Sublinear-Round Dishonest-Majority Broadcast without Trusted Setup

Andreea B. Alexandru[1], Julian Loss[2], Charalampos Papamanthou[3] and Giorgos Tsimos[4]

[1] Duality Technologies
aalexandru@dualitytech.com
[2] CISPA Helmholtz Center for Information Security
lossjulian@gmail.com
[3] Yale University
charalampos.papamanthou@yale.edu
[4] University of Maryland, College Park
tsimos@umd.edu

Abstract. Byzantine broadcast is one of the fundamental problems in distributed computing. Many practical applications from secure multi-party computation to consensus mechanisms for blockchains require increasingly weaker trust assumptions, as well as scalability for an ever-growing number of users, which rules out existing solutions with linear number of rounds or trusted setup requirements. In this paper, we propose the first *sublinear-round* and *trustless* Byzantine broadcast protocol. Unlike previous sublinear-round protocols, our protocol does not assume the existence of a trusted dealer who honestly issues keys and common random strings to the parties, but only relies on an untrusted bulletin-public key infrastructure and time delay assumptions.

Our protocol is based on a new cryptographic protocol called *verifiable graded consensus*, designed to act as an untrusted online setup, enabling n parties to almost agree on shared random strings. We propose an implementation of the verifiable graded consensus protocol using verifiable delay functions and random oracles, which is then used to run a committee-based Byzantine protocol, similar to Chan et al. (PKC 2020), in an unbiased fashion. Finally, we obtain a polylog-round trustless Byzantine broadcast with amortized communication complexity of $\tilde{O}(n^2)$, which can be further improved to $\tilde{O}(n)$ per instance for multiple instances of parallel broadcast.

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 $t < n$ Byzantine corruptions: (1) *consistency*: all honest parties output the same value, and (2) *validity*: all honest parties output v if the sender honestly follows the protocol. Broadcast is essential in ensuring a consistent view

between parties, and has important applications in protocols for multiparty computation (MPC), verifiable secret sharing (VSS) and state machine replication (SMR). To reduce the overhead of such applications, a long line of works has focused on minimizing two metrics: (1) *round-efficiency*: how many synchronous rounds does the protocol run for? and (2) *communication-efficiency*: how many bits does the protocol exchange? By the famous works of Dolev and Strong [23], and Dolev and Reischuk [22], $t+1$ synchronous rounds and $\Omega(n \cdot t)$ communication are 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 these lower bounds. 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. While major progress has been made, existing round-efficient protocols fall into one of two unsatisfactory categories.

Protocols of the first category [15,52,53] 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.

Due to the lower bound of Lamport, Pease and Shostak [45,40], we cannot hope to avoid some form of cryptographic setup altogether. A natural question is whether it is possible to design a round-efficient broadcast protocol for the dishonest majority setting in the *plain public key model*. Rather than relying on a trusted dealer, protocols for this model allow 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 before commencing protocol execution. On the downside, protocols of this second category by Garay *et al.* [30] and Fitzi and Nielsen [27] achieve security only for a very small margin where the number t of corrupted parties must satisfy $t - n/2 \leq o(n)$.

In this work, we significantly improve over the state of the art. Concretely, we design broadcast protocols in the plain public key model that are secure against an adaptive dishonest majority of $t \leq (1-\epsilon)n$, for constant $\epsilon \in (0,1)$, except for negligible probability in a statistical security parameter λ. Our contributions assume random oracles, a bulletin-board public key infrastructure, and the existence of delay functions, and can be summarized as follows.

- **Broadcast.** We provide a stand-alone binary broadcast protocol with $\tilde{O}(\lambda)$ round complexity and $\tilde{O}(n^3)$ total communication complexity. Our protocol compares favorably to the Dolev-Strong protocol [23], as it reduces the round complexity from $O(n)$ without introducing trusted dealers at the cost of only logarithmic increase in the communication cost compared to $O(n^3)$.
- **Amortized Broadcast.** We show how to amortize the setup cost, when multiple broadcast instances have to be run. This problem comes up naturally in many applications in which broadcast is relevant, e.g., VSS, MPC, SMR. We obtain an amortized broadcast communication complexity of $\tilde{O}(n^2)$

at a round complexity of $\tilde{O}(\lambda)$ using n instances. This compares favorably to Chan *et al.* [15], which achieves the same complexity using a trusted setup. Our techniques do not require these instances to be run in parallel.

We can even further amortize the communication complexity per sender instance if we are willing to consider the problem of *parallel broadcast* (n instances of broadcast run in parallel). In this case, we show that it is possible to run n instances of parallel broadcast at an amortized cost of $\tilde{O}(n^2)$ per instance. (This amounts to $\tilde{O}(n)$ per single sender instance of broadcast within each parallel invocation.)

Our broadcast protocols are build over a **Verifiable Graded Consensus on Random Strings** protocol, which provides an online setup for graded shared randomness that has $O(\lambda)$ round complexity and $\tilde{O}(n^4)$ communication complexity. We show how to reduce the communication to $\tilde{O}(n^3)$ by using the gossip techniques of Tsimos *et al.* [51], with a round complexity polylogarithmic in n.

Our solution relies on the random oracle heuristic and delay functions, but we believe this to be a minor restriction. Indeed, many works have considered setup-free protocols in the random oracle model and resource-restricted cryptography assumptions, e.g., Andrychowicz and Dziembowski [3] and Garay *et al.* [31,32].

1.1 Technical Overview

We now provide an overview of our techniques.

Chan *et al.*'s Protocol. Our starting point is the construction of Chan *et al.* [15] which achieves sublinear round complexity against $(1-\epsilon)n$ adaptive corruptions, for constant $\epsilon > 0$. At a high level, it delegates the message signing steps in the Dolev-Strong [23] protocol to a small ad-hoc committee of roughly $O(\lambda)$ parties. (To counter adaptive corruptions, there is one committee per bit.) If at least one party in the committee is honest, their protocol achieves $O(\lambda)$ round complexity and $O(\lambda^2 n^2)$ communication. 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 $y \leftarrow F(sk, v)$, 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.

In [15], 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.

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. Unfortunately, this 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.

To break this circularity, our approach follows the template (see [24,37,3,32]) of boosting a weak form of agreement on an unpredictable string to a strong form of agreement on an arbitrary sender input. Compared to prior work, however, we face a major challenge: we are in a dishonest majority setting with any constant fraction of adaptive corruptions. This disqualifies typical approaches such as VSS [24,37] (which requires an honest majority) or reducing the number of Byzantine parties to run an honest-majority broadcast protocol [30,27] (which requires either $t - n/2 \leq o(n)$ or trusted setup). Thus, we have to develop novel techniques in order to reach our objective.

Verifiable Randomness. Let us describe the difficulty of agreeing on an unpredictable and unbiased random string id. Consider a naïve protocol where parties locally generate random seeds and share them (these could also be their public keys from the bulletin-PKI); 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. These issues can be overcome using, e.g., VSS (if in honest majority), which commits a party P to a value s in such a way that s can be forcibly revealed without P's contribution. Our idea is to try to mirror the properties of VSS in a *dishonest* majority setting, which would be equivalent to associating to P an unpredictable random value that can be obtained and verified by the honest parties even without the participation of P. More exactly, we will ensure that a dishonest party P's value which passes the honest verification is guaranteed to be unpredictable and random.

To this end, we assume the existence of a *verifiable delay function* (VDF) D. Critically, D cannot be evaluated faster than within a certain predetermined time Δ, even when given access to parallel computation. This suggests the following approach: first, have parties agree on locally sampled random strings, then evaluate D on their concatenation S and set the output of D to be the final identifier id that will be used as an input to the VRF. If Δ is larger than the time it takes to agree on S, it should be impossible for Byzantine parties to pick their contributions to S in a manner that significantly biases id. Converting this approach into a working solution, however, turns out to be very challenging. While there are prior works [50,49,20] on random beacons using 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.

To avoid another circular problem—that of requiring a trusted setup for the VDFs—we base D on class groups of imaginary order [54]. Crucially, the order of the class group should not known by any party in our construction, and it suffices to generate the parameters from a *biased* discriminant. In particular, the solution of applying a random oracle over the concatenation of all public keys, which was too weak for generating the randomness to be input to the VRF, is now sufficient for the veriafiable delay function setup.

Resolving the Remaining Challenges. Our sketch above on how to use a VDF D as a "substitution" for VSS in the dishonest majority setting implicitly assumed that parties can broadcast their individual contributions to the string S to be input to D. Naturally, we need to find a means of replacing this with a suitable primitive that can be achieved using only a plain public key setup.

Following classical works in the area of round-efficient agreement, we consider a weaker primitive commonly referred to as *gradecast* (GC). In GC, parties output a grade g together with the output v. Intuitively, g indicates their confidence in v. 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 v (if any), and (ii) grades of honest parties differ by at most one. The solution is not as straightforward as simply replacing all broadcast channels with gradecasts. We need all Byzantine parties to evaluate D on some input S that includes the contribution of at least one honest party. To this end, we study a generalized notion of *moderated gradecast* (Mod-GC), introduced in [37].

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 GCs received initially with low grades, our grading scheme takes into account both the grades of the original senders P and the efforts of M in relaying those GCs. The outcome is a scenario in which, for all parties M, the honest parties attain graded agreement on a string id which they attribute to M along with a grade g, regardless of the honesty of P. Unpredictability follows because a dishonest party M can only obtain an id that holds a positive grade in the view of any honest party if id 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). We call this protocol, along with the use of the verifiable delay function, a *verifiable graded consensus on random strings*.

Wrapping up. Now that all parties have unpredictable *id*s and associated grades in the view of honest parties, we can use them in Chan *et al.*'s protocol. We set the maximum grade equal to the number of stages in [15]. 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 round. 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 grade at least a value depending on r. Then, these signatures will have a sufficiently large grade in the next round as well, and can safely be forwarded by P before accepting b. The result is a sublinear-round broadcast protocol.

Communication Complexity Reduction. At this point, we have reached our primary goal of achieving a sublinear round complexity in the number of parties. In the remaining, we turn our attention toward reducing the communication complexity. Concretely, we show it is possible to amortize the expensive setup when multiple instances of our protocol need to be run. Note that many applications of broadcast such as MPC and SMR naturally require repeated invocations of broadcast. To this end, we first show how to amortize the communication complexity to $\tilde{O}(n^2)$ bits per instance when $O(n)$ instances at $\tilde{O}(\lambda)$ rounds are run. We stress that it is not necessary to run the instances in parallel.

To achieve this, we adapt the gossip-based techniques introduced by Tsimos *et al.* [51] to reduce the communication complexity when running many moderated gradecasts in parallel. However, adapting the results of [51] to non-binary messages is highly non-trivial. In our case, parties have to drop conflicting non-binary messages in order to reduce communication, while ensuring the conflicts reach all honest parties in due time to guarantee a difference of at most one in their grades. Using these techniques reduces the communication complexity for the verifiable graded consensus (and thus, for broadcast as well) to cubic in the number of parties n, with a round complexity polylogarithmic in n.

The online setup can be used to obtain random *id*s for multiple broadcast instances via a random oracle. Therefore, the amortized communication cost for a broadcast instance is quadratic, over n instances. Extending the techniques of Tsimos *et al.* even further, amortized complexity per sender instance can be improved all the way down to $\tilde{O}(n)$, given that we run $O(n)$ instances of *parallel broadcast*, i.e., where all senders send n values in parallel. This problem, also known as *interactive consistency*, is of central importance in several of the aforementioned applications, in particular VSS and MPC. In the latter type of protocol, it is frequently the case that up to $O(n)$ parties distribute some value or accuse each other of incorrect behaviour within the same round.

Open questions. While we weaken the setup assumptions compared to a trusted dealer, our solutions require random oracles. We leave open to design sublinear round broadcast in the standard model without trusted setup. In particular, this requires to reach graded agreement on the random strings fed to the VRFs without relying on random oracles, which rules out our current approach.

One of the main bottlenecks is that in order to prevent biasing from an adversary controlling $t = O(n)$ parties, honest parties might need to evaluate $O(n)$ VDFs, which is not $o(n)$-round. Using polylogarithmic samplers has the potential to reduce the number of VDF evaluations to sublinear in n, but comes at an enormous increase in the required values of n. Finally, we would also need VDF and VRF constructions in the standard model that do not rely on a trusted setup.

1.2 Related Work

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 celebrated FLP result [26] disallows any broadcast protocol in the asynchronous setting tolerating $n/3$ or more corruptions, so we focus on the synchronous setting. Dolev and Strong [23] give a protocol against an adaptive dishonest majority $t < n$ with $O(n^3)$ total communication complexity. A line of work initiated by King *et al.* [39,38] and Boyle *et al.* [10] proposed protocols with reduced communication complexity of $\tilde{O}(n^{3/2})$ but only for honest super-majority $t < n/3$. Momose and Ren [43] also proposed a protocol with reduced communication complexity $\tilde{O}(n^2)$ but for honest majority $t < n/2$. All these works require a linear number of rounds. Abraham *et al.* [1] achieve constant rounds for broadcast but in the honest majority with trusted setup. In a recent work, Abraham *et al.* [2] achieve expected constant rounds for Parallel Broadcast, with $\tilde{O}(\kappa n^3)$ expected communication complexity against a static adversary in the honest majority setting with bulletin-board PKI.

Our work, like most of the ones described here, considers an atomic send model 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.* [18] investigate the feasibility of property-based and simulation-based adaptive broadcast (also assuming TLPs).

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.* [30] achieved $O((2t-n)^2)$ rounds, and Fitzi and Nielsen [27] achieved $O(2t - n)$ rounds, against a strongly adaptive adversary.

Chan *et al.* [15] 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. The adversary is weakly adaptive, meaning it cannot perform after-the-fact removals. Wan *et al.* [53] 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.* [52] tolerates stronger adversaries that can also erase messages. In [52], 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.* [36] describe a blockchain that tolerates dishonest majority, loosely based on the Chan *et al.* broadcast protocol [15], proof of stake, proof of work in the random oracle model (ROM). Other works in the adaptive dishonest majority case [51,44] studied the amortized communication complexity of protocols over a number of broadcast instances, but achieve a linear number of rounds. Tsimos *et al.* [51] study the parallel version of broadcast for $t \leq (1-\epsilon)n$, where every party has some input and, after the protocol, the properties of broadcast need to hold per each separate sender. By combining gossiping and a variation of the protocol from Chan *et al.* [15], they achieve parallel broadcast with amortized communication $\tilde{O}(n^2)$ assuming bulletin-board PKI, or $\tilde{O}(n)$ assuming trusted PKI. More recently, Momose *et al.* [44] study a version of multi-shot broadcast, where there are multiple sequential broadcast calls. They achieve a synchronous protocol with amortized $O(n^2)$ communication over a number of consecutive executions for any $t < n$, assuming only signatures, and against strongly adaptive adversaries. Blum *et al.* [6] present communication bounds in the number of messages required for dishonest majority in the static corruptions in the stronger adversarial model and for adaptive corruptions in the weakly adaptive model.

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 [24,19,37,30] for instance. Recently, in the honest majority case, [28] generalized graded consensus to the multi-dimensional case which deals with a vector of inputs.

Time-lock puzzles (TLP) [48], timed commitments [9] and verifiable delay functions [7] 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 VDFs have been proposed in [7,46,54,21,35]. VDFs are currently used in blockchain applications such as Ethereum and Chia [13,17].

In the context of timing assumptions and broadcast, Das *et al.* [20] 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.* [20]

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. Moreover, their constructions heavily rely on $t < n/2$.

The use of VDFs for multi-party unbiased randomness generation was first exemplified by Lenstra *et al.* [41]. Constructing randomness from VDF/TLP with transparent setup in ROM, but assuming broadcast, is also addressed by Bhat *et al.* [5], who tolerate $t < n$ only if the adversary is covert, and by Thyagarajan *et al.* [50] who tolerate $t < n$. Freitag *et al.* [29] 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).

We compare the relevant related work and our results in Table 1.

Table 1. Comparison between works for synchronous broadcast in the dishonest majority case. Here, n is the number of parties, ϵ is the honest *constant* fraction, κ is a computational security parameter and λ is a statistical security parameter (not necessarily the same across all entries). If unspecified, the calls in the last column can be either parallel or sequential. *means in the strong adaptive adversarial model, while the others are in the weak adaptive adversarial model.

Protocol	Corruptions t	Assumptions	Rounds	Communication	BC instances
Dolev & Strong [23]	$< n^*$	bulletin-PKI	$O(n)$	$O(n^3 \cdot \kappa)$	1 call
Chan *et al.* [15]	$< (1-\epsilon)n$	trusted-PKI	$O(\lambda)$	$O(n^2 \cdot \lambda\kappa)$	1 call
Wan *et al.* [52]	$< (1-\epsilon)n^*$	trusted-PKI	$O(\lambda)$	$\tilde{O}(n^3 \cdot \lambda\kappa)$	1 call
Wan *et al.* [53]	$< (1-\epsilon)n$	trusted-PKI	$O(\lambda)$	$\tilde{O}(n^4 \cdot \kappa)$	1 call
Tsimos *et al.* [51]	$< (1-\epsilon)n$	bulletin-PKI	$O(n \log n)$	$\tilde{O}(n^2 \cdot \lambda\kappa)$ amortized	n parallel calls
Tsimos *et al.* [51]	$< (1-\epsilon)n$	trusted-PKI	$O(\lambda \log n)$	$\tilde{O}(n \cdot \lambda^3\kappa)$ amortized	n parallel calls
Momose *et al.* [44]	$< n^*$	bulletin-PKI	$O(n)$	$O(n^2 \cdot \kappa)$ amortized	n^2 sequential calls
Π_{BC} (Thm. 3)	$< (1-\epsilon)n$	bulletin-PKI, ROM, delay functions	$O(\lambda)$	$O(n^4 \cdot \lambda\kappa)$	1 call
Π'_{BC} (Thm. 4)	$< (1-\epsilon)n$	bulletin-PKI, ROM, delay functions	$O(\lambda \log n)$	$\tilde{O}(n^3 \cdot \lambda^2\kappa)$	1 call
Π'_{BC} (Thm. 5.1)	$< (1-\epsilon)n$	bulletin-PKI, ROM, delay functions	$O(\lambda \log n)$	$\tilde{O}(n^2 \cdot \lambda^2\kappa)$ amortized	n calls
Π_{BC} (Thm. 5.2)	$< (1-\epsilon)n$	bulletin-PKI, ROM, delay functions	$O(\lambda)$	$O(n^2 \cdot \lambda\kappa)$ amortized	n^2 calls
Π'_{BC} w. gossiping (Thm. 5.3)	$< (1-\epsilon)n$	bulletin-PKI, ROM, delay functions	$O(\lambda \log n)$	$\tilde{O}(n \cdot \lambda^3\kappa)$ amortized	n calls of n parallel calls each

1.3 Note

An extensive update of this work, which includes additional authors can be found in https://eprint.iacr.org/2024/770.

2 Model and Preliminaries

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 and verifiable random function) 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.

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.

Security parameters. We assume all cryptographic hash functions are modeled as *random oracles* (ROs). This means that for any input x in the domain, a hash function H returns $H(x)$ if it was queried on x before and otherwise returns a random value from the codomain. For a computational security parameter κ, we assume that the hash function outputs, signatures, verifiable delay function outputs and verifiable random function outputs are of size $O(\kappa)$. For a statistical security parameter λ, we use a failure probability $\delta \in (0,1)$ that is negligible in λ but independent of n, e.g., $\log(1/\delta) = \mathsf{polylog}(\lambda)$. In general, the computational security parameter is larger than the statistical security parameter. An extended discussion on the security parameters is given in Section 6.

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.

We assume the adversary accesses a probabilistic polynomial-time (ppt) machine. Looking ahead, the adversary is Δ-limited, i.e., to evaluate a VDF with difficulty parameter Δ the adversary's machine takes sequential time at least Δ even with $\mathsf{poly}(\Delta, \kappa)$ processors. Honest users might have slightly weaker machines.

Signatures. We use the notation $\mathsf{sig}_i(m)$ to denote a signature of party P_i using sk_i on message m and $\mathsf{ver}_i(s, m)$ to denote the verification of signature s

on message m using public key pk_i. We assume idealized signatures that achieve *perfect correctness*: for any message m, it holds that $\mathsf{ver}_i(\mathsf{sig}_i(m), m) = 1$, 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 such that $\mathsf{ver}_i(s, m) = 1$.

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. We focus our presentation on binary broadcast, where input values are bits, but it can be extended to multi-bit values.

Broadcast. Introduced in the seminal work by Lamport *et al.* [40], *broadcast* ensures agreement of honest parties on a sender's message.

Definition 1. *A protocol executed by parties $P_1, P_2, \ldots, P_n$, where a sender $P^* \in \{P_1, \ldots, P_n\}$ begins holding input x^*, is a* broadcast *protocol if the following notions 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 and Moderated Gradecast. *Gradecast*, introduced by Feldman and Micali in [25] and generalized for an arbitrary grade by Garay *et al.* [30], is a relaxation of broadcast, where honest parties are allowed to disagree by a "small amount".

Definition 2. *A protocol executed by parties $P_1, P_2, \ldots, P_n$, where a sender $P^* \in \{P_1, \ldots, P_n\}$ begins holding input x^*, is a g^*-*gradecast *protocol if the following notions hold:*

(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.

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 3. *A protocol executed by parties $P_1, P_2, \ldots, P_n$, 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.

In this work, we are also interested in the parallel version of moderated gradecast, where each party acts as an initial sender and as a moderator for all other senders. Each party will output a vector of values and grades, with each tuple corresponding to a different party.

Verifiable Random Functions. Verifiable Random Functions (VRFs), introduced by Micali *et al.* [42] are functions whose output is unique and pseudorandom and, moreover, the validity of the function evaluation relative to a binding commitment can be efficiently proved and verified.

A VRF consists of algorithms $\mathsf{VRF.Gen}$, $\mathsf{VRF.Prove}$ and $\mathsf{VRF.Verify}$, and should satisfy *Uniqueness*, *Provability*, *Pseudorandomness* (Definition 8 in Appendix A). Usually, the literature using VRFs considers that the key generation is run at a trusted setup. We are interested in a variation where the key pair is generated by a potentially malicious party, yet we still want to satisfy uniqueness, provability and pseudorandomness in a modified setting. Chen and Micali [16] and Gilad *et al.* [33] also require that the VRF security holds for adaptive adversaries and adversarially generated key pairs, as long as the public keys have been chosen in advance of the seeds, but do not provide a formal definition. Their VRF construction uses random oracles and unique signatures. Goldberg *et al.* [34] discuss this issue and propose VRF constructions based on RSA or elliptic curves in the random oracle model, which have a validation predicate for the public key in order to detect maliciously generated key pairs.

In this work, we follow the approach suggested in Goldberg *et al.* First, we require a VRF to have an additional efficient predicate $\mathsf{VRF.Validate}$ to check that the public key corresponds to an admissible secret key. Second, we formalize the property of input indistinguishability of partially random inputs in the context of a maliciously generated key pair, which we call *extended pseudorandomness* (Appendix A), and we require that the VRF satisfies it.

In our constructions, we can either instantiate the VRF via ROs and unique signatures, e.g., BLS [8] with the validity predicate $\mathsf{VRF.Validate}_{\mathsf{PK}}(1^\kappa) = 1$ if $\mathsf{PK} \neq 1$ and 0 otherwise, or we can use the elliptic curve-based VRF via ROs from [34] with the validity predicate described there.

Verifiable Delay Functions. Verifiable Delay Functions (VDFs), defined by [7], are functions with a unique unpredictable output that can only be evaluated after a sequential number of steps, and moreover, the validity of the function evaluation can be efficiently proved and verified.

A VDF consists of the algorithms $\mathsf{VDF.Setup}$, $\mathsf{VDF.Eval}$ and $\mathsf{VDF.Verify}$, and should satisfy *Correctness*, *Soundness*, *Sequentiality* (Definition 9 in Appendix A).

We adapted the σ-sequentiality definition from Boneh *et al.* [7] and Wesolowski [54] to the following case: no adversary is able to compute an output for VDF.Eval on the honest random challenge concatenated with an adversarial input in parallel time $\sigma(\Delta) < \Delta$, even with up to a polynomially large number of parallel processors and after a potentially large amount of precomputation. The sequentiality property implies the *unpredictability* of the VDF output, and in the RO model, any unpredictable string can be used to extract an unpredictable κ-bit uniform random string.

Wesolowsky [54] gives a construction for a VDF without trusted setup, based on class groups of unknown order, which we adopt in our constructions.

Verifiable Graded Randomness. One of our goals in this paper is to generate strings that act as random beacons [47] that could also be validated by the network. First, these strings should satisfy unpredictability and bias resistance (which can be extended to indistinguishability from random in RO model) and termination, as random beacons do. Second, we want honest parties to agree between themselves on the beacons associated to every other party, but since broadcast is not available, we relax this requirement to graded agreement. Third, we want these beacons to be verifiable in the sense that once a party outputs such a string, the honest participants in the network should be able to validate that string as belonging to that party based on their previous information. We call a string generation protocol satisfying the requirements above a *verifiable graded consensus on random strings* protocol.

Definition 4. *A protocol executed by parties $P_1, \ldots, P_n$ is a* verifiable graded consensus on random strings *(VGC) with maximum grade g^* composed of algorithms Gen, Process, Verify and protocol Toss:*

- *Gen(1^κ) outputs public parameters pp;*
- *Toss(pp) outputs n pairs $(x_i^{(j)}, g_i^{(j)})$ to party P_i;*
- *Process(pp, x) outputs (w, ρ);*
- *Verify(pp, x, w, ρ) outputs 0 or 1.*

Gen is a ppt algorithm, Verify is a deterministic algorithm, while Process can be a ppt algorithm but w is a deterministic evaluation of x. A verifiable graded consensus protocol on random strings should satisfy the following notions, for any i, j, k:

(Graded consensus properties)

(Graded Validity) If party P_j is honest, then every honest party P_i outputs $(x^{(j)}, g_i^{(j)})$ for some $x^{(j)}$ and for $g_i^{(j)} \in \{g^* - 1, g^*\}$.

(Graded Agreement) Let P_i, P_k be two honest parties outputting $(x_i^{(j)}, g_i^{(j)})$, $(x_k^{(j)}, g_k^{(j)})$, for some j. If $g_i^{(j)} \geq 2$, then $x_i^{(j)} = x_k^{(j)}$ and $|g_i^{(j)} - g_k^{(j)}| \leq 1$. If $g_i^{(j)} = 1$, then either $x_i^{(j)} = x_k^{(j)}$ or $g_k^{(j)} = 0$.

(Termination) An honest party P_i terminates with output $(x_i^{(j)}, g_i^{(j)})$ for every $j \in [n]$, after executing Toss, and terminates with (w_i, ρ_i) after executing Process.

(Verifiable randomness properties)

(Indistinguishable Randomness) If all honest grades associated to P_i (and implicitly to w_i) are strictly positive, then w_i is indistinguishable from random, i.e., no ppt adversary $\mathcal{A}$ can win the Indistinguishable Randomness game with probability more than $\mathsf{negl}(\kappa)$.
(Correctness) For all $\kappa, pp \leftarrow \mathsf{Gen}(1^\kappa)$, if $(x_i^{(j)}, \cdot) \leftarrow \mathsf{Toss}(pp)$ and $(w_j, \rho_j) \leftarrow \mathsf{Process}(pp, x_i^{(j)})$, then $\mathsf{Verify}(pp, x_i^{(j)}, w_j, \rho_j) = 1$.
(Soundness) For a ppt adversary $\mathcal{A}$, it holds that for all $pp \leftarrow \mathsf{Gen}(1^\kappa)$:

$$\Pr\left[\begin{matrix}\mathsf{Verify}(pp, x_i^{(j)}, w_j, \rho_j) = 1 \\ (w_j, \cdot) \neq \mathsf{Process}(pp, x_i^{(j)})\end{matrix} \;:\; \begin{matrix}(x_i^{(j)}, \cdot) \leftarrow \mathsf{Toss}(pp) \\ (w_j, \rho_j) \leftarrow \mathcal{A}^{\mathsf{Toss},\mathsf{Process}}(\kappa, pp)\end{matrix}\right] \leq \mathsf{negl}(\kappa).$$

Note that Process is deterministic with respect to the output w, so it follows that the outputs of Process on the output strings of Toss also satisfy graded agreement and graded validity.

Since the verifiable graded consensus on random strings protocol can be of independent interest, above we gave a more general definition where the adversary is not limited in sequential processing time. Nevertheless, in this paper, we are working under the assumption of existence of delay functions, hence we particularize Definition 4 to this setting. The only changes in the definition are that (i) a time parameter Δ will be part of the public parameters, (ii) the adversary is limited to run in time $O(\mathsf{poly}(\Delta, \kappa))$ instead of only being limited to ppt, and (iii) we particularize the indistinguishable randomness property to:

(σ-Indistinguishable Randomness) For $\sigma(\Delta) \leq \sigma'(\Delta)$, where Process is σ'-sequential, no pair of randomized algorithms $\mathcal{A}_0$, running in time $O(\mathsf{poly}(\Delta, \kappa))$, and $\mathcal{A}_1$, running in parallel time $\sigma(\Delta)$ can win the Δ-Indistinguishable Randomness game with probability more than $\mathsf{negl}(\kappa)$.

The Δ-indistinguishable randomness game captures both the unpredictability and unbiasedness notions, as in [4,50], and is based on the σ-sequentiality definition, which says that the adversary cannot run in more time than $\sigma(\Delta)$.

Definition 5. *[Δ-Indistinguishable Randomness Game] An adversary $\mathcal{A} := (\mathcal{A}_0, \mathcal{A}_1)$ and a challenger $\mathcal{C}$ play the next game with security parameter κ and time Δ:*

1. *$\mathcal{C}$ sends to the adversary Δ.*
2. *$\mathcal{A}$ selects the parties it wants to corrupt and sends their identities to $\mathcal{C}$.*
3. *$\mathcal{C}$ computes $pp \leftarrow \mathsf{Gen}(1^\kappa, \Delta)$ and sets the key pairs for the honest parties and sends them to $\mathcal{A}$.*
4. *$\mathcal{A}$ chooses the public keys for the corrupted parties and sends them to $\mathcal{C}$.*
5. *$\mathcal{C}$ discards the parties whose public keys are not valid.*
6. *$\mathcal{C}$ and $\mathcal{A}$ execute the protocol $\mathcal{A}_0^{\mathsf{Toss}(pp)}$; $\mathcal{A}$ can corrupt additional parties.*
7. *The protocol ends when all remaining honest parties $P_i \in$ Honest have generated output $(\{x_i^{(j)}, g_i^{(j)}\}, w_i)$. Denote by st the state of $\mathcal{A}$ obtained so far.*
8. *For each j, $\mathcal{A}$ computes $w_{\mathcal{A}}^{(j)} \leftarrow \mathcal{A}_1^{\mathsf{Process}(pp,\cdot)}(pp, st)$.*

The adversary $\mathcal{A}$ wins the game if for at least one index j, $w_{\mathcal{A}}^{(j)} = w^{(j)}$ for $(w^{(j)}, \cdot) \leftarrow \mathsf{Process}(pp, x_i^{(j)})$ and for $g_i^{(j)} \geq 1$ for any honest party $P_i \in$ Honest.

In other words, the adversary should not be able to guess the honest party P_i's output from $\mathsf{Process}(\mathsf{pp}, x^{(i)})$, and even for corrupted parties P_j for which it manipulated $x^{(j)}$, it should not be able to guess the output $\mathsf{Process}(\mathsf{pp}, x^{(j)})$ as long as all honest parties P_i have strictly positive grades for P_j. The indistinguishable randomness game can be easily obtained from Definition 5 by removing the dependence on Δ.

3 Emulating a Common Random String

Consider that each party P_i, $i \in [n]$, samples a seed (honest parties sample random seeds). Each honest party needs to create and "almost" agree on aggregate strings based on these seeds, via a protocol we call *moderated gradecast*. The output string is then input to a VDF, and the VDF evaluation produces an unpredictable and unbiased random string that serves as a common random string. However, parties might not all hold the same common value, because the adversary can send different seeds to the honest parties. The grades that honest parties hold for the final strings will quantify how much confidence they have that their output strings coincide. In particular, grades greater than 2 certify that honest parties have the same VDF input and hence, VDF evaluation.

We start by describing the Toss protocol, which can be seen as a parallel moderated gradecast, which ensures the pairwise graded agreement of honest parties on their output strings. In designing Toss, we mind the randomness and verifiability requirements on the output of $\mathsf{Process}$. Then, we show how to employ the VDFs to obtain the full verifiable graded consensus on random strings protocol.

3.1 Moderated Gradecast and Graded Consensus on Seeds

We note that a single run of a parallel gradecast protocol is sufficient to achieve graded agreement as in Definition 4. Concretely, each party P_j would output for each sender P_i a message and a grade $(m_{i,j}, g_{i,j})$. However, this would not be sufficient to achieve both graded validity with a high grade for honest parties and unpredictability (after running $\mathsf{Process}$). For instance, if P_j would set its final string in Toss for P_i to be $m_{i,j}$, then this would not be random if P_i is malicious. To achieve unpredictability later on, we need intuitively the condition that every output string of Toss contains the input of at least one honest party. However, this is still insufficient, as another failed attempt with Toss implemented as only one parallel gradecast shows: if P_j would set as its random string $\|_{i\in[n], g_{i,j}\geq 1} m_{i,j}$[5], the associated grade of the aggregation could always be determined by malicious parties, potentially yielding low grades for honest parties or high grades for dishonest parties.

Therefore, we propose a second step where parties *moderate* the values they received in the first step. The message of a moderator P_s is thus made up of

[5] Throughout the text, we use the notation $a\|b$ to refer to the concatenation of a and b. More generally, $\|_{i\in[n]} a_i = a_1\|a_2\|\ldots\|a_n$.

messages that it gradecasts itself, but since it gradecasts values from other parties as well, the aggregation is random (if it has high grade). Moreover, moderators have their 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 cannot arbitrarily modify the final grades of honest parties.

In our constructions, we use the GC protocol described in [30] (see Appendix A.3), with maximum grade g^*. We construct a moderated gradecast protocol based on several instances of GC as described above, which we call Mod-GC. In Figure 1, we introduce directly the Toss protocol, which is comprised of a parallel moderated gradecast. We note that in Figure 1, and the rest of this section, $M_i^{(j)}$ replaces the message notations x_i and $x_i^{(j)}$ in Definitions 3 and 4.

Toss(pp)

Step 0: Each party P_i randomly samples a seed m_i.

Step 1: Each party gradecasts their seed.

1. Each party P_i calls $\mathsf{GC}(m_i, g^*)$.
2. Denote the output at P_j of the gradecast with sender P_i as $(m_{i,j}, g_{i,j})$, where $m_{i,j}$ is the message from party P_i and $g_{i,j}$ is the associated grade.

Step 2: Each party acts as moderator.

1. For each party P_s, let $m_i^{(s)} := [m_{i,s}, \mathsf{sig}_i(m_{i,s})]$.
2. Each party P_s calls $\mathsf{GC}(m_i^{(s)}, g^*)$, for all $i \neq s$.
3. For moderator P_s, party P_j holds $(m_{s,j}, g_{s,j}), \{(m_{i,j}^{(s)}, g_{i,j}^{(s)}) : i \in [n] \setminus \{s\}\}$.

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

1. For each party P_j, for each moderator P_s and each $i \in [n]$, P_j sets P_s's value $m_{i,j}^{(s)}$, with grade:

$$G_{i,j}^{(s)} = \begin{cases} g_{i,j}^{(s)} & , \text{ if } m_{i,j}^{(s)} = m_{i,j}, \\ \min\{g_{i,j}^{(s)}, g^* - g_{i,j}\} & , \text{ otherwise.} \end{cases}$$

2. P_j sets its general grade for the moderator P_s as $G_j^{(s)} = \min_{i\in[n]}\{G_{i,j}^{(s)}\}$ and the message $M_j^{(s)} = \|_{i\in[n], G_{i,j}^{(s)} \geq 1} m_{i,j}^{(s)}$.

Fig. 1. Toss protocol (or Parallel Moderated Gradecast on random inputs).

To analyze the properties of Toss, consider first one moderator P_s and one sender P_i running the subprotocol in Figure 1 starting from step 1. We show that this is an instance of a moderated gradecast protocol, Mod-GC(m_i), which satisfies validity, M-validity, soundness and termination against an adaptive adversary controlling $t \leq (1-\epsilon)n$ parties. Then, based on the properties of Mod-GC,

we show that Toss satisfies the graded consensus set of properties specified in Definition 4. The proofs of these results are given in Appendix C.1.

Lemma 1. Mod-GC *is a moderated gradecast protocol as in Definition 3.*

Theorem 1. *Protocol 1 is a Toss protocol with maximum grade g^* satisfying Graded Validity and Graded Agreement and Termination as in Definition 4.*

Communication and Round Complexity. The GC protocol takes $2g^* + 1$ rounds and has $O(g^*(\kappa + \ell)n^2)$ communication complexity. Toss takes $4g^* + 2$ rounds, determined by running n-parallel instances of GC followed by n^2-parallel instances of GC. The total communication complexity for n parties with inputs of length ℓ and signature size κ is $n^2 \cdot \mathsf{CC}_{\mathsf{GC}}(\ell, \kappa, g^*)$, so the total communication complexity is $O(g^* \cdot (\ell + \kappa) \cdot n^4)$.

3.2 Verifiable Graded Consensus for Random Strings

Recall the main challenges: (i) lack of trusted setup, (ii) the obvious lack of broadcast channels, (iii) dishonest majority, and (iv) requirement of sublinear number of rounds. To address point (i), we use Wesolowski's VDF construction [54] based on class group of imaginary quadratic fields, which does not contain trapdoors and does not require trusted setup. Point (iii) is addressed by the Toss construction above which is secure against a dishonest majority.

Points (ii) and (iv) limit the use of the homomorphic properties of the VDF evaluation. Parties cannot gradecast $(m_i, \mathsf{VDF.Eval}(m_i))$ because the rushing adversary can wait to see all such messages, then bias the result by checking the parity of the aggregation of the results. Using timed commitments can help solve this issue. However, in the worst case, the lack of broadcast causes each honest party to open the timed commitment of every malicious party, which is not sublinear time.

We aim for each party to only have to evaluate a single VDF, namely, its own. To this end, we need the graded agreement to happen on the aggregation of the VDF inputs, which in our case, is the concatenation of the parties' seeds, as in the random beacon construction described in [41]. Strings that have grade greater than 1 have the guarantee that they contain seeds of honest parties, therefore the value of this concatenated string will be unpredictable. The hash of these concatenated seeds will be fed into the VDF evaluation, so a rushing adversary that has parallel time smaller than the VDF difficulty parameter cannot bias the input to the VDF. Finally, looking ahead, the VDF evaluation output (w_i, ρ_i) will be used in a setting where it is multicast by the computing party P_i. An honest party P_j will have the graded input $(M_j^{(i)}, G_j^{(i)})$ and will confidently be able to check the validity of the VDF output $\mathsf{VDF.Verify}(M_j^{(i)}, w_i, \rho_j, \cdot)$ if $G_j^{(i)} \geq 1$.

To summarize, we use the VDF construction [54] for the algorithms Process and Verify, and the Toss protocol as a parallel Mod-GC protocol with local seeds sampling, to construct a verifiable graded consensus (VGC) on random strings. To achieve security against the adaptive rushing adversary, the delay required

to evaluate the VDF should be at least twice the time required to run GC (in terms of the synchronous rounds' length). Below, we give more technical details.

VGC steps and parameters. Each party P_i generates a random value $m_i \in \{0,1\}^{q(\kappa)}$, for a polynomial q, such that an adversary can guess m_i only with negligible probability $\mathsf{negl}(\kappa)$. Then, each party shares m_i via the $\mathsf{Mod\text{-}GC}$ protocol. After the end of step 1 of Protocol 1, all honest parties P_j are guaranteed to have obtained their own $M_j^{(j)} = m_j^{(j)} := \|_{i\in[n], g_{i,j}\geq 1} m_{i,j}$, so they can start $\mathsf{Process}$ before the termination of Toss. Note that the adversary can decide on its final local strings $M_j^{(j)}$ for all malicious parties P_j from the first round of the gradecast in step 1, since honest parties will relay their messages from the beginning. After the end of step 1, each party P_j is able to compute $\mathsf{VDF.Eval}_{\mathsf{EK}}(H(m_j^{(j)}))$. Importantly, security holds only when we do not use trapdoor-VDFs, otherwise malicious parties would have an advantage in biasing the output.

We now specify how to obtain the evaluation and verification keys for the VDF. It is sufficient to choose a large enough random discriminant d in order to guarantee that the class group order cannot be computed [54,14,12,11]. In particular, given a random oracle H_d and the bulletin-PKI, parties can compute the discriminant d as the closest negative prime (of size κ) to $\mathsf{coin}_{\mathsf{VDF}} := H_d(\mathsf{PK}_1\|\mathsf{PK}_2\|\ldots\|\mathsf{PK}_n)$ such that $d \bmod 4 \equiv 1$. In short, we specify the random tape used in $\mathsf{VDF.Setup}$ by $\mathsf{coin}_{\mathsf{VDF}}$. Then, each party can choose the group $\mathbb{G} = Cl(d)$ as the class group of the imaginary quadratic field $\mathbb{Q}(\sqrt{d})$. While the adversary can choose its public keys to try to bias the outcome of H_d, it is still unfeasible to compute the order of $\mathbb{G}$ even for a biased d. The only advantage the adversary could have is to obtain the exact discriminant d' for which, somehow, it know the corresponding order, but this is infeasible given only a polynomial number of queries to the random oracle. To complete the VDF key generation, two more random oracles are specified: $H_{\mathbb{G}}$ that maps to $\mathbb{G}$, and H_{prime} that is required for the Fiat-Shamir transform and has to map the inputs which are binary representation of group elements to prime numbers of size dependent of the security parameter (see [54]). One can build deterministic algorithms that take as input $\mathbb{G}$ and output the hash function $H_{\mathbb{G}}$, starting e.g., from SHA (H_{prime} as well). The output of $\mathsf{VDF.Setup}$ is deterministic given the discriminant d, so all parties will have $\mathsf{EK}_i = \mathsf{VK}_i = (\mathbb{G}, H_{\mathbb{G}}, H_{\text{prime}})$. Having the same evaluation and verification keys for all parties is not necessary (but it is also not a problem since no party can compute the group order), so in the following, we prefer to keep a general notation for different EK_i and VK_i to accommodate other potential VDF constructions. We include this deterministic setup for VDF in the $\mathsf{Process}$ algorithm, since honest parties do not require the evaluation and verification keys before obtaining their own evaluation input $M_i^{(i)}$.

We now compute the required difficulty parameter Δ_{VGC}. Let $\Delta_G := (2g^* + 1)\cdot\Delta_r$ denote the duration of the gradecast protocol, where Δ_r is the duration of a round. Toss in Protocol 1 takes double this amount, $2\cdot\Delta_G$. We want to choose the time difficulty parameter such that the adversary cannot finish evaluating the VDF before the end of the Toss protocol. The adversary can do damage even

in the last round of Toss, i.e., change the values of the honest parties' strings for a malicious moderator.

Therefore, we set the puzzle difficulty parameter Δ_{VGC} of the VDF employed in Process to be $\Delta_{\mathsf{VGC}} = 2 \cdot \Delta_G$. Then, the VDF (and Process) will be σ-sequential, for $\sigma(\Delta_{\mathsf{VGC}}) = (1-\xi) \cdot 2 \cdot \Delta_G$, and the adversary takes at least $2 \cdot \Delta_G/(1-\xi)$ total time to evaluate the VDF. (Note that we account for the adversary's speed-up by a very small $\xi > 0$, see Appendix A.2). Finally, let the total time it takes the slowest honest party to evaluate the VDF with difficulty $\Delta_{\mathsf{VGC}} = 2 \cdot \Delta_G$ be $2 \cdot \Delta_G + q_h \cdot \Delta_r \geq 2 \cdot \Delta_G/(1-\xi)$, where $q_h > 0$ is the largest slow-down of an honest party.

We instantiate the algorithms and protocols of the VGC in Figure 2. Gen is the PKI and coin generation required for the VDF setup, Toss is the protocol in Figure 1, Process is the (untrusted) setup for the VDF and the inherited VDF.Eval on the hash of the output, and Verify is the inherited VDF.Verify. Then, we prove that Π_{VGC} satisfies the desired properties (Appendix C.1).

$\Pi_{\mathsf{VGC}}(\Delta_{\mathsf{VGC}}, g^*)$

1. $\mathsf{Gen}(1^\kappa, \Delta_{\mathsf{VGC}})$: Each party P_i posts the keys on the bulletin-PKI, computes $\mathsf{coin}_{\mathsf{VDF}} \leftarrow H_d(\mathsf{PK}_1 \| \mathsf{PK}_2 \| \ldots \| \mathsf{PK}_n)$ and sets $\mathsf{pp} = (\Delta_{\mathsf{VGC}}, \mathrm{PKI}, \mathsf{coin}_{\mathsf{VDF}})$.
2. $\mathsf{Toss}(\mathsf{pp})$: Parties execute the protocol in Figure 1 and each party P_i sets as output $(M_i^{(j)}, G_i^{(j)})$ for each other party P_j and $M_i^{(i)}$ for itself.
3. $\mathsf{Process}(\mathsf{pp}, M_i^{(i)})$: Each party P_i computes $(\mathsf{EK}_i, \mathsf{VK}_i) \leftarrow \mathsf{VDF.Setup}(1^\kappa, \Delta_{\mathsf{VGC}}, \mathsf{coin}_{\mathsf{VDF}})$, where $\mathsf{coin}_{\mathsf{VDF}}$ replaces the internal random tape of VDF.Setup. Each party P_i evaluates its own random string as $(w_i, \rho_i) = \mathsf{VDF.Eval}_{\mathsf{EK}_i}(H(M_i^{(i)}))$ (they can start computing this value before Toss terminates).
4. $\mathsf{Verify}(\mathsf{pp}, x, y, \pi)$: On receiving (w_j, ρ_j) from P_j, party P_i outputs $\mathsf{VDF.Verify}_{\mathsf{VK}_j}(H(M_i^{(j)}), w_j, \rho_j, \Delta_{\mathsf{VGC}})$.

Fig. 2. Verifiable graded consensus protocol for generating random strings.

Theorem 2. *Protocol Π_{VGC} is a verifiable graded consensus protocol on random strings, cf. Definition 4, against an adaptive Δ_{VGC}-limited adversary who runs in at most $(1-\xi) \cdot \Delta_{\mathsf{VGC}} = 2 \cdot \Delta_G$ parallel time, and can adaptively corrupt up to $(1-\epsilon)n$ parties.*

Communication and Round Complexity. Protocol Π_{VGC} takes $4g^* + 2$ rounds to complete Toss, but after $2g^* + 1$ rounds, (after parties obtain their local values $M_i^{(i)}$), they start Process which takes at most $4g^* + 2 + q_h$ rounds. This means that the total round complexity for Π_{VGC} is slightly over $6g^* + 3$, i.e., $6g^* + 3 + q_h$. The total communication complexity is $O(g^* \cdot (\ell + \kappa) \cdot n^4)$.

4 Sublinear-Round Broadcast

We now construct our sublinear broadcast protocol. Our concrete tools are:

1. A Verifiable Graded Consensus on random strings (VGC) protocol satisfying Definition 4—in our case, instantiated via Gradecast and Verifiable Delay Functions (VDFs);
2. An adaptively secure Verifiable Random Function (VRF) that achieves the properties in Definition 8 (even with maliciously generated keys).

The VGC can be seen as an online setup phase that generates graded random identifier strings and proofs of the validity of these identifiers. The VRF will be applied on these identifiers and 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. The VGC outputs also help validate the committee membership against a dishonest majority. To that end, we also need:

3. A bound for the output of the VRFs such that the committees have at least one honest party and fewer malicious parties than half of the number of rounds, with overwhelming probability.

However, we do not exactly obtain the equivalent of a trusted setup via VGC, and the only guarantees that parties have are related to the grades of these distributed 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 as in Chan *et al.* [15], taken to mean consistent committee election and membership verification. Below, we will describe how to use the grades to our advantage over a number of rounds, in order to obtain true agreement between parties.

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 [15] implemented with non-interactive zero-knowledge proofs 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 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. 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.

Specifically, we set the maximum grade g^* that can be returned by the VGC 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 (even honest parties might not receive g^*), 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 $r + 1$.

Recall that we only assume a bulletin-PKI. Every party P_i has generated a signing key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$, a VRF key pair $(\mathsf{SK}_i, \mathsf{PK}_i)$. Every party P_i posts on the public bulletin-board the public keys, so all parties have access to the bulletin-PKI before commencing the broadcast protocol (i.e., before starting the VGC). Honest parties check the predicate $\mathsf{VRF.Validate}_{\mathsf{PK}_j}(1^\kappa)$ and discard the parties P_j for which it does not return 1.

After running Toss from VGC, each party has access to the strings for the rest of the network, with their accompanying grades, and the accompanying proofs for their own output string after running Process. The local string w obtained by a party for itself in VGC should be hashed first to achieve randomness from unpredictability and unbiasedness. As long as the output of Process (in our case, the output of a VDF) has length polynomial in the security parameter κ, the output of a hash function H modeled as a random oracle $H(b||w)$ is random. The outputs of Toss and Process are then enough for the parties to run $g^*/2$ stages (with g^* rounds) as described above, where in each stage parties vote and echo the votes of the valid members of the committee, via the VRF.

Now we finalize the technical points for the trustless sublinear-round protocol.

Electing a committee member. Each party P_i maintains two variables, named $call_i^0$ and $call_i^1$, both initialized by -1. The role of the variable $call_i^b$ is to store the local view of whether P_i is in the committee for bit $b \in \{0,1\}$.

Given a bit b, SK_i the VRF secret key of P_i, w_i the $\mathsf{Process}$ output of P_i on $M_i^{(i)}$, and $\mathsf{bound}_{\epsilon,\delta}$ the appropriate bound for the VRF, the party P_i checks (once) whether it is in the committee for b:

(i) Retrieve from memory $(y_i, \pi_i) \leftarrow \mathsf{VRF.Prove}_{\mathsf{SK}_i}(H(b||w_i))$.
(ii) If $call_i^b = -1$ and $y_i < \mathsf{bound}_{\epsilon,\delta}$, set $call_i^b = 1$, else set $call_i^b = \perp$.

Following the notation in Chan *et al.*, we use p_{mine} to refer to the probability of a party self-electing in a committee. We set this mining probability to take the value $p_{\text{mine}} = \min\{1, \frac{1}{\epsilon n}\log\left(\frac{2}{\delta}\right)\}$, where ϵ is a constant in $(0,1)$ denoting the fraction of honest parties and δ is a failure probability, which is constant and negligible in the statistical security parameter λ: $\delta = \exp(-\omega(\ln\lambda))$. The bound for the VRF output check for committee election is $\mathsf{bound}_{\epsilon,\delta} = p_{\text{mine}} \cdot 2^{m(\kappa)}$, where the VRF output length is $m(\kappa)$. Finally, we set the maximum grade and the number of rounds g^* to be $g^* = 2 \cdot \lceil \frac{1}{\epsilon} \ln\left(\frac{2}{\delta}\right) \rceil = O(\lambda/\epsilon)$. 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 (Lemmata 14 and 15 given in Appendix C.2).

Verifying membership, valid batches and certificates. The 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. We prefer this notation because we want to have sets of r signatures in every stage r.

Each party collects *batches* of signatures. 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. To address the difference in grades in the two rounds, we define $(r,1)$-batches and $(r,2)$-batches. Moreover, each party P_i will maintain a set $\mathsf{Extracted}_i$, initialized to the empty set. A party P_i adds a bit b to their $\mathsf{Extracted}_i$ set in a round if it receives a valid batch and a valid certificate on b for that round, as described below.

A batch consists of a number of signatures associated to distinct parties. A certificate consists of a number of tuples (w, ρ, y, π). Recall that a party P_i has obtained $M_i^{(j)}$ for a party P_j, which it can use to validate the evaluation output w_j claimed by P_j. We define a valid certificate only with respect to a certain batch. Namely, for a batch_b consisting of a number of signatures $(\mathsf{sig}_j(b))$, we say certificate cert_b is associated to batch_b if it consists of tuples $(w_j, \rho_j, y_j, \pi_j)$ for every j-signature in batch_b. A certificate cert_b is a *valid certificate* for a batch batch_b from the perspective of a verifying party P_i, if for all j-signatures in batch_b not coming from the sender, it holds that:

(i) $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||w_j), y_j, \pi_j) = 1$,
(ii) $\mathsf{VGC.Verify}(\mathsf{pp}, M_i^{(j)}, w_j, \rho_j) = 1$, where $(M_i^{(j)}, \cdot) \leftarrow \mathsf{Toss}(\mathsf{pp})$, and
(iii) $y_j \leq \mathsf{bound}_{\epsilon,\delta}$.

We say that a batch batch_b, consisting of tuples $(\mathsf{sig}_j(b))$ 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 signature $\mathsf{sig}_j(b)$, the grade $G_i^{(j)} \geq g^* - 2r + 1$, where $(\cdot, G_i^{(j)}) \leftarrow \mathsf{Toss}(\mathsf{pp})$ for P_i, and
(iii) It has associated a valid certificate cert_b (P_i receives both batch_b and cert_b).

Similarly, a batch is a *valid* $(r,2)$*-batch* if it contains at least r distinct signatures coming from parties in the b-committee, each with grade $G_i^{(j)} \geq g^* - 2r$, and with a valid certificate.

If at any point, a valid batch has accumulated more than $g^*/2$ signatures, parties only need to send $g^*/2$ of them. We assume implicitly that parties send at most $g^*/2$ signatures in a batch batch_b, and always include the sender's signature.

We call a pair of one batch and one certificate a ballot, and say that the ballot is valid if the batch is valid and the certificate is the valid certificate linked to that batch. For clarity, we preferred to define separately the validity of the batches and of the certificates, rather than lumped in a single definition of a valid ballot.

We describe the full protocol in Figure 3. We first run Π_{VGC}. Parties need the random strings from Π_{VGC} starting from round 1, since they are only used in proofs, not in the sender's initial transmission. Recall that Π_{VGC} takes $6g^*+3+q_h$ rounds, where q_h captures the delay of the slowest honest parties. The proof of Theorem 3 is given in Appendix C.2.

Theorem 3. *Consider a Δ_{VGC}-limited adversary who can adaptively corrupt $(1-\epsilon)n$ parties, for a constant $\epsilon \in (0,1)$. Fix a small constant failure probability $\delta \in (0,1)$. Then, Protocol Π_{BC} (Figure 3) is a broadcast protocol with probability $1 - \delta - \mathsf{negl}(\kappa)$.*

Communication and Round Complexity. The broadcast protocol has round complexity $O(g^* + \mathsf{R}_{\mathsf{VGC}}) = O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)\right)$ and communication complexity $O(g^* \cdot \kappa \cdot n^2 + \mathsf{CC}_{\mathsf{VGC}}) = O(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right) \cdot \kappa \cdot n^4)$. For $\delta = \exp(-\omega(\log\lambda))$ negligible in the security parameter, we obtain a round complexity of $O(\lambda/\epsilon)$ and a total communication complexity of $O(\kappa \cdot \lambda \cdot n^4/\epsilon)$.

5 Communication Reduction for Parallel Gradecast

The communication complexity of sharing and agreeing on random strings is the dominating term in our Π_{BC} communication. To improve upon that, in this section we leverage parallelization and randomization. We take inspiration from the recent work [51] and use gossiping to lower the communication cost of performing parallel gradecast by a factor of $O(n/\mathsf{polylog}(n))$, improving the communication of the verifiable graded consensus protocol (and thus of broadcast) to $\tilde{O}(n^3)$.

Tsimos *et al.* [51] formulates the notion of honest parties disseminating messages via gossiping in a communication-efficient way that is adaptively secure.

Π_{BC}: Sublinear-round, trustless Broadcast

Rounds $-6g^* - 3 - q_h$ to 0:

1. Parties run Π_{VGC}. Each party P_i obtains the following quantities $(\mathsf{pp}, \{M_i^{(j)}, G_i^{(j)}\}_{j\in[n]}, w_i, \rho_i)$.

Stage 0:

1. (Round 0) Each party P_i initializes $\mathsf{Extracted}_i = \emptyset$ and $call_i^0 = call_i^1 = -1$. The designated sender P_s sends $[b_s, \mathsf{sig}_s(b_s), \perp]$ to all parties.

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 propagates $(b, \mathsf{batch}_b, \mathsf{cert}_b)$ 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 such b, P_i checks whether $call_i^b = -1$ and if yes, it computes $(y_i, \pi_i) \leftarrow \mathsf{VRF.Prove}_{\mathsf{SK}_i}(H(b||w_i))$. If $y_i \leq \text{bound}_{\epsilon,\delta}$, P_i does:
   - sets $call_i^b = 1$;
   - sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$;
   - constructs a $(r + 1, 1)$-batch $\mathsf{batch}'_b := \mathsf{batch}_b \| \mathsf{sig}_i(b)$, $\mathsf{cert}'_b := \mathsf{cert}_b \| (w_i, \rho_i, y_i, \pi_i)$.

   Else, P_i sets $call_i^b = \perp$.
   P_i 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.

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

In their model, input values are single bits and are defined per pairs of sender and signer, meaning that the total number of valid messages is less than $2n^2$. This formulation does not apply well to our moderated gradecast step, which works on messages of size $q(\kappa)$. In our case, in the moderated gradecast, the adversary can provide as many valid different messages as it wants (up to $q(\kappa)$) for pairs of dishonest sender and dishonest moderator, so dishonest moderators can send a large number of messages from the same dishonest sender to the honest parties at the beginning of the protocol. Calling the main dissemination protocol from Tsimos *et al.* on that many messages could lead to $\tilde{O}(n^3 \cdot 2^{q(\kappa)})$ communication. For our protocol, in order to keep the communication low, we

require honest parties to propagate a constant number of messages per pair of sender and moderator, while maintaining the required properties of gradecast.

To this end, we modify a different formulation from [51] to meet our needs, which uses a function that takes a set and outputs only one message per each k-bit prefix. We define a set of valid input values to be the set of all possible messages $m_{j,s}$ for each pair of sender P_j and moderator P_s. However, in this set, we consider any two pairs of two messages for the same (j, s) as the same. We want honest parties to propagate at most two valid messages per each prefix that defines a pair of sender and moderator, as well as for each prefix that defines a separate sender during the initial gradecast, while maintaining the required properties of gradecast. We note that if a dishonest sender attempts to send multiple valid values, propagating at least two of those values is sufficient to guarantee that honest parties will output grades in $\{0, 1\}$ for that sender. We call this variation of their protocol $\mathcal{M}$-Converge* and formalize it below.

Protocol for $\mathcal{M}$-Converge*. We first describe a function which, given a set S of bit-strings and an integer value k, outputs a subset of S. This subset contains for each k-bit prefix, either exactly all strings of S with that prefix if they are at most 2, or any of the strings in S otherwise. Looking ahead, this function is combined with gossiping during the communication rounds of Parallel Moderated Gradecast. It ensures that for each k prefix (defining a specific sender) honest parties propagate at most two values. This accounts for dishonest senders attempting to flood the communication of gradecast with any number of values.

Definition 6 (couples$_k$ function). *For any set M, couples$_k(M)$ is a subset of M that contains for each distinct k-bit prefix at most two messages with that prefix, i.e., if there are fewer than two message with k-bit prefix PR, then couples$_k(M)$ contains exactly those messages, and if there are more than two messages with prefix PR, then couples$_k(M)$ contains only two of them.*

For example, for $M = \{00101, 01000, 01100, 11001, 11010, 11111\}$ we have couples$_2(M) = \{00101, 01000, 01100, 11001, 11111\}$. Since couples$_k$ is an one-to-many function, couples$_2(M)$ might also output $\{00101, 01000, 01100, 11010, 11111\}$.

We now present $\mathcal{M}$-Converge*, which captures how parties can propagate their values, so that they are all guaranteed to receive at least all the intended values from honest senders. We show an instantiation with more efficient communication than parties multicasting their values.

Definition 7 ($\mathcal{M}$-Converge* protocol). *Let $\mathcal{M} \subseteq \{0,1\}^*$ be an efficiently recognizable set (i.e. a set with efficient membership decidability). A protocol Π executed by n parties, where every honest party P_i initially holds input set $M_i \subseteq \mathcal{M}$ and a set $\mathcal{C}_i \subseteq \mathcal{M}$, is a* secure $\mathcal{M}$-Converge* protocol *if all remaining honest parties upon termination, output a set $S_i \supseteq$ couples$_k\left(\bigcup_{j\in\mathcal{H}} M_j - \bigcup_{j\in\mathcal{H}} \mathcal{C}_j\right)$, when at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of Π.*

Let $p_{prop} = (10/\epsilon + \lambda)/n$. We consider the ideal functionality $\mathcal{F}_{\mathsf{prop}}$ from [51], which allows for each party P_i to send a set of messages to an average number of $n \cdot p_{prop}$ randomly chosen parties out of a set of n parties, while achieving the property that the adversary does not gain information on which honest parties received the message. This functionality is the conceptual building block behind gossiping against an adaptive adversary; it is invoked by our Π_{CV} protocol in every of its rounds by all honest parties with input $(\mathsf{SendRandom}, M_i)$. The adversary can also call it with input $(\mathsf{SendDirect}, \mathbf{x}, J)$, to send messages in $\mathbf{x}$ to parties P_j, for $j \in J$. A formal description and a secure instantiation of $\mathcal{F}_{\mathsf{prop}}$ is provided for completeness in Appendix A.4.

We present protocol Π_{CV} in the $\mathcal{F}_{\mathsf{prop}}$-hybrid world in Figure 4 and we state its properties in Lemma 2. The proof is given in Appendix C.3.

Lemma 2. *Let $\kappa > 0$. Protocol Π_{CV} is an adaptively secure $\mathcal{M}$-$\mathsf{Converge}^*$ protocol for all $t \leq (1-\epsilon) \cdot n$ and fixed $\epsilon \in (0,1)$, with probability $1 - \mathsf{negl}(\lambda)$. The total number of bits sent by all parties is $O(n \log(\epsilon n) \cdot \max\{n, |\mathsf{couples}_k(\mathcal{M})|\} \cdot m \cdot s)$.*

Gradecast via $\mathcal{M}$-$\mathsf{Converge}^*$. We propose a new protocol for parallel gradecast with one instance per pair of ($j \in$ senders, $k \in$ casts). Each party P_j, $j \in$ senders, is expected to gradecast |casts| many messages. The trivial (and less efficient) way to do so would be for each sender P_j to call $\mathsf{GC}(m_k^j)$, for $k \in$ casts. Instead, by calling Π_{CV}, all senders can gradecast simultaneously all their |casts| many messages, with less communication, at the small additional cost of multiplicative $O(\log \epsilon n)$ rounds.

$\Pi_{\mathsf{CV}}(M_i, \mathcal{C}_i, k)$

For round 1 to $\lceil \log(\epsilon \cdot n) \rceil$:
- $\textsc{Receive}^{\mathcal{F}_{\mathsf{prop}}} \leftarrow \mathcal{F}_{\mathsf{prop}}(\mathsf{SendRandom}, \mathsf{couples}_k(M_i - \mathcal{C}_i))$;
- $\mathsf{Local}_i \leftarrow \mathsf{Local}_i \cup \textsc{Receive}^{\mathcal{F}_{\mathsf{prop}}}$;
- $\mathcal{C}_i = \mathcal{C}_i \cup M_i$;
- $M_i = \mathsf{Local}_i \cap \mathcal{M}$;

return M_i;

Fig. 4. Protocol Π_{CV} is a secure $\mathcal{M}$-$\mathsf{Converge}^*$ protocol. It uses a logarithmic number of rounds, each of which utilizes gossiping (via the call to $\mathcal{F}_{\mathsf{prop}}$) to securely and efficiently disseminate a list of messages between parties.

Parallel Vector GC in Figure 5 is a secure gradecast protocol, for each separate value each sender sends. We prove the next Lemma in Appendix C.3.

Lemma 3. *Protocol Parallel Vector $\mathsf{GC}_i^{(\cdot,\cdot)}(\cdot, g^*)$ is a g^*-gradecast protocol with probability $1 - \mathsf{negl}(\lambda)$, with round complexity $2g^* \cdot \lceil \log(\epsilon n) \rceil + 1$. The total communication complexity for all parties is $O(n \log \epsilon n \cdot g^* \cdot \max\{n, |\mathit{senders}| \cdot |\mathit{casts}|\} \cdot m \cdot s)$.*

Parallel Vector $\mathsf{GC}_i^{(\text{senders},\text{casts})}(M_i, g^*)$

Round 1:

1. Each party P_i initializes $S^i_{(j,k)} := \emptyset$, $m^i_{(j,k)} = \perp$, $\bar{g}^i_{(j,k)} = 0$, for all $j \in \text{senders}, k \in \text{casts}$.
2. If $i \in \text{senders}$, then P_i computes a signature $\sigma_k = \mathsf{sig}_i(m_k \in M_i)$, for each $k \in \text{casts}$ and multicasts (m_k, σ_k) to all parties.

Round 2 to $2g^* + 1$ from the perspective of party P_i:

1. Let $\mathbf{M}$ contain $m'_{(j,k)}$ from each pair $(m'_{(j,k)}, \sigma'_{(j,k)})$ received by the end of the previous round, where $\mathsf{ver}_s(\sigma'_{(j,k)}, m'_{(j,k)}) = 1$ and $m'_{(j,k)} \notin S^i_{(j,k)}$. Then for $m'_{(j,k)} \in \mathbf{M}$ set $S^i_{(j,k)} := S_i \cup \{m'_{(j,k)}\}$. If $|S^i_{(j,k)}| = 1$, set $m^i_{(j,k)} = m'_{(j,k)}$.
2. Call and receive messages from $\Pi_{\mathsf{CV}}(\mathbf{M}, \emptyset, k_{sc})$, where k_{sc} is the bit-length needed to express the prefix for every pair of $(j \in \text{senders}, k \in \text{casts})$.
3. If $m^i_{(j,k)} \neq \perp$ and $|S^i_{(j,k)}| = 1$, then set $\bar{g}^i_{(j,k)} := \bar{g}^i_{(j,k)} + 1$.

Output determination: Each party P_i sets $g^i_{(j,k)} := \lfloor \bar{g}^i_{(j,k)}/2 \rfloor$ and outputs $(m^i_{(j,k)}, g^i_{(j,k)})$.

Fig. 5. Gradecast protocol with maximum grade g^* using gossiping.

6 Communication-Efficient Broadcast

Finally, we discuss how to reduce the communication complexity of our trustless sublinear-round broadcast protocol.

In a formal analysis, the requirements inherited by the previous works of [15] and [51] respectively, mean that we may need two distinct statistical security parameters. Therefore, in this section, we consider two statistical security parameters λ_δ and λ_g. Concretely, λ_δ is the statistical security parameter used for the committee-election (in Section 4), and λ_g is the statistical security parameter used for the gossiping analysis (in Section 5). Chan *et al.* [15] requires the statistical security parameter λ_δ to be independent of n, so that the failure probability of secure election δ—which relates to λ_δ as $\log(1/\delta) = \mathsf{polylog}(\lambda_\delta)$—is independent of n as well. Tsimos *et al.* [51] requires the statistical security parameter λ_g to be $\mathsf{polylog}(n)$, so that the failure probability of gossiping is negligible in λ_g. (However, λ_g does not affect round complexity of our broadcast protocol, only its communication complexity.) Below, we specify each separate protocol with their respective statistical security parameters. Nevertheless, for simplicity, we could also set and use a single statistical security parameter as $\lambda = \max\{\lambda_\delta, \lambda_\mathsf{g}\}$.

Consider our Protocol 1 for Toss. During its step 2.2, instead of GC, let each party P_i call $\mathsf{Parallel\ Vector\ GC}_i^{([n],n-1)}(\{m_i^{(s)}\}_{s\in[n]-\{i\}}, g^*)$, given in Figure 5, to moderate the values it received previously. Let ℓ be the individual message length ($\ell = |m_i^{(s)}|$). This allows parties to moderate the n random strings they each received during step 1 with $\tilde{O}(g^*\cdot(\ell+\kappa)\lambda_\mathsf{g}\cdot n^3)$ total communication, while adding a multiplicative factor of $\lceil \log \epsilon n \rceil$ to the round complexity of the moderated step. Similarly, the updated Π'_{VGC} protocol that now calls the updated Toss has the

same communication and round complexity as the updated Toss. The updated moderated gradecast step has duration $\Delta'_G := (2g^* \cdot \lceil \log(\epsilon n) \rceil + 1) \cdot \Delta_r$. We update the difficulty parameter to account for the increased number of rounds: $\Delta'_{\mathsf{VGC}} = (\Delta_G + \Delta'_G)/(1 - \xi)$.

Putting it all together, the broadcast protocol with the updated VGC protocol using gossiping described in Section 5, Π'_{VGC}, achieves the following result. The proof is given in Appendix C.3.

Theorem 4. *Consider a Δ'_{VGC}-limited adversary who can adaptively corrupt $(1-\epsilon)n$ parties, for a constant $\epsilon \in (0,1)$. Fix a small constant failure probability $\delta \in (0,1)$. Then, Protocol 3 using Π'_{VGC} is a broadcast protocol, called Π'_{BC}, with probability $1 - \delta - \mathsf{negl}(\kappa) - \mathsf{negl}(\lambda_{\mathsf{g}})$.*

Communication and Round Complexity. The broadcast protocol Π'_{BC} that calls Π'_{VGC}, has round complexity $O(g^* + \mathsf{R}_{\mathsf{VGC}'}) = O\left(\frac{1}{\epsilon} \log\left(\frac{1}{\delta}\right) \lceil \log(\epsilon n) \rceil\right)$ and communication complexity $O(g^* \cdot \kappa \cdot n^2 + \mathsf{CC}_{\mathsf{VGC}'}) = O(\frac{1}{\epsilon} \log\left(\frac{1}{\delta}\right) \cdot \lambda_\delta \cdot \kappa \cdot n^3)$. For $\delta = \exp(-\omega(\log \lambda_\delta))$, we obtain a round complexity of $\tilde{O}(\lambda_\delta/\epsilon)$ and a total communication complexity of $\tilde{O}(\lambda_\delta \cdot \kappa \cdot n^3/\epsilon)$.

Notice that Protocol 3 uses the online setup of VGC for a single broadcast. However, we can bootstrap the randomness created by Π_{VGC} to obtain VRF seeds that are still unpredictable and verifiable for multiple broadcast instances. This allows us to amortize the communication cost over multiple instances (either sequential or parallel). The proof of the following result is given in Appendix C.4.

Theorem 5. *We obtain broadcast protocols secure against adaptive dishonest majority of $t \leq (1-\epsilon)n$ with overwhelming probability in the security parameters $\kappa, \lambda_{\mathsf{g}}, \lambda_\delta$ with the amortized cost of:*

1. *$\tilde{O}(\lambda_\delta)$ rounds and $\tilde{O}(n^2)$ communication over n instances, with prob. $1 - \delta - \mathsf{negl}(\lambda_{\mathsf{g}}) - \mathsf{negl}(\kappa)$;*
2. *$O(\lambda_\delta)$ rounds and $\tilde{O}(n^2)$ communication over n^2 instances, with prob. $1 - \delta - \mathsf{negl}(\kappa)$;*
3. *$\tilde{O}(\lambda_\delta)$ rounds and $\tilde{O}(n)$ communication over n^2 instances, with prob. $1 - \delta - \mathsf{negl}(\lambda_{\mathsf{g}}) - \mathsf{negl}(\kappa)$.*

References

1. I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. Communication complexity of byzantine agreement, revisited. In P. Robinson and F. Ellen, editors, *38th ACM PODC*, pages 317–326. ACM, July / Aug. 2019.
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, A. Kate, K. Nayak, and N. Shrestha. OptRand: Optimistically responsive distributed random beacons. Cryptology ePrint Archive, Report 2022/193, 2022. `https://eprint.iacr.org/2022/193`.
5. 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.
6. E. Blum, E. Boyle, R. Cohen, and C.-D. Liu-Zhang. Communication lower bounds for cryptographic broadcast protocols. In *37th International Symposium on Distributed Computing (DISC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.
7. 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.
8. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001.
9. D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, Aug. 2000.
10. 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.
11. L. Braun, I. Damgård, and C. Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *Annual International Cryptology Conference*, pages 613–645. Springer, 2023.
12. J. Buchmann and S. Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2011.
13. V. Buterin. Ethereum white paper. White paper, Ethereum, 2013.
14. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 266–296. Springer, Heidelberg, May 2020.
15. 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.
16. J. Chen and S. Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.
17. B. Cohen and K. Pietrzak. The chia network blockchain. Technical report, Chia Network, 2019.

18. R. Cohen, J. Garay, and V. Zikas. Completeness theorems for adaptively secure broadcast. In *CRYPTO 2023*, 2023.
19. 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.
20. P. Das, L. Eckey, S. Faust, J. Loss, and M. Maitra. Round efficient byzantine agreement from VDFs. Cryptology ePrint Archive, Report 2022/823, 2022. `https://eprint.iacr.org/2022/823`.
21. L. De Feo, S. Masson, C. Petit, and A. Sanso. Verifiable delay functions from supersingular isogenies and pairings. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 248–277. Springer, Heidelberg, Dec. 2019.
22. D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. In R. L. Probert, M. J. Fischer, and N. Santoro, editors, *1st ACM PODC*, pages 132–140. ACM, Aug. 1982.
23. D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
24. P. Feldman and S. Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.
25. P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.
26. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 1985.
27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. 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.
33. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. `https://eprint.iacr.org/2017/454`.
34. S. Goldberg, J. Vcelak, D. Papadopoulos, and L. Reyzin. Verifiable random functions (VRFs), 2018.
35. C. Hoffmann, P. Hubáček, C. Kamath, and T. Krňák. (verifiable) delay functions from lucas sequences. *Cryptology ePrint Archive*, 2023.

36. 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.
37. 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.
38. 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.
39. V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *17th SODA*, pages 990–999. ACM-SIAM, Jan. 2006.
40. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
41. A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.
42. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.
43. A. Momose and L. Ren. Optimal communication complexity of authenticated byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
44. A. Momose, L. Ren, E. Shi, J. Wan, and Z. Xiang. On the amortized communication complexity of byzantine broadcast. Cryptology ePrint Archive, Paper 2023/038, 2023. `https://eprint.iacr.org/2023/038`.
45. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.
46. K. Pietrzak. Simple verifiable delay functions. In A. Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, Jan. 2019.
47. M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27:256–267, 1983.
48. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.
49. 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.
50. 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.
51. G. Tsimos, J. Loss, and C. Papamanthou. Gossiping for communication-efficient broadcast. *CRYPTO 2022*, 2022.
52. 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.
53. 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.
54. B. Wesolowski. Efficient verifiable delay functions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

Table of Contents

A More Preliminaries and Primitives

A.1 Verifiable Random Functions

Definition 8. *A verifiable random function (VRF)* $F = (\mathsf{VRF.Gen}, \mathsf{VRF.Prove}, \mathsf{VRF.Verify}, \mathsf{VRF.Validate})$ *is a tuple of algorithms, where* $\mathsf{VRF.Gen}$ *is a ppt algorithm, and* $\mathsf{VRF.Prove}, \mathsf{VRF.Verify}, \mathsf{VRF.Validate}$ *are deterministic algorithms:*

- $\mathsf{VRF.Gen}(1^\kappa)$ *outputs a pair of keys* $(\mathsf{PK}, \mathsf{SK})$*;*
- $\mathsf{VRF.Prove}_{\mathsf{SK}}(x)$ *outputs a pair* $(y, \pi_{\mathsf{SK}}(x))$*, where* $y \in \{0,1\}^{m(\kappa)}$ *is the evaluation on input* x *using secret key* SK *and* $\pi_{\mathsf{SK}}(x)$ *is the proof of correctness of this evaluation;*
- $\mathsf{VRF.Verify}_{\mathsf{PK}}(x, y, \pi)$ *outputs 1 if* y *is the correct output of input* x *associated to key* SK *using proof* π *and 0 otherwise;*
- $\mathsf{VRF.Validate}_{\mathsf{PK}}(1^\kappa)$ *outputs 1 if* PK *corresponds to an admissible* SK *with respect to* $\mathsf{VRF.Gen}$ *and 0 otherwise.*

A VRF should satisfy the following properties:

(Uniqueness) No values $(\mathsf{PK}, x, y_1, y_2, \pi_1, \pi_2)$ *can satisfy both predicates* $\mathsf{VRF.Verify}_{\mathsf{PK}}(x, y_1, \pi_1) = 1$ *and* $\mathsf{VRF.Verify}_{\mathsf{PK}}(x, y_2, \pi_2) = 1$ *if* $y_1 \neq y_2$ *with more than negligible probability.*
(Provability) If $(y, \pi) \leftarrow \mathsf{VRF.Prove}_{\mathsf{SK}}(x)$ *and* $\mathsf{VRF.Validate}_{\mathsf{PK}}(1^\kappa) = 1$*, then* $\mathsf{VRF.Verify}_{\mathsf{PK}}(x, y, \pi) = 1$.
(Pseudorandomness) For any probabilistic polynomial time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *who has not yet called the oracle on* x*, it holds that:*

$$\Pr\left[\mathcal{A}_2^{\mathsf{VRF.Prove}(\cdot)}(y_b, st) = b \ : \ \begin{array}{c}(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{VRF.Gen}(1^\kappa) \\ (x, st) \leftarrow \mathcal{A}_1^{\mathsf{VRF.Prove}(\cdot)}(\mathsf{PK}, l(\kappa)) \\ y_0 \leftarrow \mathsf{VRF.Prove}_{\mathsf{SK}}(x),\ y_1 \leftarrow \{0,1\}^{m(\kappa)} \\ b \leftarrow \{0,1\}\end{array}\right] \leq \mathsf{negl}(\kappa) + \frac{1}{2}.$$

(Extended pseudorandomness) For any probabilistic polynomial time algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *and* $\mathcal{B}$ *who has not yet called the oracle on* $x||u$*, it holds that:*

$$\Pr\left[\mathcal{B}^{\mathsf{SK},x}(y_b, st_0, st_1) = b : \begin{array}{c}(\mathsf{PK}, \mathsf{SK}, st_0) \leftarrow \mathcal{A}_0(1^\kappa) \\ (x, st_1) \leftarrow \mathcal{A}_1^{\mathsf{VRF.Prove}(\cdot)}(\mathsf{PK}, l(\kappa) - n(\kappa), st_0) \\ \textit{If } \begin{cases}\mathsf{VRF.Validate}_{\mathsf{PK}}(1^\kappa) = 1 \\ x \in \{0,1\}^{l(\kappa)-n(\kappa)}\end{cases} \textit{, then } \begin{array}{l}u \xleftarrow{\$} \{0,1\}^{n(\kappa)} \\ z \xleftarrow{\$} \{0,1\}^{l(\kappa)}\end{array} \\ (y_0, \pi_0, y_1, \pi_1) \leftarrow \mathcal{A}_2^{\mathsf{SK}}(x||u, z, st_1) \\ \textit{If } \begin{cases}\mathsf{VRF.Verify}_{\mathsf{PK}}(x||u, y_0, \pi_0) = 1 \\ \mathsf{VRF.Verify}_{\mathsf{PK}}(z, y_1, \pi_1) = 1\end{cases} \textit{, then } b \xleftarrow{\$} \{0,1\}\end{array}\right] \leq \mathsf{negl}(\kappa) + \frac{1}{2}.$$

The input x to be fed to $\mathsf{VRF.Prove}$ has length $l(\kappa)$ and the output length of y is $m(\kappa)$; in the case of extended pseudorandomness, the adversarial input has length $l(\kappa) - n(\kappa)$ and the challenger generated input has length $n(\kappa)$, where l, m, n are polynomials in κ.

This extended pseudorandomness property is different from the pseudorandomness property where the adversary only chooses the input but not the key pair and thus can't evaluate the VRF. The extended pseudorandomness game given above works as follows. The adversary generates the key pair and an input, and a challenger samples a random string that should be concatenated with the input and a random input. The adversary has to generate VRF outputs and proofs for the concatenated input and random string and for the random input. A distinguisher is given the secret key, the input chosen by the adversary (but not the random strings of the challenger) and the two outputs, and should not be able to distinguish between the last two.

Note that we formulated the definition of extended pseudorandomness to be more general and to account for a partial input x chosen by the adversary. However, in our protocols, x will be the empty string.

A.2 Verifiable Delay Functions

Definition 9. *A Verifiable Delay Function (VDF)* $V = (\mathsf{VDF.Setup}, \mathsf{VDF.Eval}, \mathsf{VDF.Verify})$ *is a triplet of algorithms:*

- $\mathsf{VDF.Setup}(1^\kappa, \Delta) \to \mathit{pp} = (\mathsf{EK}, \mathsf{VK})$ *takes the security parameter* κ *and target puzzle difficulty* Δ *that outputs public parameters pp that consist of an evaluation key* EK *and verification key* VK*;*
- $\mathsf{VDF.Eval}_{\mathsf{EK}}(x) \to (w, \rho)$ *outputs* $w \in \mathcal{Y}$*, the evaluation on input* $x \in \{0,1\}^{l(\kappa)}$ *with evaluation key* EK *and* ρ*, the proof of correctness of this evaluation;*
- $\mathsf{VDF.Verify}_{\mathsf{VK}}(x, w, \rho, \Delta)$ *outputs 1 if* w *is the correct output of input* x *associated to verification key* VK *and difficulty* Δ*, possibly using proof* ρ*, and 0 otherwise.*

For all pp *generated by* $\mathsf{VDF.Setup}(1^\kappa, \Delta)$ *and all* $x \in \{0,1\}^{l(\kappa)}$*, algorithm* $\mathsf{VDF.Eval}_{\mathsf{EK}}(x)$ *must run in parallel time* Δ *with* $\mathsf{poly}(\log \Delta, \kappa)$ *processors and evaluates a deterministic function of* x*. Algorithm* VDF.Verify *is deterministic and must run in total time polynomial in* $\log \Delta$ *and* κ*.*

A VDF should satisfy the following properties:
(Correctness) A VDF V *is* correct *if* $\forall \kappa, \Delta, \mathit{pp} \leftarrow \mathsf{VDF.Setup}(1^\kappa, \Delta)$ *and* $\forall x \in \{0,1\}^{l(\kappa)}$*, if* $(w, \rho) \leftarrow \mathsf{VDF.Eval}_{\mathsf{EK}}(x)$ *then* $\mathsf{VDF.Verify}_{\mathsf{VK}}(x, w, \rho, \Delta) = 1$*.*
(Soundness) A VDF V *is* sound *if for all algorithms* $\mathcal{A}$ *that run in time* $O(poly(\Delta, \kappa))$*, it holds:*

$$\Pr\left[\begin{matrix}\mathsf{VDF.Verify}_{\mathsf{VK}}(x, w, \rho, \Delta) = 1 \\ w \neq \mathsf{VDF.Eval}_{\mathsf{EK}}(x)\end{matrix} : \begin{matrix}\mathit{pp} = (\mathsf{EK}, \mathsf{VK}) \leftarrow \mathsf{VDF.Setup}(1^\kappa, \Delta) \\ (x, w, \rho) \leftarrow \mathcal{A}(\kappa, \mathit{pp}, \Delta)\end{matrix}\right] \leq \mathit{negl}(\kappa).$$

((p,σ)-Sequentiality) For functions $\sigma(\Delta)$ *and* $p(\Delta)$*, a VDF* V *is* (p,σ)*-sequential if no pair of randomized algorithms* $\mathcal{A}_0$*, which runs in total time* $O(\mathsf{poly}(\Delta, \kappa))$*, and* $\mathcal{A}_1$*, which runs in parallel time* $\sigma(\Delta)$ *on at most* $p(\Delta)$ *processors, can win the sequentiality game (Definition 10) with probability greater than* $\mathit{negl}(\kappa)$*.*

Definition 10. *[Sequentiality Game] An adversary* $\mathcal{A} := (\mathcal{A}_0, \mathcal{A}_1)$ *and a challenger* $\mathcal{C}$ *play a sequentiality game with security parameter* κ *and time* Δ*:*

1. $\mathcal{C}$ *runs setup and obtains* $\mathit{pp} \leftarrow \mathsf{VDF.Setup}(\kappa, \Delta)$*.*
2. $\mathcal{A}$ *processes the public parameters and obtains* $L \leftarrow \mathcal{A}_0(\kappa, \mathit{pp}, \Delta)$*.*
3. $\mathcal{C}$ *samples a uniform random string* $x_1 \xleftarrow{\$} \{0,1\}^{l(\kappa)-n(\kappa)}$*.*
4. $\mathcal{A}$ *chooses a string* $x_2 \in \{0,1\}^{n(\kappa)}$*, processes* x_1 *and computes an output* $w_A \leftarrow \mathcal{A}_1(L, \mathit{pp}, x_1||x_2)$*.*

The adversary $\mathcal{A}$ *wins the game if* $w_A = w$ *for* $(w, \rho) \leftarrow \mathsf{VDF.Eval}_{\mathsf{EK}}(x_1||x_2)$*.*

The underlying group of the VDF in [54] is a class group $\mathbb{G}$ of an imaginary quadratic field. For a specific parametrization, there is no efficient algorithm to compute the order of the group, so this VDF construction does not have trapdoors. The construction uses two hash functions modeled as random oracles. We detail the setup algorithm, for later reference, while for the rest of the algorithms we refer to [54].

$\mathsf{VDF.Setup}(1^\kappa, \Delta) \to \mathsf{pp} = (\mathsf{EK}, \mathsf{VK})$: Sample a random value of κ bits coin. Find the closest negative prime number d to coin that satisfies $d \bmod 4 \equiv 1$. Set the group $\mathbb{G} = Cl(d)$ as the class group of the imaginary quadratic field $\mathbb{Q}(\sqrt{d})$. Given $\mathbb{G}$, find $H_{\mathbb{G}} : \{0,\ 1\}^{l(\kappa)} \to \mathbb{G}$, and $H_{\text{prime}} : \{0,\ 1\}^* \to \text{Primes}(2\kappa)$, where $\text{Primes}(2\kappa)$ is the set containing the $2^{2\kappa}$ first prime numbers, e.g., by adapting the SHA family of hashes. Set $\mathsf{EK} = \mathsf{VK} = (\mathbb{G}, H_{\mathbb{G}}, H_{\text{prime}})$.

$\mathsf{VDF.Setup}$ is a randomized algorithm, but with coin given, it can be seen as a determistic algorithm. Note that given the discriminant d, honest parties can check whether VK, EK are valid.

Furthermore, the VDF in [54] is $(\cdot, \sigma)$-sequential for any number p of processors and $\sigma = (1-\xi)\cdot\Delta$, for very small $\xi > 0$. Therefore, we drop the p parameter and say that the Wesolowski VDF with difficulty parameter Δ is $((1-\xi)\cdot\Delta)$-sequential against a Δ-limited adversary that runs in parallel time $\sigma(\Delta)$.

Lemma 4. *The VDF proposed in [54] achieves σ-Sequentiality in the random oracle model (ROM) for any p and for any $\sigma(\Delta) = (1-\xi)\cdot\Delta$.*

Proof. The proof follows from the properties of the evaluation function from Wesolowski's VDF and their proof of Proposition 1 from [54].

Since $H_{\mathbb{G}}$ is a random oracle, $H_{\mathbb{G}}(x_1||x_2)$ is random, and for large enough polynomial $n(\kappa)$, it is also unpredictable by the adversary, despite its choice of x_1. The challenger $\mathcal{C}$ instructions in the construction from Proposition 1 only differ in the check where it performs to abort. In particular, instead of aborting if $x \xleftarrow{\$} \{0,1\}^{l(\kappa)}$ is already queried by the oracle $H_{\mathbb{G}}$, $\mathcal{C}$ now aborts if $x_1\|x_2 : x_1 \xleftarrow{\$} \{0,1\}^{l(\kappa)-n(\kappa)}$ is already queried by the oracle. This still occurs with probability at most $q/2^{-\kappa}$, where $q = O(\mathsf{poly}(\Delta, \kappa))$, so the rest follows. □

A.3 Gradecast

Lemma 5. *Protocol 6 is a g^*-gradecast protocol with round complexity $2g^* + 1$ and communication complexity $O(g^* \cdot (\kappa + \ell) \cdot n^2)$ for messages of length ℓ.*

Proof. The round complexity is by construction. The termination, validity and soundness are proved in [30]. Note that the third condition in round $r \geq 1$ step 1, which limits honest parties to only multicast two valid tuples per round does not change the proof, since any conflicting set of tuples stops the grade increase, and honest parties can receive conflicting values at most one round apart. This extra condition ensures that the total communication per round can be at most $O((\ell+\kappa)\cdot n^2)$ per round. Hence, the total communication complexity of gradecast is $O(g^* \cdot (\kappa + \ell) \cdot n^2)$.

$\mathbf{GC}(m, g^*)$

Round 1:

1. Each party P_i initializes $S_i := \emptyset$, $m_i = \perp$, $\bar{g}_i = 0$.
2. The sender P_s computes a signature $\sigma = \mathsf{sig}_s(m)$ and multicasts (m, σ) to all parties.

Round 2 to $2g^* + 1$ from the perspective of party P_i:

1. For each pair (m', σ') received by the end of the previous round, if $\mathsf{ver}_s(\sigma', m') = 1$ and $m' \notin S_i$, then:
   - Set $S_i := S_i \cup \{m'\}$. If $|S_i| = 1$, set $m_i = m'$.
   - Multicast (m', σ').
   - If multicasted two pairs, stop multicasting this round.
2. If $m_i \neq \perp$ and $|S_i| = 1$, then set $\bar{g}_i := \bar{g}_i + 1$.

Output determination: Each party P_i sets $g_i := \lfloor \bar{g}_i/2 \rfloor$ and outputs (m_i, g_i).

Fig. 6. Gradecast protocol with maximum grade g^*.

A.4 Parallel Vector Gradecast

$\mathcal{F}_{\mathsf{prop}}$ is an ideal functionality required for randomly propagating an average number of $n \cdot p_{prop}$ values in a private fashion, for a probability p_{prop}. We instantiate $\mathcal{F}_{\mathsf{prop}}$ (Figure 7) via the protocol Propagate (Figure 8) as it appears in [51], assuming a CPA-secure public key encryption scheme and erasures, cf. Lemma 8 in [51]. This secure instantiation requires a one-time use of a CPA-secure public key encryption scheme:

Definition 11 (PKE). *A public key encryption (PKE) scheme is a tuple of ppt algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- KeyGen *takes as input the security parameter* κ *and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *is referred to as the public key and* sk *as the private key.*
- Enc *takes as input a public key* pk *and a message* m *and outputs a ciphertext* c*, denoted as* $c \leftarrow \mathsf{Enc}(pk, m)$*.*
- Dec *takes as input a private key* sk *and a ciphertext* c *and outputs a message* m *or a special symbol* $\perp$ *denoting failure to decrypt. We denote* $m := \mathsf{Dec}(\mathsf{sk}, c)$*. The decryption algorithm is deterministic.*

B Inequalities

Lemma 6 (Chernoff's inequality). *Let* $X_1, X_2, \ldots, X_n$ *be independent random binary variables such that, for* $1 \leq i \leq n$, $\mathbb{P}[X_i = 1] =: p_i$*. Then, for* $X := \sum_{i=1}^{n} X_i$ *and* $\mu := \mathbb{E}[X] = \sum_{i=1}^{n} p_i$*:*

$$\mathbb{P}[X \geq (1+\zeta)\mu] \leq e^{-\frac{\zeta^2 \mu}{\zeta+2}}, \qquad 0 \leq \zeta. \tag{1}$$

Functionality: $\mathcal{F}_{\mathsf{prop}}$

Let $p_{prop} = (10/\epsilon + \lambda)/n$. For every party $i \in [n]$, $\mathcal{F}_{\mathsf{prop}}$ keeps a set O_i which is initialized to $\emptyset$. Let M_i be party i's input messages' set.

On input $(\mathsf{SendRandom}, M_i)$ by honest party i:
- For all $x \in M_i$ and for all $j \in [n]$ add (i, x) to O_j with probability p_{prop};
- **return** M_i to adversary $\mathcal{A}$;
- **return** O_i to party i.

On input $(\mathsf{SendDirect}, \mathbf{x}, J)$ by adversary $\mathcal{A}$ (for a corrupted party i):
- Add $(i, x[j])$ to O_j for all $j \in J$;
- **return** O_i to adversary $\mathcal{A}$.

Fig. 7. Functionality $\mathcal{F}_{\mathsf{prop}}$ for parties $P_1, \ldots, P_n$.

Propagate$(\mathsf{SendRandom}, M_i)$

Let $m = 10/\epsilon + \lambda$, $\Lambda_i = 2m \left\lceil \frac{|M_i|}{n} \right\rceil$, $O_i = \emptyset$ and for all $j \in [n]$ let $\mathcal{L}_j = \emptyset$.

1. P_i samples a new pair of keys: $(pk_i^{\mathsf{prop}}, sk_i^{\mathsf{prop}}) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
2. P_i posts pk_i^{prop} on the bulletin-PKI and reads the public keys of the other n parties.
3. For all $x \in M_i$ and for all $j \in [n]$ add x to list $\mathcal{L}_j$ with probability m/n.
4. For all $j \in [n]$:
   - Pad list $\mathcal{L}_j$ to maximum size Λ_i;
   - $ct_j \leftarrow \mathsf{Enc}(pk_j^{\mathsf{prop}}, \mathcal{L}_j)$;
   - Erase $\mathcal{L}_j$ from memory;
5. For all $j \in [n]$ send (ct_j, j) to P_j.
6. Receive messages, say set $\mathcal{C}$.
7. For all $ct \in \mathcal{C}$ decrypt ct using sk_i^{prop} and output a list $\mathcal{L}$ and add $\mathcal{L}$ to O_i.
8. Erase sk_i^{prop} from memory.
9. **return** O_i.

Fig. 8. A secure instantiation of $\mathcal{F}_{\mathsf{prop}}$.

Lemma 7 (Bernoulli's inequality). *For every $x \in \mathbb{R}$ and any positive exponent $r > 0$, it holds:*

$$(1 + x)^r \leq \exp(xr). \tag{2}$$

C Deferred Proofs

C.1 Verifiable Graded Consensus

Proof of Lemma 1. Let Mod-GC be the subprotocol in Figure 1 for a single pair of sender P_i and moderator P_s, between step 1 and step 3.1. We show that Mod-GC satisfies validity, M-validity, soundness, and termination.

***Validity and M-validity*:** Let P_s be the honest moderator and let P_j be an honest party. P_s will gradecast the value obtained from sender P_i correctly, thus $g_{i,j}^{(s)} = g^*$. If P_i is also honest, then $m_i = m_{i,j} = m_{i,j}^{(s)}$, which implies $G_{i,j}^{(s)} = g^*$. Thus, all honest parties P_j output the same tuple (m_i, g^*), and validity holds. However, if P_i is dishonest, then it could be that $m_{i,j} \neq m_{i,j}^{(s)}$. If that is the case, then P_i gradecasts different values to P_s 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 $G_{i,j}^{(s)} = \min\{g_{i,j}^{(s)}, g^* - g_{i,j}\} \geq g^* - 1$. Finally, since P_s is honest, all honest parties receive (and output) the same message from P_s, $m_{i,j}^{(s)} = m_{i,s}$, and M-validity holds.

Before proving the soundness of $\mathsf{Mod\text{-}GC}$, we make the following observation.

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. □

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

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

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

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

$m_{i,k}^{(s)} \neq m_{i,k}$.
- if $g_{i,k} \leq 1$, then there are again two subcases. Subcase 1): $g_{i,k}^{(s)} = g^*$, in which case $G_{i,k}^{(s)} \in \{g^* - 1, g^*\}$ and $G_{i,j}^{(s)} \in \{g^* - 1, g^*\}$ and by the moderator's GC soundness, $m_{i,j}^{(s)} = m_{i,k}^{(s)}$. Subcase 2): $g_{i,k}^{(s)} \leq g^* - 1$, in which case $G_{i,k}^{(s)} = g_{i,k}^{(s)}$, and by GC soundness, it holds that $|G_{i,j}^{(s)} - G_{i,k}^{(s)}| \leq 1$, as well as that if both $G_{i,j}^{(s)}, G_{i,k}^{(s)}$ are greater or equal to 1, then $m_{i,j}^{(s)} = m_{i,k}^{(s)}$.

Case 3. $m_{i,j}^{(s)} \neq m_{i,j}$ and $m_{i,k}^{(s)} \neq m_{i,k}$. Then $G_{i,j}^{(s)} = \min\{g_{i,j}^{(s)}, g^* - g_{i,j}\}$ and $G_{i,k}^{(s)} = \min\{g_{i,k}^{(s)}, g^* - g_{i,k}\}$, and we can use Proposition 1 to get that $|G_{i,j}^{(s)} - G_{i,k}^{(s)}| \leq 1$. If both $G_{i,j}^{(s)}, G_{i,k}^{(s)}$ 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 $m_{i,j}^{(s)} = m_{i,k}^{(s)}$.

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

This completes the proof of Lemma 1. □

Proof of Theorem 1. We argue the graded validity, graded agreement and termination of Toss.

***Graded validity*:** Since each Mod-GC instance called by a honest moderator P_s achieves M-validity, then it holds that any honest party P_j has $(m_{i,j}^{(s)}, G_{i,j}^{(s)})$ for any sender P_i, with $G_{i,j}^{(s)} \in \{g^*, g^* - 1\}$. This implies that all honest parties will also have $G_j^{(s)} \in \{g^*, g^* - 1\}$. Finally, all honest parties form their output strings $M_j^{(s)}$ in step 2.2 with all $m_{i,j}^{(s)}$ for $i \in [n]$ since all associated grades are strictly positive when the moderator is honest. Again by M-validity of Mod-GC, another honest party P_k will have $m_{i,k}^{(s)} = m_{i,j}^{(s)}$ for all $i \in [n]$, immediately implying that $M_j^{(s)} = M_k^{(s)}$ for any honest moderator P_s.

***Graded agreement*:** We reformulate the statement as follows. Let P_j, P_k be two honest parties outputting $(M_j^{(s)}, G_j^{(s)})$ and $(M_k^{(s)}, G_k^{(s)})$, respectively. For any moderator P_s, after the execution of the protocol, it holds that $|G_j^{(s)} - G_k^{(s)}| \leq 1$. Moreover, if $G_j^{(s)} > 1$ then $M_j^{(s)} = M_k^{(s)}$, otherwise if $G_j^{(s)} = 1$, then either $M_j^{(s)} = M_k^{(s)}$ or $G_k^{(s)} = 0$. We prove this below.

Let α such that $\min_{i \in [n]}\{G_{i,j}^{(s)}\} = G_{\alpha,j}^{(s)}$. Similarly, let β so that $\min_{i \in [n]}\{G_{i,k}^{(s)}\} = G_{\beta,k}^{(s)}$. Suppose without loss of generality that $G_{\beta,k}^{(s)} \geq G_{\alpha,j}^{(s)}$. This implies also

that $G^{(s)}_{\alpha,k} \geq G^{(s)}_{\alpha,j}$, since otherwise, $G^{(s)}_{\alpha,k} < G^{(s)}_{\beta,k}$, contradiction. Then, $|G^{(s)}_k - G^{(s)}_j| = |\min_{i\in[n]}\{G^{(s)}_{i,k}\} - \min_{i\in[n]}\{G^{(s)}_{i,j}\}| = |G^{(s)}_{\beta,k} - G^{(s)}_{\alpha,j}| = G^{(s)}_{\beta,k} - G^{(s)}_{\alpha,j} \leq G^{(s)}_{\alpha,k} - G^{(s)}_{\alpha,j} = |G^{(s)}_{\alpha,k} - G^{(s)}_{\alpha,j}| \leq 1$, from the soundness part of Lemma 1.

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

***Termination*:** Follows immediately from the termination of all Mod-GC instances and the fact that the sampling in step 0 is guaranteed to return a result.

This completes the proof of Theorem 1. □

Proof of Theorem 2. We need to show that Π_{VGC} satisfies termination, graded agreement, graded validity, indistinguishable randomness, correctness and soundness. Theorem 1 covers the first three properties, while the results in the Lemmata below prove the rest of the properties. We first prove an intermediate result about strictly positive grades and about the σ-indistinguishable property of the VDF construction with untrusted setup.

Lemma 8. *If a party P_j is graded as $G^{(j)}_i \geq 1$ by an honest party P_i in Toss, then $M^{(j)}_i$ contains at least one entry m_k from an honest party P_k.*

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

Lemma 9. *Given access to the bulletin-PKI, the VDF construction with setup $\mathsf{VDF.Setup}(1^\kappa, \Delta, \mathsf{coin}_{\mathsf{VDF}})$ achieves σ-indistinguishable randomness.*

Proof. All parties obtain $\mathsf{coin}_{\mathsf{VDF}}$ as $H_d(\|_{i\in[n]}\mathsf{PK}_i)$ from the PKI. Given the use of the random oracle H_d on a pseudorandom input (the keys of the honest parties have randomness in them), $\mathsf{coin}_{\mathsf{VDF}}$ is random. Then, parties can deterministically obtain $\mathsf{EK}_j = \mathsf{VK}_j = (\mathbb{G}, H_{\mathbb{G}}, H_{\text{prime}})$ from $\mathsf{VDF.Setup}(1^\kappa, \mathsf{pp}, \mathsf{coin}_{\mathsf{VDF}})$ (Appendix A.2), based on $\mathsf{coin}_{\mathsf{VDF}}$, as specified in Section 3.2.

The idea of a distributed setup for the class group parameters is natural, see also [14, Sec 3.2] and [11, Sec 2.2]. The adversary can select its public keys after seeing honest parties' public keys, in an attempt to obtain a discriminant d' for which it somehow knows the group order for the associated class group $\mathbb{G}'$. But the adversary only gets access to a polynomial number of queries to the random oracle H_d, therefore its advantage in knowing the group order and being able to evaluate faster than Δ_{VDF} is also a negligible function $\mathsf{negl}(\lambda)$, as otherwise we would obtain a distinguisher for the unknown order assumption and the time-lock assumption [54]. Therefore, the VDF with untrusted setup based on a random oracle applied on pseudorandom inputs is also σ-sequential. □

Lemma 10. *Π_{VGC} achieves σ-indistinguishable randomness, for $\sigma(\Delta_{\mathsf{VGC}}) = (1-\xi)\cdot\Delta_{\mathsf{VGC}}$ and $\Delta_{\mathsf{VGC}} = 2\cdot\Delta_G$.*

Proof. Recall the Δ-indistinguishable randomness game in Definition 5. The adversary $\mathcal{A}$ receives Δ_{VGC} and the public keys of the honest users from the challenger $\mathcal{C}$, and sets the public keys of the currently corrupted parties $\mathcal{M}$ (representing the malicious set). Every time the adversary corrupts a new party, it will add its identity to $\mathcal{M}$, but cannot change the public keys.

The public keys (PKI) do not formally include the VDF evaluation and verification keys, since these are obtained by applying a deterministic algorithm on all public keys. All parties can deterministically obtain $\mathsf{EK}_j = \mathsf{VK}_j = \mathsf{EK} = \mathsf{VK} = (\mathbb{G}, H_{\mathbb{G}}, H_{\text{prime}})$, based on the output of a random oracle that takes as input the PKI. Therefore, for each $j \in \mathcal{M}$, EK_j and VK_j can be checked by the challenger to be valid, since they should be the same as what the challenger obtains in $\mathsf{Process}$. Moreover, the adversary can start evaluating the VDF on any string it wishes after it has the evaluation key (which is obtained after the PKI is published).

The challenger and adversary start executing Toss. The challenger selects the input seeds $m_i \in \{0,1\}^{q(\kappa)}$ for every $P_i \in [n] \setminus \mathcal{M}$ and sends them to the adversary in the first round of GC. The event that the adversary guesses the input seed m_i of an honest party before seeing it in round 1 has probability $\mathsf{negl}(\kappa)$. A union bound over all such events still yields a negligible probability $p_0 = \mathsf{negl}(\kappa)$. Therefore, any VDF evaluation the adversary has computed so far is independent of the strings m_i.

From round 1, the adversary can start evaluating $\mathsf{VDF.Eval}$ on any combination of strings m_i of the honest parties and any other strings. Because the VDF is σ-sequential, the adversary will obtain the evaluations only after Toss has completed, since the duration of Toss is less than Δ_{VGC}. The VDF does not have trapdoors despite the untrusted setup as shown above, therefore the adversary cannot obtain the output faster.

The adversary participates in the GC instances in step 1 and step 2 of Protocol 1 with whatever behavior $\mathcal{A}_0$ it chooses. In step 3 of Protocol 1, the remaining honest parties P_i (at least ϵn), set for every $j \in [n]$ the output $(M_i^{(j)}, G_i^{(j)})$. At any point, the adversary also chooses output values $(M_j^{(j)})$ for $P_j \in \mathcal{M}$, based

on all values it has seen so far and all computations done so far. It starts $\mathsf{Process}$ using behavior $\mathcal{A}_1$ to obtain $w_A^{(k)}$, for any party P_k, either honest or corrupted.

By the fact that $\mathcal{A}$ is $\sigma(\Delta_{\mathsf{VGC}})$-parallel time limited and the VDF is σ-sequential (Lemmata 4 and 9), it holds that for any value $M^{(i)}$ obtained after the start of Toss as a function of the honest parties, the advantage of the adversary guessing $w^{(i)} = \mathsf{VDF.Eval}_{\mathsf{EK}_i}(H(M_i^{(i)}))$ for honest P_i is $p_1 = \mathsf{negl}(\kappa)$.

We now turn to corrupted parties P_j, $j \in \mathcal{M}$. To obtain a biased output $w^{(j)}$, by the σ-sequentiality of the VDF, the adversary must have computed it on values not depending on the honest parties' random seeds, with overwhelming probability. Lemma 8 shows that the only way a corrupted party can obtain a strictly positive grade from another honest party is if it relays correctly at least one honest string corresponding to an honest party. This implies that each value $M_j^{(j)}$ for which $\mathcal{A}$ could have guessed $w^{(j)}$ has to have grade $G_i^{(j)} = 0$ by all other honest parties.

Therefore, the advantage the adversary $\mathcal{A}$ can have in winning the Δ_{VGC}-indistinguishability game is at most $p_0 + p_1 = \mathsf{negl}(\kappa)$, so Π_{VGC} is $\sigma(\Delta_{\mathsf{VGC}})$-indistinguishable. □

Lemma 11. *Π_{VGC} achieves soundness.*

Proof. In the definition of the soundness of VGC, the inputs $x^{(j)}$ are fixed to be obtained through Toss for an honest party, and can be any string chosen by the adversary for a corrupted party (if it does not care to have a positive grade). We want to show that an honest party P_i will never validate a string w_j and a proof ρ_j, for which $(w_j, \rho_j) \neq \mathsf{Process}(\mathsf{pp}, x_i^{(j)})$. Note that even if the locally held string $x_i^{(j)}$ at P_i differs from the string $x^{(j)}$ obtained by a different party from Toss, P_i would not accept a pair $(w_j, \rho_j) \neq \mathsf{Process}(\mathsf{pp}, x^{(j)})$.

The result follows from the soundness of the VDF construction (Definition 9) with difficulty parameter Δ_{VGC}, which holds for *any* VDF inputs, outputs and proofs that a $\sigma(\Delta_{\mathsf{VGC}})$-limited adversary chooses, as long as the adversary does not set the output and proof to what was obtained from $\mathsf{VDF.Eval}$. Therefore, as long as the adversary does not honestly follow Toss to obtain a string $x^{(j)} = x_i^{(j)}$ and does not compute $(w_j, \rho_j) \leftarrow \mathsf{Process}(\mathsf{pp}, x_i^{(j)}) = \mathsf{VDF.Eval}_{\mathsf{EK}_j}(x_i^{(j)})$, an honest party P_i will return 0 as the output $\mathsf{VGC.Verify}(\mathsf{pp}, x_i^{(j)}, w_j, \rho_j)$ with overwhelming probability. □

Lemma 12. *Π_{VGC} achieves termination and correctness.*

Proof. The termination of Toss was proved above, and the termination of $\mathsf{Process}$ is guaranteed by the property of $\mathsf{VDF.Eval}$ that an output is generated after time Δ_{VGC}. Correctness is inherited from the correctness of the VDF construction (Definition 9). □

This concludes the proof of Theorem 2. □

C.2 Broadcast

We first give the intuition of the proof of Theorem 3. The Π_{VGC} protocol achieves indistinguishable randomness for a difficulty parameter of Δ_{VGC}, graded validity, graded agreement, termination, correctness and soundness. Therefore, we are guaranteed that by the start of Stage 1, each party P_i has a random string w_i that they feed to the VRF, and a corresponding proof π_i. Moreover, by the start of Stage 1, each party P_i also holds graded strings $(M_i^{(j)}, G_i^{(j)})$ for each other party P_j. Honest parties are certain that the grades for the local strings they hold for the other parties differ by at most 1, and moreover, local strings with grades greater than 2 are the same among honest parties.

The indistinguishable randomness of the Π_{VGC} guarantees that malicious parties cannot bias the output of Process while having grade greater than 0 in the views of honest parties. Each party evaluates only one VDF, the one corresponding to its own election process, and it only reveals it when it wants to prove it is elected in a bit-committee—the election depends both on its local secret key and the beacon. Furthermore, the VRF satisfies uniqueness, provability, pseudorandomness and extended pseudorandomness. Therefore, if malicious parties do not compute their VRF on the bit value concatenated with the random string, they cannot produce a valid proof for membership in the bit-committee that will be accepted by the honest parties.

We note that to streamline this proof, we prove this result as a composition of the properties of the protocols and algorithms for the online setup, election and voting, that comprise the protocol; nevertheless, it could be proved as a monolithical protocol that does not assume composition results.

We first prove some intermediate results on valid batches and certificates, and on the number of honest and dishonest parties elected in the committees. Notice that parties with associated grades equal to zero from the perspective of an honest party P_i do not affect the outcome of the election, since the conditions of validity for the allowed $(r, \cdot)$-batches always require a strictly positive grade. Therefore, we focus on strictly positive grades.

Lemma 13. *If a party P_j is graded with $G_i^{(j)} \geq 1$ by P_i as a result of Toss, then P_j can add a valid contribution to a batch batch'_b with an associated valid certificate cert'_b, only if P_j correctly executes Toss, Process and* VRF.Prove *to obtain y_j, unless with negligible probability $\mathsf{negl}(\kappa)$.*

Proof. Let the adversary submit a maliciously generated tuple $(w_j, \rho_j, y_j, \pi_j)$ to a certificate $\mathsf{cert}'_b = \mathsf{cert}_b || (w_j, \rho_j, y_j, \pi_j)$ associated to a $(r+1, 1)$-batch $\mathsf{batch}'_b = \mathsf{batch}_b || \mathsf{sig}_j(b)$, where batch_b and cert_b were valid. Towards a contradiction, assume that the resulting batch and certificate are valid with more than negligible probability. Party P_j needs to be corrupted, otherwise, the unforgeability of the signatures would immediately prevent the adversary from impersonating an honest party P_j and batch'_b would not be validated.

Recall that an honest party P_i validates a certificate tuple $(w_j, \rho_j, y_j, \pi_j)$ associated to a signature $\mathsf{sig}_j(b)$ if the following hold:

(i) $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||w_j), y_j, \pi_j) = 1$,
(ii) $\mathsf{VGC.Verify}(\mathsf{pp}, M_i^{(j)}, w_j, \rho_j) = 1$, where $(M_i^{(j)}, \cdot) \leftarrow \mathsf{Toss}(\mathsf{pp})$, and
(iii) $y_j \leq \mathsf{bound}_{\epsilon,\delta}$.
Maliciously generated strings y_j that do not satisfy condition (iii) are trivially discarded, so we assume (iii) holds for y_j.

Let us look at condition (ii). The soundness of Π_{VGC} (Lemma 11) states that a malicious party P_j can compute $(w^{(j)}, \rho^{(j)})$ in another way than as the output of $\mathsf{Process}$ on a string equal to $M_i^{(j)}$, so that P_i accepts $\mathsf{VGC.Verify}(\mathsf{pp}, M_i^{(j)}, w_j, \rho_j) = 1$, with only negligible probability. Furthermore, since the evaluation in $\mathsf{Process}$ is deterministic, and Toss satisfies graded agreement and graded validity by Theorem 1, the outputs $w_i^{(j)} \leftarrow \mathsf{Process}(\mathsf{pp}, M_i^{(j)})$, where $w_j := w_j^{(j)}$, with the grades $G_i^{(j)}$ obtained from Toss, also satisfy graded agreement and graded validity over all parties P_i, which holds unconditionally (since we assume idealized signatures). Note that no party has to compute $\mathsf{Process}$ for all n strings it outputs from Toss and this is just a global view. Therefore, if condition (ii) of the certificate check holds with more than negligible probability for a maliciously generated certificate tuple, since the graded validity and agreement properties hold unconditionally, then the adversary can break the soundness of Π_{VGC} (which can happen with probability $\mathsf{negl}(\kappa)$). We remark that this implies that if an honest party P_i receives a value w_j for which $\mathsf{VGC.Verify}(\mathsf{pp}, M_i^{(j)}, w_j, \rho_j) = 0$, then either P_i has grade 0 associated to P_j, which contradicts the assumption from the statement that $G_i^{(j)} \geq 1$, or $(w_j, \rho_j) \neq \mathsf{Process}(\mathsf{pp}, M_i^{(j)})$ by the soundness property.

Assume now that the given tuple passes conditions (ii) and (iii), and let us look at condition (i). Passing condition (ii) means that the string w_j has grade ≥ 1 and is unbiased, with overwhelming probability, against a $\sigma(\Delta_{\mathsf{VGC}})$-limited adversary, according to the indistinguishable randomness of Π_{VGC} (Lemma 10). Note that in the ROM, applying a hash function implies that $H(b||w_i)$ are random, not just unpredictable, as required in the VRF definition.

Assume first the adversary corrupted P_j after it posted its keys on the public bulletin-board, so the key pair $(\mathsf{SK}_j, \mathsf{PK}_j)$ was honestly generated. An adversary for which an honest P_i would return $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||w_j), y_j, \pi_j) = 1$, for maliciously generated y_j and π_j, breaks the uniqueness and provability of the VRF (Definition 8), which can happen with only negligible probability. Second, assume P_j was corrupted from the beginning, and the key pair was not honestly generated, but satisfies $\mathsf{VRF.Validate}_{\mathsf{PK}_j}(1^\kappa) = 1$. Still, if y_i was not computed as the output of $\mathsf{VRF.Prove}$ on the bit value concatenated with the random string w_j, and if an honest P_i returns $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||w_j), y_j, \pi_j) = 1$, the adversary would break the extended pseudorandomness property of VRF since w_i is random, and this happens with only negligible probability.

Therefore, condition (i) on the certificate check holds only with negligible probability for a maliciously generated tuple, contradicting the assumption made in the beginning of the proof. □

Corollary 1. *The adversary cannot bias y_j for a corrupted party P_j such that it is less than the bound $\mathsf{bound}_{\epsilon,\delta}$.*

Proof. The pseudorandomness and extended pseudorandomness properties of the VRF (Definition 8) when applied on the random input $H(b||w_j)$ means that $(y_j, \pi_j) \leftarrow \mathsf{VRF.Prove}_{\mathsf{PK}_j}(H(b||w_j))$ is pseudorandom. Therefore, P_j cannot bias y_j such that it is less than the bound $\mathsf{bound}_{\epsilon,\delta}$. □

This corollary enables the proofs of Lemma 15, since it ensures that unless with negligible probability $\mathsf{negl}(\lambda)$, the election succeeds as dictated by p_{mine}.

Lemma 14 (Honest Lower Bound). *For $p_{mine} = \min\left\{1, \frac{1}{\epsilon n}\log\left(\frac{2}{\delta}\right)\right\}$, at least one honest party self-elects in a committee for a bit b with probability $1-\delta/2$.*

Proof. There are at least ϵn parties that are forever honest by assumption. Let X denote the number of honest parties that elect themselves in a committee for a given bit b. The expected value of X is:

$$\mathbb{E}[X] = \epsilon n \cdot p_{\text{mine}} = \epsilon n \cdot \min\left\{1, \frac{1}{\epsilon n}\log\left(\frac{2}{\delta}\right)\right\} = \begin{cases} \epsilon n & \text{if } \log\left(\frac{2}{\delta}\right) \geq \epsilon n \\ \log\left(\frac{2}{\delta}\right) & \text{if } \log\left(\frac{2}{\delta}\right) < \epsilon n \end{cases}.$$

The first case corresponds to all honest parties always getting elected with $p_{\text{mine}} = 1$. In the second case, the probability that no honest party can ever self-elect is given by:

$$\mathbb{P}[X = 0] \leq (1 - p_{\text{mine}})^{\epsilon n} \leq \exp(-\epsilon n p_{\text{mine}}) = \exp(\log(2/\delta)) = \delta/2.$$

The second inequality holds by Bernoulli's inequality (equation (2) in Appendix B).

The VRF constructions we consider use random oracles, so we can assume the output is distributed uniformly at random. Then, the statement in the Lemma holds, $\mathbb{P}[X \geq 1] \geq 1 - \delta/2$. Since δ is negligible in the security parameter λ, the result holds with overwhelming probability in λ. □

Lemma 15 (Dishonest Upper Bound). *For $p_{mine} = \min\left\{1, \frac{1}{\epsilon n}\log\left(\frac{2}{\delta}\right)\right\}$, at most $g^*/2 = \left\lceil \frac{1}{\epsilon} \cdot \log\left(\frac{2}{\delta}\right)\right\rceil$ dishonest parties can self-elect in a committee for a bit b with probability $1 - \delta/2 - \mathsf{negl}(\kappa)$, for any $\epsilon \in (0, 1)$.*

Proof. There are at most $(1-\epsilon)n$ parties that can be corrupted at any time by assumption. Let X denote the number of dishonest parties managing to elect themselves in a committee for a given bit b. Even if nodes are corrupted adaptively, the election probability does not change. The expected value of X is:

$$\begin{aligned} \mathbb{E}[X] = (1-\epsilon)n \cdot p_{\text{mine}} &= \epsilon n \cdot \min\left\{1, \frac{1}{\epsilon n}\log\left(\frac{2}{\delta}\right)\right\} \\ &= \begin{cases} \epsilon n & \text{if } \log\left(\frac{2}{\delta}\right) > \epsilon n \\ \frac{1-\epsilon}{\epsilon}\log\left(\frac{2}{\delta}\right) & \text{if } \log\left(\frac{2}{\delta}\right) \leq \epsilon n \end{cases}. \end{aligned}$$

The first case corresponds to all parties always getting elected with $p_{\text{mine}} = 1$, but which also means $g^*/2 > 2n$, which is a case we are not interested in, since we want g^* to be sublinear.

Therefore we focus on the second case. For simplicity, set $R := g^*/2$. We want to use Chernoff's inequality (equation (1) in Appendix B). Therefore, setting $R = (1+\zeta) \cdot \mathbb{E}[X]$ yields $\zeta = R/\mathbb{E}[X] - 1$.

The probability that more than R dishonest parties can ever self-elect is given by:

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

where $*$ holds if

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

We want to find ζ that satisfies (3) for any value of $\epsilon \in (0,1)$ and from it find the minimum R for which $\mathbb{P}[X \geq R] \leq \delta/2$.

The roots of (3) are $\frac{\epsilon-\sqrt{8\epsilon-7\epsilon^2}}{2(1-\epsilon)}$ and $\frac{\epsilon+\sqrt{8\epsilon-7\epsilon^2}}{2(1-\epsilon)}$ and inequality holds outside of the roots. To account for any value of $\epsilon \in (0,1)$, we choose to set

$$\begin{aligned}R &\geq \left(1+\frac{\epsilon+\sqrt{8\epsilon-7\epsilon^2}}{2(1-\epsilon)}\right)\cdot\mathbb{E}[X] = \frac{2-\epsilon+\sqrt{8\epsilon-7\epsilon^2}}{2(1-\epsilon)}\cdot\frac{1-\epsilon}{\epsilon}\cdot\log\left(\frac{2}{\delta}\right)\\ &= \frac{2-\epsilon+\sqrt{8\epsilon-7\epsilon^2}}{2\epsilon}\cdot\log\left(\frac{2}{\delta}\right) \geq \frac{2}{2\epsilon}\cdot\log\left(\frac{2}{\delta}\right).\end{aligned}$$

Thus we get $R := \lceil \frac{1}{\epsilon}\cdot\log\left(\frac{2}{\delta}\right)\rceil$, obtaining the $g^*/2$ value in the statement.

Since the VRF verification can fail with probability negligible in the computational parameter κ, the result holds with probability $1 - \delta/2 - \mathsf{negl}(\kappa)$. □

Proof of Theorem 3. We now prove the main result of the paper. First, termination holds by the termination of the Π_{VGC}, where Toss and Process act like an online setup, and by construction of the subsequent $g^*/2$ stages. Then, we prove validity and consistency.

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 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 $\mathsf{batch}_{\bar{b}_s}$.

Then, by the correctness of Π_{VGC} (Definition 4, proved in Theorem 2) and the correctness of the VRF (Definition 8), honest parties will correctly form valid batches on b, such that the after g^* rounds, all honest parties P_i will have $\mathsf{Extracted}_i = \{b_s\}$ unless with negligible probability. □

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

Proof. Suppose that an honest party P_i adds message b to $\mathsf{Extracted}_i$ at stage r. We prove that by the end of the protocol, all honest parties add b to their $\mathsf{Extracted}$ sets with overwhelming probability. Assume first that until the last round, there are no more than $g^*/2$ signatures in a valid batch.

Case 1. P_i adds message b to $\mathsf{Extracted}_i$ during the first round of stage $r < g^*/2$: Then, b is accompanied by a valid $(r,1)$-batch, say batch_b. Also, b is accompanied by a valid certificate for batch_b, say cert_b. Note that by Lemma 13, P_i accepts a cert_b as valid for batch_b if it was honestly formed, unless with probability $\mathsf{negl}(\kappa)$. So, for all signatures $\mathsf{sig}_k(b) \in \mathsf{batch}_b$ it holds that $G_i^{(k)} \geq g^* - 2r + 1$. Thus, during the second round of stage r, all honest parties receive $[b, \mathsf{batch}_b, \mathsf{cert}_b]$. Note that with probability $1 - \delta/2 - \mathsf{negl}(\kappa)$ by Lemma 14, at least one honest party is in the b-committee.

Such a party, say P_j, checks that batch_b is a valid $(r,2)$-batch for b and that cert_b is a valid associated certificate. For all signatures $\mathsf{sig}_k(b) \in \mathsf{batch}_b$, from the graded agreement property of Π_{VGC}, it holds that $|G_i^{(k)} - G_j^{(k)}| \leq 1$. Since $G_i^{(k)} \geq g^* - 2r + 1$, it holds that $G_j^{(k)} \geq g^* - 2r$. From the graded agreement property of Π_{VGC}, it also holds that $M_i^{(k)} = M_j^{(k)}$ for all $k \in [n] : \mathsf{sig}_k(b) \in \mathsf{batch}_b$, since $G_i^{(k)}, G_j^{(k)} > 1$. So, $\mathsf{VGC.Verify}(\mathsf{pp}, M_j^{(k)}, w_k, \rho_k) = \mathsf{VGC.Verify}(\mathsf{pp}, M_i^{(k)}, w_k, \rho_k) = 1$. Moreover, $\mathsf{VRF.Verify}_{\mathsf{PK}_k}(H(b||w_k), y_k, \pi_k) = 1$ and $y_k \leq \mathsf{bound}_{\epsilon,\delta}$, by uniqueness of the VRF. Therefore, cert_b is considered a valid certificate for batch_b from P_j. So, P_j adds b to its $\mathsf{Extracted}_j$ set and adds its own signature to the batch, producing a $(r+1,1)$-batch batch'_b, and adds its proof to the updated certificate cert'_b. This batch'_b is valid for all honest parties P_l, because for all signatures $\mathsf{sig}_k(b) \in \mathsf{batch}'_b$ either 1) $k \neq j$ and from graded agreement of Π_{VGC}, $|G_l^{(k)} - G_j^{(k)}| \leq 1$, meaning that $G_l^{(k)} \geq g^* - 2r - 1$, or 2) $k = j$ and from the graded validity property of Π_{VGC}, $G_l^{(j)} \geq g^* - 1$. For similar reasons as above, the updated certificate is also valid. Thus, during the first round of stage $r+1$, all honest parties observe a valid $(r+1,1)$-batch for message b, accompanied by a valid certificate and thus add b to their $\mathsf{Extracted}$ sets.

Case 2. P_i adds message b to $\mathsf{Extracted}_i$ during the second step of stage $r < g^*/2$: Then, P_i holds a valid $(r,2)$-batch batch_b and a valid certificate for it, say cert_b. P_i is in the committee for b, thus P_i adds their own signature to batch_b, creating a valid $(r+1,1)$-batch batch'_b, and also adds $(w_i, \rho_i, y_i, \pi_i)$ to an updated certificate cert'_b. So, P_i sends $(b, \mathsf{batch}'_b, \mathsf{cert}'_b)$ to all parties. Thus, during the first round of stage $r+1$, all honest parties observe a valid $(r+1,1)$-batch for b, and a valid certificate (validity holds from the same arguments as in **Case 1.**) and add b to their $\mathsf{Extracted}$ sets.

Case 3. P_i adds message b to $\mathsf{Extracted}_i$ during stage $r = g^*/2$: Then, P_i holds a valid $(g^*/2,1)$-batch, i.e. a batch_b of more than $g^*/2$ signatures from parties P_j, where one of the signatures is from P_s and the rest of them have grade $G_i^{(j)} \geq 1$, and an associated certificate cert_b. By Lemma 15, at least one of the signatures is from another honest party, so every honest party received this

valid $(g^*/2, 1)$-batch batch_b and the associated valid cert_b, and added b to their $\mathsf{Extracted}$ set by this stage.

Now, assume that at any point in the protocol, the batch batch_b received already contained $g^*/2$ valid signatures with grades $G_k^{(j)} \geq g^* - 1$. Since it is a valid $(r, 1)$-batch, then it will also be a valid $(r+1, 1)$-batch, and more specifically, a valid $(g^*/2, 1)$-batch (the rules for the valid certificates also hold). With overwhelming probability, there cannot be $g^*/2$ malicious parties in the committee (Lemma 15) and there is at least one honest party in the b-committee (Lemma 14), so limiting the size of the transmitted batches to $g^*/2$ does not impact consistency. □

This concludes the proof of Theorem 3. □

C.3 Parallel Vector Gradecast

Proof of Lemma 2. The proof can be adapted from the proof of Theorem 2 in [51]. Each honest party always calls $\mathsf{couples}_k$ before sending. The use of $\mathsf{couples}_k$ for the input sets means that each party at any point can input at most the set $\mathsf{couples}_k(\mathcal{M})$. Thus, the number of bits sent by a single party is, with probability $1 - \mathsf{negl}(\lambda)$:

$$\sum_{i=1}^{\lceil \log \epsilon n \rceil} O\big(m \cdot (n + |\mathsf{couples}_k(\mathcal{M})|) \cdot s\big) =$$
$$O(m \cdot \log(\epsilon \cdot n) \cdot \max\{n, |\mathsf{couples}_k(\mathcal{M})|\} \cdot s).$$ □

Proof of Lemma 3. We prove validity and consistency separately.

***Validity*:** Let P_s be an honest sender with some value $m^*_{(s,k)}$ for $k \in$ casts, and let P_i be any honest party still honest by the end of the protocol. In the first round, all parties receive $(m^*_{(s,k)}, \sigma^*_{(s,k)})$. The adversary is unable to forge signatures of honest parties, thus honest party P_i holds $|S^i_{(s,k)}| = 1$ and $m^i_{(s,k)} = m^*_{(s,k)}$ at all times throughout the protocol. Therefore, at the end of the protocol it holds that $g^i_{(s,k)} = 2g^*$ and P_i outputs $(m^*_{(s,k)}, g^*)$.

***Consistency*:** Assume an honest party P_i for some ($s \in$ senders, $k \in$ casts) with output grade $g^i_{(s,k)} \geq 1$. So, $\bar{g}^i_{(s,k)} \geq 2g^i_{(s,k)}$ at the end of the protocol. Assume that some round r is the round during which P_i adds message $m^i_{(s,k)}$ in $S^i_{(s,k)}$. We claim that if some honest party P_j gets $(m^j_{(s,k)}, \sigma^j_{(s,k)})$, $m^j_{(s,k)} \neq m^i_{(s,k)}$ in round r' with valid signature $\sigma^j_{(s,k)}$, then $r' > r + 2g^i_{(s,k)} - 3$. Assume that $r' \leq r + 2g^i_{(s,k)} - 3$. P_j is honest, thus it calls $\Pi_{\mathsf{CV}}(\mathbf{M}_j, \emptyset, \cdot)$ during round $r' + 1$, with $m^j_{(s,k)} \in \mathbf{M}_j$. From Lemma 2, by the end of round $r' + 1$ all honest parties received two distinct messages for (s, k); either exactly $m^i_{(s,k)}, m^j_{(s,k)}$, if no other valid messages are propagated from adversarial sender s for its k-th cast, or any

two valid messages else. Thus, by the end of step 1 of round $r'+2$, it holds that $|S^i_{(s,k)}| \geq 2$ and thus $\bar{g}^i_{(s,k)} \leq r'+2-r \leq 2g^i_{(s,k)} - 1$, which is a contradiction. So, $r' > r + 2g^i_{(s,k)} - 3$.
Party P_i, therefore calls $\Pi_{\mathsf{CV}}(\mathbf{M}_i, \emptyset, \cdot)$ during round r, with $m^i_{(s,k)} \in \mathbf{M}_i$. Thus, by the end of round r all honest parties receive at least one message. If some honest party does not receive $m^i_{(s,k)}$ by the end of round r, then by the construction of Π_{CV} each honest party received at least two distinct messages for (s,k), with probability $1 - \mathsf{negl}(\lambda)$. From the previous claim it then holds, $2g^i_{(s,k)} < 3$, i.e. $g^i_{(s,k)} \leq 1$ and thus from the assumption: $g^i_{(s,k)} = 1$. At the same time, from the previous claim, no honest party P_j receives $m^j_{(s,k)} \neq m^i_{(s,k)}$ before round $r' > r-1$. Since by the end of round r it is $|S^j_{(s,k)}| \geq 2$, then $\bar{g}^j_{(s,k)} \leq 1$ if P_j outputs $m^j_{(s,k)}$, i.e. $g^j_{(s,k)} = 0$, if $m^i_{(s,k)} \neq m^j_{(s,k)}$.

Else, if all honest parties receive $m^i_{(s,k)}$ by the end of round r, then let us consider the value of $\bar{g}^j_{(s,k)}$ at the end of the protocol:

- If $g^i_{(s,k)} \geq 2$, then $\bar{g}^j_{(s,k)} > (r + 2g^i_{(s,k)} - 3) - r \geq (r + 2g^i_{(s,k)} - 3) - r + 1 = 2g^i_{(s,k)} - 2$. Thus, P_j outputs m_i with grade $g^j_{(s,k)} > g^i_{(s,k)} - 1$.
- If $g^i_{(s,k)} = 1$, then from the previous claim, no honest party P_j receives $m^j_{(s,k)} \neq m^i_{(s,k)}$ before round $r-1$. Since by the end of round r P_j receives $m^i_{(s,k)}$ (it is $m^i_{(s,k)} \in |S^j_{(s,k)}|$ by round $r+1$), then either $g^j_{(s,k)} = 0$ or $m^j_{(s,k)} = m^i_{(s,k)}$.

The total number of rounds for our protocol is 1 (for Round 1) $+2g^*$ (Rounds 2 to $2g^*+1$), where for each of the latter $2g^*$ rounds, a call to Π_{CV} is made, adding $\lceil \log \epsilon n \rceil$ additional rounds in each, leading to the stated total number of rounds.

The total communication complexity is $O(g^* \cdot n \log \epsilon n \cdot \max\{n, |\mathsf{couples}_{k_{sc}}(\mathcal{M})|\} \cdot m \cdot s)$, where $\mathcal{M}$ is the set containing all valid messages from pairs of (senders, casts). Since, k_{sc} is the prefix size defined exactly to differentiate between messages of different ($j \in$ senders, $k \in$ casts), it holds that $|\mathsf{couples}_{k_{sc}}(\mathcal{M})| = 2|\text{senders}| \cdot |\text{casts}|$ and the proof follows. □

C.4 Communication Reduction

We denote by Π'_{Toss} the protocol obtained by replacing in the Toss protocol from Figure 1 the call to GC in step 1.1. with a call to Parallel Vector GC. We have the following lemma.

Lemma 18. *Protocol Π'_{Toss} is a Toss protocol with maximum grade g^* satisfying Graded Validity and Graded Agreement and Termination as in Definition 4, with probability $1 - \mathsf{negl}(\lambda_{\mathsf{g}})$.*

Proof. The proof follows similarly to the proof of Theorem 1, with the use of Lemma 3 instead of Lemma 1 in the respective arguments. Since the result of

Lemma 3 holds with probability $1 - \mathsf{negl}(\lambda_g)$, the current result inherits this same probability of success. □

As described in Section 6, the use of Π'_{Toss} leads to a new VGC protocol, called Π'_{VGC}, that is used during Π'_{BC}. Mirroring the previous analysis, we can prove the following lemma for Π'_{VGC}.

Lemma 19. *Protocol Π'_{VGC} is, with probability $1 - \mathsf{negl}(\lambda_g)$, a verifiable graded consensus protocol on random strings, cf. Definition 4, against an adaptive Δ'_{VGC}-limited adversary who runs in at most $(1-\xi)\cdot\Delta'_{VGC} = \Delta_G + \Delta'_G$ parallel time, and can adaptively corrupt up to $(1-\epsilon)n$ parties.*

Proof. All properties follow from arguments similar to the ones in the proof of Theorem 2. Lemma 18 covers the properties of termination, graded agreement and graded validity, while we can argue about the rest of the properties of indistinguishable randomness, correctness and soundness following the proofs of Lemmata 10, 11 and 12 respectively (with the respective changes of Δ'_{VGC} and the additional failure probability term of $\mathsf{negl}(\lambda_g)$ incurred from calling Π_{CV} instead of multicasting). □

Proof of Theorem 4. The modularity of the changes to the updated broadcast protocol allow us to argue about the proof of Theorem 4 without repeating the same arguments made for the proof of Theorem 3. Note that the gossiping part in Toss' does not affect the stages 0 through $g^*/2$ from the broadcast protocol. We briefly sketch all the distinctions between the two proofs.

Termination follows from the termination of the protocols and algorithms of Π'_{VGC}, and the following $g^*/2$ stages. Validity is argued as in Lemma 16; since it requires the correctness of the underlying Π'_{VGC} protocol, it inherits the additional probability of Lemma 19, and thus validity holds with probability $1-\mathsf{negl}(\lambda_g)-\mathsf{negl}(\kappa)$. Consistency is argued as in Lemma 17, again inheriting the additional probability and thus holding with probability $1-\mathsf{negl}(\lambda_g)-\delta-\mathsf{negl}(\kappa)$. □

Proof of Theorem 5. First, we describe how to amortize the communication cost of broadcast over multiple instances. Concretely, instead of feeding $(b||w_i)$ in broadcast instance $\mathsf{br_{id}}$ to the VRF, P_i can feed $(b||H(w_i||\mathsf{br_{id}}))$, which is also random, as long as w_i has length polynomial in the security parameter κ. Thus, the adversary can not predict the committee membership for a broadcast instance $\mathsf{br_{id}}'$ even after seeing the committee membership for all broadcast instances $\mathsf{br_{id}} < \mathsf{br_{id}}'$. Since the broadcast executions are independent with the exception of w_i, their composition is secure.

Therefore, in order to self-elect as a member in a committee, a party P_i now computes $(y_i, \pi_i) = \mathsf{VRF.Prove}_{\mathsf{SK}}(H(b||w_i||\mathsf{br_{id}}))$. As part of the verification of the membership of a different party P_j, P_i now computes in Verify: $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||w_j||\mathsf{br_{id}}), y_j, \pi_j)$. Define $\bar{\Pi}_{\mathsf{BC}}$ to be the subprotocol run between stages 0 and $g^*/2$ in the broadcast protocol Π_{BC} from Figure 3, with the

committee-related computations described above. Then, multiple secure broadcast instances are obtained by running the Π_{VGC} protocol (with any given instantiation) and then run $\bar{\Pi}_{\mathsf{BC}}(\mathsf{br_{id}})$, for $\mathsf{br_{id}} = 1, \ldots, \mathsf{br_{max}}$.

For item 1 in Theorem 5, we use one instance of Π'_{VGC} to generate random strings for $\mathsf{br_{id}} \in \{1, \ldots, n\}$. Using Π'_{VGC} where we instantiate the parallel GC via Protocol 5, and $\mathsf{br_{id}} \in \{1, \ldots, n\}$, yields an amortized communication complexity of broadcast of $\tilde{O}(n^2)$, and $\tilde{O}(\lambda_\delta)$ round complexity.

For item 2, using the online setup Toss from Π_{VGC} without gossiping (using the GC from Appendix A.3) and $\mathsf{br_{id}} \in \{1, \ldots, n^2\}$ yields an amortized communication complexity of broadcast of $\tilde{O}(n^2)$, and $O(\lambda_\delta)$ round complexity.

For item 3, we amortize the communication cost of parallel broadcast with techniques from Tsimos *et al.* [51]. Specifically, we use the Π'_{VGC} protocol with communication cost $\tilde{O}(n^3)$ to bootstrap randomness for n^2 broadcast instances. We then run n separate parallel broadcast instances where we apply the gossiping techniques of [51] on Protocol 3. The sole difference from $\bar{\Pi}_{\mathsf{BC}}$ is in the distribution round of stages 1 to $g^*/2 - 1$. Now parties, instead of multicasting their tuples of values, batches and certificates to all, propagate them via the gossiping dissemination protocol presented for the trusted setup broadcast in [51]. However, our Π'_{VGC} replaces the trusted setup. Thus, we obtain an amortized cost of $\tilde{O}(n)$ per broadcast instance, $\tilde{O}(\lambda_\delta)$ rounds, and without trusted setup.

Related to the composition of multiple broadcast instances, we make the following observations. The verifiable graded consensus protocol is run only once, before the "online" part of the broadcast instances start, and is independent of those. The sequentiality of Π_{VGC} (Π'_{VGC}) thus guarantees the unpredictability of the graded beacons that are used in any of the broadcast instances. The parties obtain the graded beacons for all other parties from Toss in Π_{VGC} (Toss' in Π'_{VGC}). Then in $\mathsf{Process}$, using the ROM, they evaluate only the VDF on the hash of their own beacon for each instance of broadcast, and store the other inputs for later verification. Even if the adversary obtains any of its inputs faster than the rest of the honest parties (as long as it still takes Δ_{VGC} (Δ'_{VGC}) with overwhelming probability), it cannot bias the beacons. The VRF properties will guarantee that the elections are verifiable and cannot be falsified. Moreover, we can add a time buffer between VGC and the stages of broadcast to ensure all honest parties finish evaluating their VDF with overwhelming probability before stage 0 of the first broadcast instance (in the parallel case, stage 0 of all instances starts at the same time). Security of the multiple broadcast instances is ensured as long as one honest committee member participates in stage 1 of that instance even if the adversary slows down some honest parties. Finally, note that there is no "gossiping composition" since gossiping is only used in Π_{VGC} and not in the rest of the broadcast protocols. □