundness, we also have $g_i^{(j)} \in \{g^* - 1, g^*\}$ and $\mathsf{pretags}_i^{(j)} = \mathsf{pretags}_i^{(k)}$. To obtain $|G_{i,m}^{(j)} - G_{i,m}^{(k)}| \leq 1$, we need to show it cannot hold that $g_{i,m}^{(j)} = 2$. If $g_{i,m}^{(j)} = 2$, then $\mathsf{pretags}_{i,m}^{(k)} = \mathsf{pretags}_{i,k}^{(j)}$, which is also equal to $\mathsf{pretags}_i^{(j)}$, contradicting the assumption that $\mathsf{pretags}_{i,m}^{(k)} \neq \mathsf{pretags}_i^{(k)}$.
- if $g_i^{(k)} \leq 1$, then there are again two subcases. Subcase 1): $g_{i,m}^{(k)} = g^*$, in which case $G_{i,m}^{(k)} \in \{g^* - 1, g^*\}$ and $G_{i,m}^{(j)} \in \{g^* - 1, g^*\}$ and by the moderator's GC soundness, $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_{i,m}^{(k)}$. Subcase 2): $g_{i,m}^{(k)} \leq g^* - 1$, in which case $G_{i,m}^{(k)} = g_{i,m}^{(k)}$, and by GC soundness, it holds that $|G_{i,m}^{(j)} - G_{i,m}^{(k)}| \leq 1$, as well as that if both $G_{i,m}^{(j)}, G_{i,m}^{(k)}$ are greater or equal to 1, then $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_{i,m}^{(k)}$.

Case 3. $\mathsf{pretags}_{i,m}^{(j)} \neq \mathsf{pretags}_i^{(j)}$ and $\mathsf{pretags}_{i,m}^{(k)} \neq \mathsf{pretags}_i^{(k)}$. Then $G_{i,m}^{(j)} = \min\{g_{i,m}^{(j)}, g^* - g_i^{(j)}\}$ and $G_{i,m}^{(k)} = \min\{g_{i,m}^{(k)}, g^* - g_i^{(k)}\}$, and we can use Proposition 1 to get that $|G_{i,m}^{(j)} - G_{i,m}^{(k)}| \leq 1$. If both $G_{i,m}^{(j)}, G_{i,m}^{(k)}$ are greater or equal to 1, then following the soundness of the appropriate GC instance in the four cases in Proposition 1, i.e., the sender's or the moderator's, we obtain $\mathsf{pretags}_{i,m}^{(j)} = \mathsf{pretags}_{i,m}^{(k)}$.

To finish the proof of soundness for $\mathsf{Mod\text{-}GC}$, we need to consider the case where where $G_{i,m}^{(j)} = 1$ and $\mathsf{pretags}_{i,m}^{(j)} \neq \mathsf{pretags}_{i,m}^{(k)}$. Then, by the soundness property of GC for sender P_m gradecasting its value $\mathsf{pretags}_i^{(m)}$, we know that $\mathsf{pretags}_{i,m}^{(j)} \neq \mathsf{pretags}_{i,m}^{(k)}$ implies $g_{i,m}^{(j)} = 0$ or $g_{i,m}^{(k)} = 0$. Assume $g_{i,m}^{(k)} \neq 0$, then $g_{i,m}^{(j)} = 0$ which would lead to $G_{i,m}^{(j)} = 0$, contradiction. Thus, $G_{i,m}^{(k)} = 0$ and the proof is complete.

***Termination*:** Each honest party generates a value and an accompanying grade for every moderator, by construction, regardless of the adversary's behaviour. □

We are now ready to prove the properties of the $\mathsf{GCES.Toss}$ protocol.

Lemma 8 (Graded Validity of $\mathsf{GCES.Toss}$)**.** *If party P_j is honest, then every honest party P_i outputs $((\mathsf{pretags}^{(i)}_{k,j} := \mathsf{pretags}^{(j)}_k)^n_{k=1}, G^{(i)}_j)$ for $G^{(i)}_j \in \{g^* - 1, g^*\}$.*

Proof. Since each $\mathsf{Mod\text{-}GC}$ instance called by an honest moderator P_j achieves M-validity, it holds that for any honest moderator P_j, any honest party P_i outputs $(\mathsf{pretags}^{(j)}_k, G^{(i)}_{k,j})$ for any sender P_k, with $G^{(i)}_{k,j} \in \{g^*, g^* - 1\}$. This implies that all honest parties will also have $G^{(i)}_j \in \{g^*, g^* - 1\}$. Finally, all honest parties form their output strings with all $\mathsf{pretags}^{(j)}_k$ for $k \in [n]$ since all associated grades are strictly positive when the moderator is honest. Again, by M-validity of $\mathsf{Mod\text{-}GC}$, another honest party P_m will have $\mathsf{pretags}^{(m)}_{k,j} = \mathsf{pretags}^{(j)}_k$ for all $k \in [n]$, implying the result for any honest moderator P_j. □

Lemma 9 (Graded Agreement of $\mathsf{GCES.Toss}$)**.** *If P_i, P_j are two honest parties outputting $((\mathsf{pretags}^{(i)}_{k,m})^n_{k=1}, G^{(i)}_m)$ and $((\mathsf{pretags}^{(j)}_{k,m})^n_{k=1}, G^{(j)}_m)$, respectively, for any P_m, then it holds that $|G^{(i)}_m - G^{(j)}_m| \leq 1$. Moreover, if $G^{(i)}_m > 1$, then $\mathsf{pretags}^{(i)}_{k,m} = \mathsf{pretags}^{(j)}_{k,m}$, for all $k \in [n]$, otherwise if $G^{(i)}_m = 1$, then either $\mathsf{pretags}^{(i)}_{k,m} = \mathsf{pretags}^{(j)}_{k,m}$, for all $k \in [n]$, or $G^{(m)}_j = 0$.*

Proof. Let α such that $\min_{k\in[n]}\{G^{(i)}_{k,m}\} = G^{(i)}_{\alpha,m}$. Similarly, let β such that $\min_{k\in[n]}\{G^{(j)}_{k,m}\} = G^{(j)}_{\beta,m}$. Suppose without loss of generality that $G^{(j)}_{\beta,m} \geq G^{(i)}_{\alpha,m}$. This implies also that $G^{(j)}_{\alpha,m} \geq G^{(i)}_{\alpha,m}$, since otherwise, $G^{(j)}_{\alpha,m} < G^{(j)}_{\beta,m}$, contradiction. Then, $|G^{(j)}_m - G^{(i)}_m| = |\min_{k\in[n]}\{G^{(j)}_{k,m}\} - \min_{k\in[n]}\{G^{(i)}_{k,m}\}| = |G^{(j)}_{\beta,m} - G^{(i)}_{\alpha,m}| = G^{(j)}_{\beta,m} - G^{(i)}_{\alpha,m} \leq G^{(j)}_{\alpha,m} - G^{(i)}_{\alpha,m} = |G^{(j)}_{\alpha,m} - G^{(i)}_{\alpha,m}| \leq 1$, from the soundness part of Lemma 7.

If both $G^{(i)}_m, G^{(j)}_m$ are greater or equal to 1, then all $G^{(i)}_{k,m} \geq 1$ and $G^{(j)}_{k,m} \geq 1$ and then by the soundness of $\mathsf{Mod\text{-}GC}$, it holds that for all parties P_k, $\mathsf{pretags}^{(i)}_{k,m} = \mathsf{pretags}^{(j)}_{k,m}$. To finish the proof, consider a case where $G^{(i)}_m = 1$ and some there exists some $k \in [n]$ for which $\mathsf{pretags}^{(i)}_{k,m} \neq \mathsf{pretags}^{(j)}_{k,m}$. Thus, from the soundness of $\mathsf{Mod\text{-}GC}$, $G^{(i)}_{k,m} = 0$ or $G^{(j)}_{k,m} = 0$. Assume $G^{(j)}_{k,m} \neq 0$, then $G^{(i)}_{k,m} = 0$ which would lead to $G^{(i)}_m = 0$, contradiction. Thus, $G^{(j)}_{k,m} = 0$, meaning that $G^{(j)}_m = 0$ and the proof is complete. □

Lemma 10 (Termination of $\mathsf{GCES.Toss}$)**.** *After running $\mathsf{GCES.Toss}$, any honest party P_i terminates with output $((\mathsf{pretags}^{(i)}_{k,j})^n_{k=1}, G^{(i)}_j)$ for every $j \in [n]$.*

Proof. This follows immediately from the termination of all $\mathsf{Mod\text{-}GC}$ instances and the fact that the sampling in Step 0 is guaranteed to return a result. □

We will need to use the fact that *all* honest tags have to be present in the adversary's identity id in order to be eligible for election. To this end, we prove that each value to which an honest party associates a strictly positive grade correctly uses the tags from the honest parties.

Lemma 11. *If a party P_j is graded as $G^{(i)}_j \geq 1$ by an honest party P_i in Toss, then $\mathsf{pretags}^{(i)}_{k,j}$ contains the puzzle $\mathsf{pretags}_k$ from each honest party P_k.*

Proof. Recall that the final grade for a party P_j is set by honest party P_i as $G^{(i)}_j = \min_{k\in[n]}\{G^{(i)}_{k,j}\}$. The fact that $G^{(i)}_j \geq 1$ means that for all $k \in [n]$, $G^{(i)}_{k,j} \geq 1$. For each index k corresponding to an honest party P_k, it means that honest party P_i has observed $\mathsf{pretags}^{(i)}_{k,j} = \mathsf{pretags}^{(i)}_k$, which implies that the honest value $\mathsf{pretags}^{(i)}_k = \mathsf{pretags}_k$ was included in $G^{(i)}_j$. To see why party P_i could not have observed $\mathsf{pretags}^{(i)}_{k,j} \neq \mathsf{pretags}^{(i)}_{k}$ and set $G^{(i)}_{k,j} = \min\{g^{(i)}_{k,j}, g^* - g^{(i)}_k\} > 0$, note that since $g^{(i)}_k = g^*$, the rule from Step 3 in Protocol 2 specifies that $G^{(i)}_{k,j} = 0$. □

Using the agreement properties we have now shown, we will show security properties of the committee elected via the process described in Section 4.1. These properties mirror those of the committee games from Section 3. Specifically, recall that in Section 3, we did not explicitly discuss different committee views for the honest parties. In the current section, since two different honest parties might hold different views on the pretags and public keys that define the committee, we explicitly consider each party's committee view separately. Recall that the properties we need are that (i) the adversary cannot corrupt the entire committee and (ii) the adversary cannot elect more than B parties in the committee. For the moment, we only prove these properties for parties with a strictly positive grade assigned. Later, we will ensure that the views of different honest parties regarding the committee are aligned during the broadcast protocol for all grades' values.

4.3 Graded Committee Corruption Game

The graded committee corruption game is executed between n parties. We can think of it as an extension of the committee corruption game (see Section 3.3), differing in two major ways; 1) the previous use of the public bulletin-board for anything else than signing keys and the common view assumption at the challenger are replaced by different flavors of gradecast, which leads to: 2) each party has a different view of whether another party is in the committee or not. The current game aims to capture that all honest parties still have consistent views about the honest committee members, leading to the equivalent of Lemma 3 in this setting. The *graded committee corruption game* is defined as follows:

1. **Corruptions.** $\mathcal{A}$ can corrupt a party at the beginning of any round of the simulated protocols; before the game simulates a round, $\mathcal{A}$ can select additional parties to corrupt. Throughout, the game simulates all honest parties and queries $\mathcal{A}$ during the respective steps of the protocol for the actions of dishonest parties.

2. **Setup.** The game and $\mathcal{A}$ execute protocol $\mathsf{GCES.Gen}(\mathsf{par})$. As a result, for each honest party (at that time) P_i the game obtains $\Big((\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}^{(i)}_j, \tilde{g}^{(i)}_j)_{j\in[n]\setminus\{i\}}\Big)$.

3. **Toss phase.** The game and $\mathcal{A}$ execute protocol $\mathsf{GCES.Toss}(\mathsf{par})$. As a result, for each honest party (at that time) P_i the game obtains $((\mathsf{pretags}^{(i)}_{k,j})_{k\in[n]}, G^{(i)}_j)_{j\in[n]}$, where $\mathsf{pretags}^{(i)}_j := \mathsf{pretags}^{(i)}_{j,i}$ for each $j \in [n]$.

4. **Winning Condition.** $\mathcal{A}$ wins the game if no honest party is *elected unanimously*:
   (a) Let Hon^* be the set of remaining honest parties.
   (b) The game runs $(\mathsf{res}_k, \mathsf{ticket}_k) \leftarrow \mathsf{CES.TryElect}(\mathsf{par}, \mathsf{sk}_k, (\mathsf{pretags}^{(k)}_i)_{i\in[n]})$ for each honest party P_k.
   (c) The game defines the set of honest committee members as
   $$\mathsf{HonComm} := \left\{ k \in \mathsf{Hon}^* \;\middle|\; \begin{array}{l} \mathsf{res}_k = 1 \wedge \forall i \in \mathsf{Hon}^* : \Big(\tilde{g}^{(i)}_k = g^* \text{ and } G^{(i)}_k \geq g^* - 1\Big) \wedge \\ \Big(\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}^{(i)}_k, (\mathsf{pretags}^{(i)}_{j,k})_{j\in[n]}, \mathsf{ticket}_k) = 1\Big) \end{array} \right\}.$$
   (d) The game outputs 1 if $\mathsf{HonComm} = \emptyset$. Otherwise, it outputs 0.

Intuitively, if the graded committee corruption game outputs 0, then it is guaranteed that at least one honest party will be in the committee and can convince any honest party that it is in the committee by sharing its ticket. We now show that this is the case for efficient adversaries.

Lemma 12. *Assume that the conditions for Lemmata 3 and 5 hold. Assume also that* GC *is a* g^**-gradecast protocol for* n *parties with corruption threshold* t *(see Definition 9). Then, the probability that the graded committee corruption game outputs* 1 *is at most* $\mathsf{negl}(\kappa) + \mathsf{negl}(\lambda)$.

Proof. Let $\mathcal{A}$ be a PPT adversary and $\varepsilon^{\mathcal{A}}$ denote the probability that the graded committee corruption game outputs 1. We aim to upper bound $\varepsilon^{\mathcal{A}}$. We do so by providing a sequence of hybrid games. For each hybrid $\mathbf{H}_i$, we denote the probability that it outputs 1 when run with adversary $\mathcal{A}$ by $\varepsilon_i^{\mathcal{A}}$.

Hybrid $\mathbf{H}_0$. With this we denote the graded committee corruption game. Therefore, by definition

$$\varepsilon^{\mathcal{A}} = \varepsilon_0^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_1$. We change *setup*, i.e., how the public keys are distributed. Specifically, recall that until now, the game would run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{CES.Gen}(\mathsf{par})$ for each honest party P_i, and then run a gradecast, i.e., party P_i would call $\mathsf{GC}(\mathsf{pk}_i, g^*)$. Then, each honest party P_i would receive an output $(\mathsf{pk}_j^{(i)}, \tilde{g}_j^{(i)})$ from the gradecast with sender P_j. Now, in $\mathbf{H}_1$, we directly set $\mathsf{pk}_j^{(i)} := \mathsf{pk}_j$ and $\tilde{g}_j^{(i)} := g^*$ for honest sender P_j and honest receiver P_i. By gradecast's validity property, we get

$$\varepsilon_0^{\mathcal{A}} = \varepsilon_1^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_2$. We change how $\mathsf{HonComm}$ is defined. Recall that so far, this set has been defined as

$$\mathsf{HonComm} := \left\{ k \in \mathsf{Hon}^* \;\middle|\; \begin{array}{l} \mathsf{res}_k = 1 \wedge \forall i \in \mathsf{Hon}^* : \left(\tilde{g}_k^{(i)} = g^* \text{ and } G_k^{(i)} \geq g^* - 1\right) \wedge \\ \left(\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k^{(i)}, (\mathsf{pretags}_{j,k}^{(i)})_{j\in[n]}, \mathsf{ticket}_k) = 1\right) \end{array} \right\}.$$

From $\mathbf{H}_2$ on, we instead define $\mathsf{HonComm}$ as

$$\mathsf{HonComm} := \left\{ k \in \mathsf{Hon}^* \;\middle|\; \mathsf{res}_k = 1 \wedge \mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_k, (\mathsf{pretags}_j^{(k)})_{j\in[n]}, \mathsf{ticket}_k) = 1 \right\}.$$

We claim that the two definitions are equivalent for these two hybrids. Indeed, if $k \in \mathsf{HonComm}$ in $\mathbf{H}_2$, then trivially $k \in \mathsf{HonComm}$ in $\mathbf{H}_1$. Conversely, assume that $k \in \mathsf{HonComm}$ in $\mathbf{H}_1$, so, by definition, P_k is honest. Due to the change in $\mathbf{H}_1$, we know that for all honest parties P_i, it holds that $\tilde{g}_k^{(i)} = g^*$ and $\mathsf{pk}_k^{(i)} := \mathsf{pk}_k$. From the graded validity property (see Lemma 8), we get that for an honest moderator P_k, the game obtains with respect to each honest receiver P_i the values $(\mathsf{pretags}_{j,k}^{(i)})_{j\in[n]} = (\mathsf{pretags}_j^{(k)})_{j\in[n]}$ and $G_k^{(i)} \geq g^* - 1$. Thus, $k \in \mathsf{HonComm}$ in $\mathbf{H}_2$. So, we have

$$\varepsilon_1^{\mathcal{A}} = \varepsilon_2^{\mathcal{A}}.$$

Reduction to the Committee Corruption Game. Finally, one can easily bound $\varepsilon_2^{\mathcal{A}}$ via a reduction to the committee corruption game of Lemma 3. We sketch the reduction:

1. The reduction gets as input from the committee corruption game $\mathsf{par} \leftarrow \mathsf{CES.Setup}(1^\kappa)$. It then starts to simulate hybrid $\mathbf{H}_2$ for $\mathcal{A}$. After $\mathcal{A}$ made its initial corruptions, the reduction declares the set of initially corrupted parties to the committee corruption game.

2. The reduction gets from the committee corruption game a public key pk_i for every honest party P_i and uses them to simulate the *setup* of $\mathbf{H}_2$. It also gets access to a corruption oracle, which it uses whenever $\mathcal{A}$ decides to corrupt a party.

3. Recall that in the tag phase of the committee corruption game, the reduction has to output n^2 pretags. To do so, the reduction runs the *toss phase* of the graded committee corruption game. Then, for each honest P_k, the reduction outputs $\mathsf{pretags}_i^{(k)}$ for each $j \in [n]$, and for each other $k \in [n]$ it outputs some default values. Note that these are not relevant for the committee corruption game.

The reduction perfectly simulates $\mathbf{H}_2$ for $\mathcal{A}$ and is efficient. Further, the committee corruption game outputs 1 whenever $\mathbf{H}_2$ does, which is primarily due to the change in $\mathbf{H}_2$. Then, via Lemmata 3 and 5, we obtain that

$$\varepsilon_2^{\mathcal{A}} \leq \mathsf{negl}(\kappa) + \mathsf{negl}(\lambda).$$

□

4.4 Graded Large Committee Game

The graded large committee game is executed between n parties. It extends the large committee game from Section 3.4, where, similarly to Section 4.3, honest parties might have different views on whether a party is in the committee. The current game captures that even so, the adversary cannot select—within an acceptable time frame depending on the difficulty parameter of the puzzles—pretags such that the committee exceeds size B in the view of any honest party. The difficulty parameter of the puzzles should be determined by the first two steps of $\mathsf{GCES.Toss}$, which correspond to two sequential runs of GC with grade g^*. In particular, let $\Delta' = (4 \cdot g^* + 2) \cdot \Delta_r$ be the duration of these steps. Then, the difficulty of the puzzle is $\Delta(\Delta')$ as in Remark 1. The *graded large committee game* is formally written as follows:

1., 2., 3. The same steps as in the graded committee corruption game.

4. **Winning Condition.** $\mathcal{A}$ wins the game if there exists an honest party P_i for which too many parties are elected in its view:

   (a) Let Hon^* be the set of remaining honest parties.

   (b) The game runs $\mathcal{A}$ and gets, for every $i \in \mathsf{Hon}^*$, a set $\mathsf{Committee}^{(i)} \subseteq [n]$ and $\mathsf{ticket}_j^{(i)}$ for all $j \in \mathsf{Committee}^{(i)}$.

   (c) The game outputs 1 if the following two conditions hold for at least one $i \in \mathsf{Hon}^*$:

      i. We have $|\mathsf{Committee}^{(i)}| > B$.

      ii. For all $j \in \mathsf{Committee}^{(i)}$, it holds that:

$$\tilde{g}_j^{(i)} \geq 1, G_j^{(i)} \geq 1, \text{ and } \mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_j^{(i)}, (\mathsf{pretags}_{k,j}^{(i)})_{k\in[n]}, \mathsf{ticket}_j^{(i)}) = 1.$$

   Otherwise, the game outputs 0.

Lemma 13. *Assume that the conditions for Lemmata 5 and 6 hold. Assume also that* GC *is a* g^**-gradecast protocol for* n *parties with corruption threshold* t *(see Definition 9). Finally, assume that the conditions for Lemma 4 hold, where* TLP *is* $\Delta(\Delta')$*-CPA secure, for* $\Delta' := (4 \cdot g^* + 2) \cdot \Delta_r$*. Then, the probability that the graded large committee game outputs* 1 *is at most* $\mathsf{negl}(\kappa) + \mathsf{negl}(\lambda)$.

Proof. Let $\mathcal{A}$ be a PPT adversary and $\varepsilon^{\mathcal{A}}$ denote the probability that the graded large committee game outputs 1. We aim to upper bound $\varepsilon^{\mathcal{A}}$. We do so by providing a sequence of hybrid games. For each hybrid $\mathbf{H}_i$, we denote the probability that it outputs 1 when run with adversary $\mathcal{A}$ by $\varepsilon_i^{\mathcal{A}}$.
Hybrid $\mathbf{H}_0$. With this we denote the graded large committee game. Therefore, by definition

$$\varepsilon^{\mathcal{A}} = \varepsilon_0^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_1$. We change *setup*, i.e. how the public keys are distributed. Specifically, recall that in the previous hybrid, for each honest party P_i, the game runs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{CES.Gen}(\mathsf{par})$ and then a gradecast in which party P_i calls $\mathsf{GC}(\mathsf{pk}_i, g^*)$. Then, each honest party P_i would receive an output $(\mathsf{pk}_j^{(i)}, \tilde{g}_j^{(i)})$ from the gradecast with sender P_j. Now, in $\mathbf{H}_1$, the game directly sets $\mathsf{pk}_j^{(i)} := \mathsf{pk}_j$ and $\tilde{g}_j^{(i)} := g^*$ for honest P_i and P_j. Further, for each dishonest sender P_j at that point P_j, recall that honest party P_i obtains $(\mathsf{pk}_j^{(i)}, \tilde{g}_j^{(i)})$ from the gradecast. The game outputs 0 if the values and grades do not satisfy the soundness property of Definition 9. Therefore, assuming hybrid $\mathbf{H}_1$ outputs 1, we know that for each such j, we are in one of the following three cases:

- For every honest pair P_k, P_l and dishonest party P_j it holds that $\mathsf{pk}_j^{(k)} = \mathsf{pk}_j^{(l)} =: \mathsf{pk}_j$, or
- there is at least one honest party P_k for which $\tilde{g}_j^{(k)} = 1$. In this case, fix the minimal such k and set $\mathsf{pk}_j := \mathsf{pk}_j^{(k)}$. We know that for every honest party P_l for which $\mathsf{pk}_j^{(l)} \neq \mathsf{pk}_j^{(k)}$ it holds that $\tilde{g}_j^{(l)} = 0$; or
- we have $\tilde{g}_j^{(k)} = 0$ for every honest party P_k. In this case, note that P_j cannot be part of any winning committee. For convenience, let pk_j be a default public key in this case.

For the first and second cases, for every honest party P_i and dishonest party P_j we set $\mathsf{pk}_j^{(i)} = \mathsf{pk}_j$ and $\tilde{g}_j^{(i)} = 1$. The new game is not indistinguishable from $\mathbf{H}_0$ but is at least as likely to output 1, since it defines a superset of cases that satisfy the winning condition. Therefore,

$$\varepsilon_0^{\mathcal{A}} \leq \varepsilon_1^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_2$. We now equalize the honest parties' view on honestly generated pretags: Assuming that hybrid $\mathbf{H}_1$ outputs 1, then for any honest party P_k, we can observe (via Lemma 11) that for any moderator P_l (honest or dishonest) one of the following holds, where $\mathsf{pretags}_k$ denotes the value that the game generates when executing Step 1 of $\mathsf{GCES.Toss}$ with respect to honest party P_k:

- For any pair of honest parties P_i, P_j, we have $\mathsf{pretags}_{k,l}^{(i)} = \mathsf{pretags}_{k,l}^{(j)} = \mathsf{pretags}_k$. In that case, set $\mathsf{pretags}_{k,l} := \mathsf{pretags}_k$; or
- For any honest party P_i for which $\mathsf{pretags}_{k,l}^{(i)} \neq \mathsf{pretags}_k$, it holds that $G_l^{(i)} = 0$. In this case, note that P_l cannot be part of winning committee $\mathsf{Committee}^{(i)}$.

We set for all honest parties P_i, P_k and all moderators j: $\mathsf{pretags}_{k,j}^{(i)} = \mathsf{pretags}_k$ and reset $G_j^{(i)} = 1$, if $G_j^{(i)}$ was 0 before. This new game is not necessarily indistinguishable from the previous, but is at least as likely to output 1 (previous winning pretags are still winning now). So, we have

$$\varepsilon_1^{\mathcal{A}} \leq \varepsilon_2^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_3$. We now equalize the honest parties' view on pretags coming from dishonest parties. Therefore, note that assuming hybrid $\mathbf{H}_2$ outputs 1, we know that for any dishonest party P_k and moderator P_l one of the following holds:

- For any pair of honest parties P_i, P_j, we have $\mathsf{pretags}^{(i)}_{k,l} = \mathsf{pretags}^{(j)}_{k,l}$. In that case, set $\mathsf{pretags}_{k,l} := \mathsf{pretags}^{(i)}_{k,l}$; or

- There is at least one honest party P_i for which $G^{(i)}_{k,l} = 1$. In this case, fix the minimal such i and set $\mathsf{pretags}_{k,l} := \mathsf{pretags}^{(i)}_{k,l}$. We know from the soundness property in Lemma 7 that for every honest party P_j for which $\mathsf{pretags}^{(j)}_{k,l} \neq \mathsf{pretags}_{k,l}$ it holds that $G^{(j)}_{k,l} = 0$ and thus $G^{(j)}_{l} = 0$; or

- We have $G^{(i)}_{l} = 0$ for every honest party P_i. In this case, note that P_l cannot be part of any winning committee.

For the first and second cases, for all honest parties P_i and all moderator j, we set $\mathsf{pretags}^{(i)}_{k,j} = \mathsf{pretags}_{k,j}$ and $G^{(i)}_{j} = 1$. This new game is not indistinguishable from the previous one, but is at least as likely to output 1 (previous winning pretags are still winning now). So, we have

$$\varepsilon_2^{\mathcal{A}} \leq \varepsilon_3^{\mathcal{A}}.$$

Hybrid $\mathbf{H}_4$. We change the winning condition. Recall that the winning condition before was the existence of $i \in \mathsf{Hon}^*$ such that (1) $\forall j \in \mathsf{Committee}^{(i)}$: $\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}^{(i)}_j, (\mathsf{pretags}^{(i)}_{k,j})_{k\in[n]}, \mathsf{ticket}^{(i)}_j) = 1$ and $\tilde{g}^{(i)}_j \geq 1$, $G^{(i)}_j \geq 1$, and (2) $|\mathsf{Committee}^{(i)}| > B$. Now we set the game to output 1 if there exists $i \in \mathsf{Hon}^*$: (1) $\forall j \in \mathsf{Committee}^{(i)}$: $\mathsf{CES.VerElect}(\mathsf{par}, \mathsf{pk}_j, (\mathsf{pretags}_{k,j})_{k\in[n]}, \mathsf{ticket}^{(i)}_j) = 1$, and (2) $|\mathsf{Committee}^{(i)}| > B$. By our previous changes, all grades are set to 1 in $\mathbf{H}_3$ and all pretags are equalized for the honest parties, the two hybrids are equivalent. Thus, we have

$$\varepsilon_3^{\mathcal{A}} = \varepsilon_4^{\mathcal{A}}.$$

Reduction to the Large Committee Game. One can clearly bound $\varepsilon_4^{\mathcal{A}}$ via a reduction to the large committee game of Lemma 4. The reduction internally simulates hybrid $\mathbf{H}_4$ for the adversary, and its goal is to win in the large committee game. We sketch the reduction below.

1. The reduction samples a random party and starts simulating hybrid $\mathbf{H}_4$ for the adversary as explained below. If at point this party is corrupted, the reduction aborts. For convenience, we denote this party as P_1, to match the notation of the large committee game.

2. The reduction gets as input from the large committee game $\mathsf{par} \leftarrow \mathsf{CES.Setup}(1^\kappa)$. Recall that it has to output public keys pk_i for every $i \in [n]$. To do so, the reduction simulates the *setup* of hybrid $\mathbf{H}_4$ for the adversary. Note that due the change in hybrid $\mathbf{H}_1$, pk_i is defined for every $i \in [n]$. The reduction outputs these pk_i.

3. In the *tag phase* of the large committee game, the reduction gets as input $\mathsf{pretags}_1$ and has to output $n \cdot (n-1)$ pretags, within bounded time $\Delta' = (4 \cdot g^* + 2) \cdot \Delta_r$. To do so, the reduction runs the *toss phase* of the graded large committee game, rigged with $\mathsf{pretags}_1$ specifically for P_1. Recall that the *toss phase* incurs exactly the execution of $\mathsf{GCES.Toss}(\mathsf{par})$, which takes $4 \cdot g^* + 2$ rounds, where each round consists of Δ_r timesteps. Therefore, the reduction receives the outcome of $\mathsf{GCES.Toss}(\mathsf{par})$ within $(4 \cdot g^* + 2) \cdot \Delta_r = \Delta'$ time. Then, for each party P_k, for all $k \in [n]$, the reduction outputs

   (a) $\mathsf{pretags}_{i,k} = \mathsf{pretags}_i$ for each $i \in [n] \setminus \{1\}$ that is honest (per hybrid $\mathbf{H}_2$) and

(b) $\mathsf{pretags}_{j,k}$ according to hybrid $\mathbf{H}_3$ for each dishonest party P_j.

4. Finally, the reduction has to output $\mathsf{Committee}$ and a ticket ticket_k for all $k \in \mathsf{Committee}$. To do so, the reduction continues running the adversary until it outputs $\mathsf{Committee}^{(1)}$ and tickets $\mathsf{ticket}_k^{(1)}$ for all $k \in \mathsf{Committee}^{(1)}$. The reduction simply sets and outputs $\mathsf{Committee} := \mathsf{Committee}^{(1)}$ and $\mathsf{ticket}_k := \mathsf{ticket}_k^{(1)}$.

Clearly, the reduction is PPT, and as already explained, it satisfies the time constraints of the large committee game. Further, perfectly simulates $\mathbf{H}_4$ for $\mathcal{A}$ assuming it guessed correctly and does not abort. If $\mathbf{H}_4$ outputs 1, then there is at least one honest party for which the winning condition of the large committee game holds. As the view of the adversary does not depend on the guessed party, we get $\varepsilon_4^{\mathcal{A}} \leq n\varepsilon'$, where ε' is the probability that the reduction wins the large committee game. By Lemmata 4 and 6, we get

$$\varepsilon_4^{\mathcal{A}} \leq n\varepsilon' \leq \mathsf{negl}(\kappa) + \mathsf{negl}(\lambda).$$

□

5 Our Broadcast Protocol

We now construct our sublinear broadcast protocol. Our protocol follows the committee-based protocol of Chan *et al.* [10]. The main difference is that we do not assume a trusted setup of keys and of public parameters. Instead, parties rely on a bulletin-PKI (only for the signature public keys), an unstructured common random string, and a graded protocol, which generates keys and tags, and can be seen as an online setup phase that issues graded random identifier strings and proofs of the validity of these identifiers. These identifiers are then used to correctly and verifiably elect bit-specific committees. We need bit-specific committees instead of a single committee to prevent the adaptive adversary from corrupting a party who voted for one bit and make it also vote for the other bit.

However, we do not exactly obtain the equivalent of a trusted setup from Section 4, as the only guarantees that parties have are related to the grades of these random strings, where the maximum grade is denoted as g^* and the minimum grade can be 0. This can be seen as a *graded mining functionality*, with the terminology of mining from Chan *et al.* [10], taken to mean consistent committee election and membership verification. Below, we describe how to use the grades to our advantage over a number of rounds, in order to obtain true agreement between parties.

5.1 Voting and Vote Distribution

Let us first review in more detail the broadcast protocol of Chan *et al.*: a party checks if it is in the committee for the bit b via an (ungraded) ideal mining functionality, which also allows other parties to validate this statement. This mining functionality is instantiated via an adaptively secure VRF (in [10] implemented with non-interactive zero-knowledge proofs with trusted setup and commitment schemes). This enables parties to secretly but verifiably self-elect in a committee for a specific bit and only reveal their membership after they have performed their committee task, thus achieving security against an adaptive adversary. The protocol is composed of stages, each stage r having two rounds: distribution and voting. For a fixed number of rounds, each party observing a batch of r valid signatures from the committee members of b, which can be interpreted as votes, echoes this batch to all parties (distribution round). A party that is in the committee adds its vote if it observes a batch of r votes on the bit b for the first time, and multicasts the updated batch of $r+1$ signatures (voting round). Chan *et al.* show that it is possible to achieve consistency with overwhelming probability even if the number of rounds is constant and the committee size is also constant, against a constant corrupted fraction.

In our case, there is a key difference with respect to the mining functionality from Chan *et al.*, which is that the verification performed by other parties on the membership of one party will return a binary answer *and* a grade in $\{0, \ldots, g^*\}$. This mining functionality does not necessarily return the same grade to all parties, but the returned grades to two honest parties can differ by at most one. However, dishonest parties might try to convince honest parties to accept their membership despite having lower grades. To address this, we will interconnect the validity of the membership at a given round with the grade associated to the party that wants to prove is a committee member, such that the parties in which the honest parties do not have confidence can only vote in the very last round or not at all. Specifically, we set the maximum grade g^* that can be returned by the gradecast protocols to be equal to the number of rounds the Chan *et al.* protocol requires. We also say that a batch of r signatures is valid only if there are r signatures from parties on which the verification predicate returns 1 *and* which have a sufficiently large grade, greater than g^*-2r+1 (we will define this more formally below). A symmetric way to view this is that at each Round ρ, the value of the grades have to be at least $g^*-\rho$, and the number of signatures has to be at least half of the round number. This ensures that parties that have a grade of 1 can only submit their signatures in the last possible round and parties that have grade of 0 cannot submit their signatures at all.

Since the grades obtained by the honest parties might differ by one (honest parties can be graded by g^* or g^*-1), an honest party P_i might accept a batch with r signatures, i.e., all grades for the signers that P_i has are at least g^*-2r+1, but another honest party P_j might not accept it if it has a lower grade g^*-2r for one of the same signers. To avoid this, in Stage r, in the distribution Round $2r-1$, where parties just echo what they received, honest parties accept r-batches with grade at least g^*-2r+1. At the end of voting Round $2r$ however, where committee parties multicast their votes after seeing a valid batch for that round (with grades at least g^*-2r), committee parties are allowed to have a lower grade of at least g^*-2r, in order to be picked up in the distribution round of Stage $r+1$.

5.2 Broadcast Protocol Description

With these ideas in mind, we now describe our broadcast protocol in more detail.
Preparing Committee Election. There will be one committee for each value (0 or 1) that may be broadcasted. Therefore, the protocols $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$ will produce strings for two bits, and the inputs of the algorithms $\mathsf{CES.TryElect}$ and $\mathsf{CES.VerElect}$ will be parametrized by the bit b. In the previous section, we preferred to describe $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$ for a single committee for clarity purposes; in this section, we slightly abuse the previous notation to accommodate two committees. We emphasize that $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$ are run a single time, not twice in parallel.

Parties read the transparent system parameters par and first execute the key generation and distribution protocol $\mathsf{GCES.Gen}$ from Figure 1 to receive keys that correspond to each committee. In particular, each party P_i obtains the output

$$\left((\mathsf{pk}_i^b, \mathsf{sk}_i^b)_{b\in\{0,1\}}, \left((\mathsf{pk}_j^{(i),b})_{b\in\{0,1\}}, \tilde{g}_j^{(i)} \right)_{j\in[n]\setminus\{i\}} \right) \leftarrow \mathsf{GCES.Gen}(\mathsf{par}). \tag{4}$$

In the $\mathsf{GCES.Toss}$ protocol from Figure 2, parties will compute their pretags twice, namely, one set for each bit b. As such, each party P_i will obtain the output

$$\left(\left((\mathsf{pretags}_{k,j}^{(i),b})_{k=1}^n, G_j^{(i)} \right)_{j=1}^n \right) \leftarrow \mathsf{GCES.Toss}(\mathsf{pp}). \tag{5}$$

To recall, P_j is the moderator.

Note that in both $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, although there are two (sets of) strings gradecast, one for every bit, the final grade is the same for both. In other words, if a party submits incorrect messages for one bit but not for the other, it will be penalized in both cases.

A party P_i can then run algorithm $\mathsf{CES.TryElect}$ algorithm for a specific bit $b \in \{0,1\}$, namely, it can run:

$$\left(\mathsf{res}^b, \mathsf{ticket}_i^b\right) \leftarrow \mathsf{CES.TryElect}\left(\mathsf{par}, \mathsf{sk}_i^b, (\mathsf{pretags}_j^{(i),b})_{j\in[n]}\right). \tag{6}$$

Finally, to verify that a party P_j is in the committee for bit b, P_i will run $\mathsf{CES.VerElect}$:

$$\mathsf{res}_j^b := \mathsf{CES.VerElect}\left(\mathsf{par}, (\mathsf{pretags}_{k,j}^{(i),b})_{k\in[n]}, \mathsf{ticket}_j^{(i),b}\right). \tag{7}$$

That is, in verifying the committee claim of P_j, the verifying party P_i uses the values $(\mathsf{pretags}_{k,j}^{(i),b})_{k\in[n]}$ that it obtained as an output in Toss for P_j, but the values $\mathsf{ticket}_j^{(i),b}$ that party P_j provided to convince P_i that it is in the committee.

Tracking Committee Membership. Each party P_i maintains two variables, named res_i^0 and res_i^1. The role of the variable res_i^b is to store the local view of whether P_i is in the committee for bit $b \in \{0,1\}$. Recall that as part of $\mathsf{CES.TryElect}$, for the bit b, the party P_i checks (once) whether it is in the committee for b and sets the variable res_i^b. In particular, at the beginning of the broadcast protocol, P_i checks the value of res_i^b for each bit b. During the later stages of broadcast, once they receive the first vote on b, P_i uses res_i^b to decide whether it will also send a vote on b or not. Once this check for b is done, P_i sets $\mathsf{res}_i^b = \perp$ to internally record this and not check (and vote) again.

The main part of the broadcast protocol can start after the amount of time it took for each honest party P_i to obtain the result of its committee election for each bit $(\mathsf{res}_i^0, \mathsf{res}_i^1)$. Recall that Δ_r denotes the duration of a round and that the time an honest party takes to solve a puzzle of difficulty $\Delta(\Delta')$ is $p_1(\kappa,\Delta)+p_2(\kappa,n)$ (Definition 7 and Remark 1). Based on Lemma 13, the difficulty parameter $\Delta(\Delta')$ is set based on the duration of two gradecast protocols, i.e., on $\Delta' := (4g^*+2)\cdot\Delta_r$ rounds. We obtain this value of Δ' by instantiating the predicate Pred as in Section 3.5 and setting the maximum grade in the gradecast protocols g^* to be twice the committee bound, $g^* = 2\cdot\lceil 3\log\left(\delta^{-1}\right)/\epsilon\rceil = O(\lambda/\epsilon)$. For convenience, we define $q_h := (p_1(\kappa,\Delta) + p_2(\kappa,n) - \Delta')/\Delta_r$, which the additional number of rounds an honest party takes to solve the batch puzzle compared to Δ' (in reality, this slow-down is very small and is definitely sublinear in the number of parties). We also set the number of the rest of the rounds in the broadcast protocol to also be g^*, which is sublinear in the number of parties. Intuitively, Lemma 5 and Lemma 6 guarantee that for this choice of parameters, the generated bit-specific committees contain at least one honest party, and at most $g^*/2$ dishonest parties with overwhelming probability. More details on the total number of rounds and the security properties are given at the end of this section.

Verifying Membership, Valid Batches and Certificates. Our protocol consists of stages, where each stage is composed of two rounds. We denote the stage number by r for $r \in \{1,\ldots,g^*/2\}$. Then round 1 of stage r will be the $2r-1$'th round, and round 2 of stage r will be the $2r$'th round.

Each party collects *batches* of signatures. We will refer to a signature $\mathsf{Sig}_j(b)$ from party P_j as a j-signature. Parties (except for the sender) will send the previously collected signatures in a batch batch, as well as proofs of the committee elections for the bit b in a *certificate* called cert. In particular, a new batch batch'_b constructed from another batch batch_b, using a signature $\mathsf{Sig}_i(b)$ from a party P_i whose signature was not in batch_b, is denoted as $\mathsf{batch}'_b := \mathsf{batch}_b \| \mathsf{Sig}_i(b)$, where $\|$ means concatenation. Similarly, a new certificate constructed from another certificate cert_b, using a ticket ticket_i^b corresponding to a party P_i whose ticket was not in cert_b, is denoted as $\mathsf{cert}'_b := \mathsf{cert}_b \| \mathsf{ticket}_i^b$.

To address the difference in grades in the two rounds, we define $(r, 1)$-batches and $(r, 2)$-batches. For notation purposes, we will use *ballot* to denote a batch and its associated certificate. For clarity, we prefer to define separately the validity of the batches and of the certificates, rather than lumped in a single definition of a valid ballot.

Definition 11 (Batches, Certificates, and Ballots)**.** *A* batch *consists of a number of votes, which are signatures associated to distinct parties. A* certificate *consists of a number of tickets coming from distinct parties. For a batch for bit b, batch_b, consisting of a number of signatures $(\mathsf{Sig}_j(b))_j$, we say certificate cert_b is associated to batch_b if it consists of tuples $(\mathsf{ticket}_j^b)_j$ for every j-signature present in batch_b, for parties $j \in [n]$. Finally, we call a pair of one batch and its associated certificate a* ballot.

We define a valid certificate only with respect to a certain batch. In our protocols, the verification of signatures is performed implicitly.

Definition 12 (Validity of Certificates)**.** *We say that a certificate cert_b is a* valid certificate *for a batch batch_b from the view of a verifying party P_i if for all valid j-signatures in batch_b not coming from the sender, it holds that:*

$$\mathsf{CES.VerElect}\left(\mathsf{par}, (\mathsf{pretags}_{k,j}^{(i),b})_{k\in[n]}, \mathsf{ticket}_j^{(i),b}\right) = 1,$$

where for party P_i we used the notation $(\mathsf{ticket}_j^{(i),b})_j = \mathsf{cert}_b$ for all j-signatures in batch_b, and $\left((\mathsf{pretags}_{k,j}^{(i),b})_{k\in[n]}\right)_j$ are from P_i's state.

Definition 13 (Validity of Batches and Ballots)**.** *We say that a batch batch_b, consisting of a tuple of form $(\mathsf{Sig}_j(b))_j$, $j \in [n]$, in Stage r for a bit b, is a* valid $(r, 1)$-batch *from the perspective of a verifying party P_i if:*

(i) It contains at least r valid distinct signatures and one of the signatures is from the sender P_s;

(ii) For every valid j-signature $\mathsf{Sig}_j(b)$, it holds that $G_j^{(i)} \geq g^* - 2r + 1$, and $\tilde{g}_j^{(i)} \geq g^* - 2r + 1$, where $G_j^{(i)}, \tilde{g}_j^{(i)}$ are from P_i's state;

(iii) It has associated a valid certificate cert_b (P_i receives both batch_b and cert_b).

Similarly, a batch is considered a valid $(r, 2)$-batch *by party P_i if (i), (iii) hold as above and also*

(ii)' For every valid j-signature $\mathsf{Sig}_j(b)$, it holds that $G_j^{(i)} \geq g^* - 2r$, and $\tilde{g}_j^{(i)} \geq g^* - 2r$, where $G_j^{(i)}, \tilde{g}_j^{(i)}$ are from P_i's state.

Moreover, a ballot is valid *if its batch is valid and its certificate is a valid certificate for that batch.*

Further, each party P_i will also maintain a set $\mathsf{Extracted}_i$, initialized to the empty set. A party P_i adds a bit b to their set $\mathsf{Extracted}_i$ in a round if it receives a valid batch and a valid certificate on b for that round. With these definitions, the full broadcast protocol is described in Figure 3.

Π_{BC}: Sublinear-round, plain PKI Broadcast

Let $g^* := 2\left\lceil 3\log\left(\delta^{-1}\right)/\epsilon\right\rceil = O(\lambda/\epsilon)$.

Rounds $-6g^* - 3 - q_h$[a] to -1 :

1. Each party P_i performs (4), (5), (6) and stores their outputs. In particular, it stores $(\mathsf{res}_i^0, \mathsf{ticket}_i^0, \mathsf{res}_i^1, \mathsf{ticket}_i^1)$ as its own election result for committees 0 and 1.

Stage 0:

1. (Round 0) Each party P_i initializes $\mathsf{Extracted}_i = \emptyset$.
   The designated sender P_s sends $(b_s, \mathsf{Sig}_s(b_s), \perp)$ to all parties, where b_s is the sender's input bit.

Stage $r = 1$ to $g^*/2 - 1$:

1. (Round $2r - 1$) Each party P_i accepts a message $b \notin \mathsf{Extracted}_i$, i.e., sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$, only if it is accompanied by some valid ballot $(\mathsf{batch}_b, \mathsf{cert}_b)$, where batch_b is a valid $(r,1)$-batch and cert_b is a valid certificate for batch_b.

   P_i then multicasts $(b, \mathsf{batch}_b, \mathsf{cert}_b)$ to all parties.
2. (Round $2r$) Each party $P_i \neq P_s$ checks all bits b that it received on whether they are accompanied by a valid ballot $(\mathsf{batch}_b, \mathsf{cert}_b)$, where batch_b is a valid $(r,2)$-batch and cert_b is a valid certificate for batch_b.
   For each bit b, P_i checks if $\mathsf{res}_i^b = 1$, and if so:

   - sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$;
   - constructs a $(r+1,1)$-batch $\mathsf{batch}'_b := \mathsf{batch}_b \| \mathsf{Sig}_i(b)$, $\mathsf{cert}'_b := \mathsf{cert}_b \| \mathsf{ticket}_i^b$.

   P_i then sets $\mathsf{res}_i^b = \perp$ and sends $(b, \mathsf{batch}'_b, \mathsf{cert}'_b)$ to all n parties.

Stage $r = g^*/2$:

1. (Round $g^* - 1$) Each party P_i accepts each message $b \notin \mathsf{Extracted}_i$, i.e., sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$, that is accompanied by a valid $(g^*/2, 1)$-batch and a valid certificate for that batch.
   P_i then outputs either the message $b' \in \mathsf{Extracted}_i$ if $|\mathsf{Extracted}_i| = 1$, or a canonical message otherwise.

[a] q_h is the extra number of rounds over $4g^* + 2$ that an honest party takes to solve a puzzle with difficulty $(4g^* + 2) \cdot \Delta_r$ (negligible in practice).

Figure 3: Broadcast protocol for designated sender P_s and parties $P_1, \ldots, P_n$.

5.3 Proof of Broadcast Properties

We now show that Π_{BC} is a broadcast protocol as defined in Definition 8. Our setup assumptions are the existence of a bulletin-PKI and a uniform common random string, and our computational assumptions are the hardness of discrete logarithm and the security of time-lock puzzles and zero-knowledge proofs.

Theorem 1. *Consider a PPT adversary who can adaptively corrupt $(1-\epsilon)n$ parties, for a constant $\epsilon \in (0,1)$. Then, under the assumptions above, the protocol Π_{BC} given in Figure 3 satisfies the properties of a broadcast protocol except with probability* $\mathsf{negl}(\lambda) + \mathsf{negl}(\kappa)$.

The proof ties together all the results based on committee elections, graded consensus on the committees, and on the valid ballot creation and verification. We will write the proof of Theorem 1 as three separate lemmata, each proving a property of the broadcast protocol: Lemma 16 for validity, Lemma 17 for consistency, and Lemma 18 for termination. For validity and consistency, we first show some helper results.

Lemma 14. *Consider a Stage $r < g^*/2$, a bit b and a ballot $(\mathsf{batch}_b, \mathsf{cert}_b)$ that is $(r,1)$-valid for honest party P_i. Then, $(\mathsf{batch}_b, \mathsf{cert}_b)$ will be $(r,2)$-valid for any honest party P_j.*

Proof. We will prove that if an honest party P_i accepted a ballot in Round $2r-1$ and forwarded it, then an honest party P_j will accept the same ballot in Round $2r$.

Since batch_b is considered valid by party P_i, it means it is accompanied by a valid certificate cert_b, and according to Definition 13, the following hold:

(i) It contains at least r valid distinct signatures and one of the signatures is from the sender P_s;

(ii) For every valid k-signature $\mathsf{Sig}_k(b) \in \mathsf{batch}_b$, it holds that $G_k^{(i)} \geq g^* - 2r + 1$; and $\tilde{g}_k^{(i)} \geq g^* - 2r + 1$, where $G_k^{(i)}, \tilde{g}_k^{(i)}$ are from P_i's state.

(iii) cert_b is valid.

Now assume honest party P_j receives $(b, \mathsf{batch}_b, \mathsf{cert}_b)$ during the second round of Stage r and checks whether batch_b is a valid $(r,2)$-batch. We show that all three properties according to Definition 13 will hold for P_j.

(i) It holds trivially from the fact that batch_b is a $(r,1)$-valid batch and the signing public keys are posted on the bulletin-PKI.

(ii)' For all signatures $\mathsf{Sig}_k(b) \in \mathsf{batch}_b$, from the graded agreement property of GC in $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, it holds that $|\tilde{g}_k^{(i)} - \tilde{g}_k^{(j)}| \leq 1$ and $|G_k^{(i)} - G_k^{(j)}| \leq 1$. Since $G_k^{(i)} \geq g^* - 2r + 1$, it also holds that $G_k^{(j)} \geq g^* - 2r$, for any $r < g^*/2$, and similarly $\tilde{g}_k^{(j)} \geq g^* - 2r$.

(iii) From item (ii), it holds that $\tilde{g}_k^{(i)}, \tilde{g}_k^{(j)}, G_k^{(i)}, G_k^{(j)}$ are always strictly positive for $r \leq g^*/2 - 1$ (in particular, the minimum grade is at least 2). Then, from the graded agreement of $\mathsf{GCES.Toss}$, it holds that $\mathsf{pretags}_{l,k}^{(i),b} = \mathsf{pretags}_{l,k}^{(j),b}$ for all $k, l \in [n] : \mathsf{Sig}_k(b) \in \mathsf{batch}_b$. The same argument holds for $\mathsf{pk}_k^{(i)} = \mathsf{pk}_k^{(j)}$ for all $k \in [n] : \mathsf{Sig}_k(b) \in \mathsf{batch}_b$. Therefore, it follows that since cert_b is valid for P_i, it is also valid for P_j (since P_j will perform the same checks on the same values of pretags and public keys, contained in batch_b, and tickets, contained in cert_b, as P_i did).

□

Lemma 15. *Consider a Stage $r < g^*/2$, a bit b and a batch batch_b (with certificate cert_b) that is $(r,2)$-valid for honest party P_i for which $\mathsf{res}_i^b = 1$ and $\mathsf{Sig}_i(b) \notin \mathsf{batch}_b$. Then, the ballot $(\mathsf{batch}'_b, \mathsf{cert}'_b)$ with $\mathsf{batch}'_b := \mathsf{batch}_b \| \mathsf{Sig}_i(b)$ and $\mathsf{cert}'_b := \mathsf{cert}_b \| \mathsf{ticket}_i^b$ will be $(r+1,1)$-valid for any honest party P_j.*

Proof. For a party P_i in the committee for bit b (i.e. $\mathsf{res}_i^b = 1$), that receives a valid $(r,2)$-batch in Stage r, we prove that its updated batch will be $(r+1,1)$-valid for all honest parties in Stage $r+1$.

Since batch_b is considered $(r,2)$-valid by party P_i, it means that the following hold for batch_b according to Definition 13:

(i) It contains at least r valid distinct signatures and one of the signatures is from the sender P_s;

(ii)' For every valid k-signature $\mathsf{Sig}_k(b) \in \mathsf{batch}_b$, it holds that $G_k^{(i)} \geq g^* - 2r$; and $\tilde{g}_k^{(i)} \geq g^* - 2r$, where $G_k^{(i)}, \tilde{g}_k^{(i)}$ are from P_i's state.

(iii) cert_b is valid.

Now assume honest party P_j receives $(b, \mathsf{batch}'_b, \mathsf{cert}'_b)$ during the first round of Stage $r+1$ and checks whether batch'_b is a valid $(r+1,1)$-batch. We show that all three properties according to Definition 13 will hold for P_j.

(i) First, since P_i is honest and by assumption $\mathsf{Sig}_i(b) \notin \mathsf{batch}_b$, then P_j will accept $\mathsf{Sig}_i(b)$ as a valid, distinct signature. Moreover, given that the signing public keys are posted on the bulletin-PKI, all signatures that were valid for P_i are valid for P_j. Therefore, it holds that $\mathsf{batch}'_b = \mathsf{batch}_b \| \mathsf{Sig}_i(b)$ contains at least $r+1$ distinct signatures and one of them is from the sender P_s from property (i) of batch_b above.

(ii) For all signatures $\mathsf{Sig}_k(b) \in \mathsf{batch}_b$, from the graded agreement property of GC in $\mathsf{GCES.Gen}$ and $\mathsf{GCES.Toss}$, it holds that $|\tilde{g}_k^{(i)} - \tilde{g}_k^{(j)}| \le 1$ and $|G_k^{(i)} - G_k^{(j)}| \le 1$. Since $G_k^{(i)} \ge g^* - 2r$, it holds that $G_k^{(j)} \ge g^* - 2r - 1 = g^* - 2(r+1) + 1$. Furthermore, since P_i is honest, from graded validity of $\mathsf{GCES.Toss}$ it holds that for every honest party P_j: $G_i^{(j)} \ge g^* - 1 = g^* - 2(1+1) + 1$. Similarly, it holds that $g_k^{(j)} \ge g^* - 2r - 1 = g^* - 2(r+1) + 1$, and moreover, honest party P_j will have for honest party P_i that $\tilde{g}_i^{(j)} = g^*$. To conclude, for any honest party P_j it holds that for every valid k-signature $\mathsf{Sig}_k(b) \in \mathsf{batch}'_b$: $G_k^{(j)} \ge g^* - 2(r+1) + 1$, and $\tilde{g}_k^{(j)} \ge g^* - 2(r+1) + 1$.

(iii) From item (ii), for all $r \le g^*/2 - 1$, it holds that $\tilde{g}_k^{(i)}, \tilde{g}_k^{(j)}, G_k^{(i)}, G_k^{(j)}$ are strictly positive (in particular, the minimum grade is at least 1). Then, from the graded agreement property of $\mathsf{GCES.Toss}$, it holds that $\mathsf{pretags}_{l,k}^{(i),b} = \mathsf{pretags}_{l,k}^{(j),b}$ for all $k, l \in [n] : \mathsf{Sig}_k(b) \in \mathsf{batch}_b$. The same argument holds for $\mathsf{pk}_k^{(i)} = \mathsf{pk}_k^{(j)}$ for all $k \in [n] : \mathsf{Sig}_k(b) \in \mathsf{batch}_b$. Therefore, it follows that since cert_b is valid for P_i, it is also valid for P_j (since P_j will perform the same checks on the same values of pretags and public keys, contained in batch_b, and tickets, contained in cert_b, as P_i did). Finally, from the graded validity of $\mathsf{GCES.Toss}$ it holds that $\mathsf{pretags}_{k,i}^{(j),b} = \mathsf{pretags}_k^{(i),b}$, for all $k \in [n]$. So, ticket_i^b is valid for P_j with respect to $\mathsf{pretags}_k^{(i),b}$, since $\mathsf{res}_i^b = 1$. Therefore cert' is a valid certificate for batch'.

□

Proof of Theorem 1. We now prove Theorem 1. To this end, we show validity, consistency, and termination of our protocol in Figure 3.

Lemma 16. *Protocol* Π_{BC} *achieves validity with probability* $1 - \mathsf{negl}(\kappa)$.

Proof. In Stage 0, the sender P_s sends $(b_s, \mathsf{Sig}_s(b_s), \perp)$ to all parties. By Round 1, all honest parties have a valid $(1,1)$-batch for b_s and add b_s to their extracted sets. Since P_s is honest, the security of the signature scheme ensures that no malicious party can inject another validly signed message, so parties only elect themselves in a single committee for b_s and will not accept as valid any batch $\mathsf{batch}_{\bar{b}_s}$. Thus, validity holds, unless with negligible probability of the adversary forging the sender's signature. □

Lemma 17. *Protocol* Π_{BC} *achieves consistency with probability* $1 - \mathsf{negl}(\lambda) - \mathsf{negl}(\kappa)$.

Proof. Let P_i, P_j be two honest parties. Assume without loss of generality that P_i adds a value $b \in \mathsf{Extracted}_i$ before P_j. We show that by the end of the protocol $b \in \mathsf{Extracted}_j$. This is sufficient to prove consistency, since it implies that by the end of the protocol $\mathsf{Extracted}_i = \mathsf{Extracted}_j$ (set equality). Therefore the outputs of P_i, P_j for the broadcast protocol will match. We distinguish two cases.

Case 1. P_i adds b during Stage $r^* = g^*/2$: In that case, P_i must have received a valid $(g^*/2, 1)$-ballot for b. According to Definition 13, the batch in the ballot must contain valid signatures from at least $g^*/2$ parties and for each such party P_k it must hold that $\tilde{g}_k^{(i)} \geq 1$ and $G_k^{(i)} \geq 1$. Furthermore, the batch is accompanied by a valid certificate cert_b for it, i.e. a set of values ticket_k^b, for each P_k whose signature is in the batch. To prove consistency, we need that at least one of the signatures must have come from an honest party P_h with probability $1 - \mathsf{negl}(\lambda) - \mathsf{negl}(\kappa)$. This is true because, first, by Lemma 13, the committee cannot be larger than $B = g^*/2$ except with negligible probability—else there exists a direct reduction that wins the graded honest committee game with non-negligible probability—therefore the ballot contains the signatures of all parties in the committee, and there cannot exist be a distinct valid ballot from $g^*/2$ parties. Secondly, by Lemma 12, the adversary cannot corrupt the entire committee before the honest committee members reveal themselves, except with negligible probability—else there exists a direct reduction that wins the graded committee game with non-negligible probability—so the ballot contains a signature from at least an honest party P_h.

We now show that the honest committee member P_h would have sent its ballot to all parties by Round 1 of Stage r^*. If P_h added its own signature to a batch at Round 2 of a Stage $r < r^* - 1$ (and thus sent the updated ballot to all parties), then according to Lemma 15, P_i would have added b during $r + 1$, which contradicts our assumption. So, P_h added its own signature to a batch at Round 2 of Stage $r^* - 1$ and sent the updated ballot to all parties, including honest party P_j. So, by Lemma 15, it must hold that $b \in \mathsf{Extracted}_j$.

Case 2. P_i adds b during Stage $r < g^*/2$: First note that if P_i has $\mathsf{res}_i^b = 1$ then, in the second round of Stage r, P_i would send an updated ballot to all parties. By Lemma 15, any honest party P_j would add $b \in \mathsf{Extracted}_j$ in Round 1 of Stage $r + 1$.

Now assume that P_i adds b during Round 1 of Stage $r < g^*/2$ and $\mathsf{res}_i^b = 0$. P_i would then send the valid $(r, 1)$-ballot for b to all parties. By Lemma 14, any honest party considers that ballot to also be $(r, 2)$-valid. Then, by Lemma 12, at least one honest party P_h would have $\mathsf{res}_h^b = 1$ with probability $1 - \mathsf{negl}(\lambda) - \mathsf{negl}(\kappa)$; otherwise, the adversary would have to corrupt the entire committee before the honest committee members reveal themselves.

The honest committee member P_h however would have sent its ballot to all parties and all honest parties P_j would add $b \in \mathsf{Extracted}_j$ by Round 1 of Stage $r + 1$.

Therefore, in any case $b \in \mathsf{Extracted}_j$. □

Lemma 18. *Protocol* Π_{BC} *terminates.*

Proof. Termination holds by the termination of $\mathsf{GCES.Gen}$, $\mathsf{GCES.Toss}$ and $\mathsf{CES.TryElect}$, which act like an online setup, and by construction of the subsequent $g^*/2$ stages. □

Combining 16, 17 and 18 completes the proof of 1.

Round Complexity. Next, we give a bound on the round complexity of our protocol.

Theorem 2. *Consider a PPT adversary who can adaptively corrupt* $(1 - \epsilon)n$ *parties, for a constant* $\epsilon \in (0, 1)$. *Then, the protocol* Π_{BC} *(Figure 3) has round complexity* $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$.

Proof. Recall that $q_h = (p_1(\kappa, \Delta) + p_2(\kappa, n) - \Delta')/\Delta_r$ is the additional number of rounds (the slowest) honest party takes to solve the batch puzzle compared to Δ'/Δ_r, for $\Delta' = (4g^* + 2) \cdot \Delta_r$, and is a small constant number. Then, the first step of the broadcast protocol, before Stage 0, takes $6g^* + 3 + q_h$ rounds, as follows. $\mathsf{GCES.Gen}$ has to be ran before $\mathsf{GCES.Toss}$ and takes $2g^* + 1$ rounds; $\mathsf{GCES.Toss}$ takes $4g^* + 2$ rounds, but after the first $2g^* + 1$ of $\mathsf{GCES.Toss}$, honest parties can start

$\mathsf{CES.TryElect}$, which involves solving a batch puzzle with time complexity Δ', so the total number of rounds is $2g^* + 1 + \Delta'/\Delta_r + q_g = 6g^* + 3 + q_h$.

Stage 0 takes one round and the subsequent $g^*/2$ stages take a total of $g^* - 1$ rounds. As a result, the round complexity of broadcast in terms of g^* and q_h is $7g^* + 3 + q_h$. For $g^* = 2 \cdot \lceil 3 \log\left(\delta^{-1}\right)/\epsilon \rceil = O(\lambda/\epsilon)$ as described above, the broadcast protocol has round complexity $O\left(\log\left(\delta^{-1}\right)/\epsilon\right)$. For $\delta = \exp(-\omega(\log \lambda))$ negligible in the security parameter, we obtain a round complexity of $O(\lambda/\epsilon)$. $\square$

Communication Complexity. The communication complexity of broadcast is dominated by the $\mathsf{GCES.Toss}$ protocol, which is executed once for every broadcast instance. Therefore, $O(\mathsf{CC}_{\mathsf{Gen}} + \mathsf{CC}_{\mathsf{Toss}} + g^* \cdot \kappa \cdot n^3) = O(\kappa \cdot n^5 \cdot \log\left(\delta^{-1}\right)/\epsilon)$. For $\delta = \exp(-\omega(\log \lambda))$ negligible in the security parameter, we obtain a total communication complexity of $O(\kappa \cdot \lambda \cdot n^5/\epsilon)$. Using the version of $\mathsf{GCES.Toss}$ with gossiping, the total communication complexity reduces to $\tilde{O}(\kappa \cdot \lambda \cdot n^4/\epsilon)$.

6 Conclusion and Open Problems

We have constructed a synchronous, binary broadcast protocol, secure against adaptive adversaries who can corrupt up to $t = (1 - \epsilon)n$ parties. Our setup assumptions are limited to the plain public key model and a uniform common random string, and our computational assumptions are the discrete logarithm assumption and the existence of time-lock puzzles. Our protocol is the first to achieve broadcast in this model, while obtaining round complexity sublinear in n. This shows that sublinear-round broadcast protocols are achievable even without trusted setup. There are several remaining open questions, such as obtaining a similar protocol with improved communication efficiency or for messages of larger sizes. Another line of potentially interesting future work is whether one can obtain a similar result without requiring the assumption of a uniform CRS. A potential idea towards this would be to replicate our method for generating unpredictable values to construct graded unpredictable parameters to be used in the NIZKs and TLPs.

Acknowledgments. Julian Loss was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 507237585, and by the European Union, ERC-StG-2023-101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. Charalampos Papamanthou and Giorgos Tsimos were supported by the National Science Foundation. Benedikt Wagner was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 507237585.

References

[1] A. Abadi and A. Kiayias. Multi-instance publicly verifiable time-lock puzzle and its applications. In N. Borisov and C. Díaz, editors, *FC 2021, Part II*, volume 12675 of *LNCS*, pages 541–559. Springer, Heidelberg, Mar. 2021.

[2] I. Abraham, K. Nayak, and N. Shrestha. Communication and round efficient parallel broadcast protocols. Cryptology ePrint Archive, Paper 2023/1172, 2023. `https://eprint.iacr.org/2023/1172`.

[3] M. Andrychowicz and S. Dziembowski. PoW-based distributed cryptography with no trusted setup. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 379–399. Springer, Heidelberg, Aug. 2015.

[4] A. Bhat, N. Shrestha, Z. Luo, A. Kate, and K. Nayak. RandPiper - reconfiguration-friendly random beacons with quadratic communication. In G. Vigna and E. Shi, editors, *ACM CCS 2021*, pages 3502–3524. ACM Press, Nov. 2021.

[5] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters. Time-lock puzzles from randomized encodings. In M. Sudan, editor, *ITCS 2016*, pages 345–356. ACM, Jan. 2016.

[6] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, Aug. 2018.

[7] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, Aug. 2000.

[8] E. Boyle, R. Cohen, and A. Goel. Breaking the o $(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 319–330, 2021.

[9] J. Buchmann and S. Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2011.

[10] T.-H. H. Chan, R. Pass, and E. Shi. Sublinear-round byzantine agreement under corrupt majority. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 246–265. Springer, Heidelberg, May 2020.

[11] R. Cohen, J. Garay, and V. Zikas. Completeness theorems for adaptively secure broadcast. In *CRYPTO 2023*, 2023.

[12] J. Considine, M. Fitzi, M. Franklin, L. A. Levin, U. Maurer, and D. Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, 2005.

[13] P. Das, L. Eckey, S. Faust, J. Loss, and M. Maitra. Round efficient byzantine agreement from vdfs. In *International Conference on Security and Cryptography for Networks*, pages 139–160. Springer, 2024.

[14] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[15] J. Dujmovic, R. Garg, and G. Malavolta. Time-lock puzzles with efficient batch solving. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–341. Springer, 2024.

[16] P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.

[17] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[18] M. Fitzi and J. B. Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *International Symposium on Distributed Computing*, pages 449–463. Springer, 2009.

[19] A. Flamini, R. Longo, and A. Meneghetti. Multidimensional byzantine agreement in a synchronous setting. *Applicable Algebra in Engineering, Communication and Computing*, pages 1–19, 2022.

[20] C. Freitag, I. Komargodski, R. Pass, and N. Sirkin. Non-malleable time-lock puzzles and applications. In K. Nissim and B. Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 447–479. Springer, Heidelberg, Nov. 2021.

[21] J. A. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, Oct. 2007.

[22] J. A. Garay, A. Kiayias, N. Leonardos, and G. Panagiotakos. Bootstrapping the blockchain, with applications to consensus and fast PKI setup. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 465–495. Springer, Heidelberg, Mar. 2018.

[23] J. A. Garay, A. Kiayias, R. M. Ostrovsky, G. Panagiotakos, and V. Zikas. Resource-restricted cryptography: Revisiting MPC bounds in the proof-of-work era. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2020.

[24] R. Hou, H. Yu, and P. Saxena. Using throughput-centric byzantine broadcast to tolerate malicious majority in blockchains. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1263–1280. IEEE, 2022.

[25] J. Katz and C.-Y. Koo. On expected constant-round protocols for byzantine agreement. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, Aug. 2006.

[26] V. King and J. Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In A. W. Richa and R. Guerraoui, editors, *29th ACM PODC*, pages 420–429. ACM, July 2010.

[27] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *17th SODA*, pages 990–999. ACM-SIAM, Jan. 2006.

[28] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, a. shelat, and E. Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.

[29] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[30] A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.

[31] Y. Liu, Q. Wang, and S.-M. Yiu. Towards practical homomorphic time-lock puzzles: Applicability and verifiability. In V. Atluri, R. Di Pietro, C. D. Jensen, and W. Meng, editors, *ESORICS 2022, Part I*, volume 13554 of *LNCS*, pages 424–443. Springer, Heidelberg, Sept. 2022.

[32] G. Malavolta and S. A. K. Thyagarajan. Homomorphic time-lock puzzles and applications. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, Aug. 2019.

[33] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[34] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.

[35] P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. R. Weippl. RandRunner: Distributed randomness from trapdoor VDFs with strong uniqueness. In *NDSS 2021*. The Internet Society, Feb. 2021.

[36] S. Srinivasan, J. Loss, G. Malavolta, K. Nayak, C. Papamanthou, and S. A. K. Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In A. Boldyreva and V. Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 554–584. Springer, Heidelberg, May 2023.

[37] S. A. K. Thyagarajan, G. Castagnos, F. Laguillaumie, and G. Malavolta. Efficient CCA timed commitments in class groups. In G. Vigna and E. Shi, editors, *ACM CCS 2021*, pages 2663–2684. ACM Press, Nov. 2021.

[38] G. Tsimos, J. Loss, and C. Papamanthou. Gossiping for communication-efficient broadcast. In Y. Dodis and T. Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 439–469. Springer, Heidelberg, Aug. 2022.

[39] J. Wan, A. Momose, L. Ren, E. Shi, and Z. Xiang. On the amortized communication complexity of byzantine broadcast. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 253–261, 2023.

[40] J. Wan, H. Xiao, S. Devadas, and E. Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 412–456. Springer, Heidelberg, Nov. 2020.

[41] J. Wan, H. Xiao, E. Shi, and S. Devadas. Expected constant round byzantine broadcast under dishonest majority. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411. Springer, Heidelberg, Nov. 2020.