Sublinear-round Broadcast without trusted setup against dishonest majority

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

[1] University of Maryland, College Park
{aandreea,tsimos}@umd.edu
[2] CISPA Helmholtz Center for Information Security
lossjulian@gmail.com
[3] Yale University
charalampos.papamanthou@yale.edu

Abstract. Byzantine broadcast is a central component of several cryptographic protocols, from secure multiparty computation to consensus mechanisms for blockchains. Many practical applications 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. This poses the question of achieving broadcast with efficient communication and round complexity against powerful adversarial models and without trusted setup assumptions.

In this paper, we answer this question positively. We present a Broadcast protocol that runs in rounds sublinear in n, the number of users, with asymptotic communication $\tilde{O}(n^3)$. Our protocol does not assume the existence of a trusted dealer who issues keys or common random strings. Instead, it is built upon a trustless verifiable delay function and the existence of random oracles in order to achieve a graded form of shared randomness between parties. This graded shared randomness acts as an untrusted online setup that can be used to securely run a committee-based protocol, similar to Chan et al. (PKC 2020). We also show that the graded shared randomness protocol we design can be used to seed multiple instances of Broadcast, which further amortizes the communication cost per instance to $\tilde{O}(n^2)$ over n instances or even to $\tilde{O}(n)$ per n 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 a fundamental problem in distributed computing that has been studied extensively for four decades. It is also of central importance to cryptographic protocols for multiparty computation and verifiable secret sharing to ensure a consistent view between the parties.

To keep the overhead of such applications as low as possible, a long line of works has focused on optimizing the efficiency of broadcast protocols. Most commonly, efficiency in broadcast is measured by 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 lower bounds of Dolev and Strong [15] and Dolev and Reischuk [14], deterministic protocols require $t + 1$ rounds and $O(n \cdot t)$ communication in order to tolerate t (statically) corrupted parties. This severely limits the practicality of such protocols as n grows large.

Fortunately, randomized protocols are known to bypass both of these lower bounds. Thus, an active area of research has studied robust and efficient broadcast protocols for the most challenging setting with a *dishonest majority* of $n/2 < t < n$ corrupted parties. While major progress has been made toward this goal, existing round-efficient protocols either achieve security only for a very small margin (less than 51% dishonest parties) or rely on trusted setup assumptions. On the other hand, deterministic protocols can easily be instantiated from plain public key setup and provide a suitably efficient layer of consensus if n does not grow too large.

In this work we revisit the question of building randomized *round-efficient broadcast protocols for the dishonest majority setting without trusted setup*. This suggests the need of a trustless form of efficient online setup secure against a dishonest majority that obtains randomness which can be used to replace the trusted setup. Concretely, we show a broadcast protocol that achieves the following: (i) runs in $o(n)$ rounds, (ii) requires $\tilde{O}(n^3)$ communication complexity, but can be amortized to n^2 communication complexity if it is run n many times, not necessarily in parallel, (iii) is secure against adaptive corruptions, and (iv) works in the *plain public key model*: parties generate their own keys and register them to a public bulletin board thereafter.

Our solution significantly improves over the state of the art. The celebrated Dolev-Strong protocol for broadcast [15] requires $O(t)$ rounds for $t < n$ malicious parties. Garay *et al.* [19], and Fitzi and Nielsen [17] achieved $o(n)$ rounds, but in a narrow case where $t/n - 1/2 \leq o(n)$. Recently, the elegant line of work [9,37,38] achieved $o(n)$ rounds for a widened gap of $t/n - 1/2 \geq \omega(1)$. While our solution relies on the random oracle heuristic and delay functions, we believe this to be a minor restriction. Indeed, many works have considered setup-free protocols in the random oracle model and proof of work or proof of time assumptions. An incomplete list includes the works of Andrychowicz and Dziembowski [1], and Garay *et al.* [20,21].

1.1 Contributions

We design protocols that are secure against an adaptive dishonest majority of $t \leq (1 - \epsilon)n$, for constant $\epsilon \in (0,\ 1)$, unless with negligible probability in the security parameter κ. Our protocols assume random oracles, a bulletin-board public key infrastructure, and the existence of delay functions, but no trusted setup in the sense of trusted dealer.

- **Verifiable Graded Consensus on Random Strings.** We provide a protocol for an online setup for graded shared randomness that has $O(\kappa)$ round complexity and $\tilde{O}(n^4)$ total communication complexity. We then show how to reduce the communication to $\tilde{O}(n^3)$ by using gossiping techniques, while keeping the round complexity to be polylogarithmic in n.
- **Broadcast.** We provide a stand-alone binary broadcast protocol using the verifiable graded consensus protocol, with $\tilde{O}(\kappa)$ round complexity and $\tilde{O}(n^3)$ total communication complexity. This compares favorably to the Dolev-Strong protocol [15], as it reduces the round complexity from $O(n)$ without introducing trusted dealers but requiring random oracles, and 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 over multiple broadcast instances and obtain an amortized broadcast communication complexity of $\tilde{O}(n^2)$ at $\tilde{O}(\kappa)$ round complexity, using n instances, or $O(\kappa)$, using n^2 instances. This compares favorably to the Chan *et al.* protocol [9], which achieves the same complexity but using a trusted setup in the standard model. Furthermore, we can use the techniques in Tsimos *et al.* [36] to amortize to $\tilde{O}(n^2)$ the total communication cost of *parallel broadcast* (n instances of broadcast run in parallel), if we run parallel broadcast n times.

Technical overview. First, we design a broadcast protocol which requires a sublinear number of rounds, but has no restriction on the communication complexity. Our construction builds on the Chan *et al.* broadcast protocol, which achieves a sublinear number of communication rounds by cleverly electing committees for each bit in an adaptively secure way, despite a dishonest majority. The membership in these committees is revealed as the rounds progress and batches (with round-dependent length) of valid votes are accumulated. The parties self-elect via Verifiable Random Functions (VRFs), which allow the others to check the validity of the membership claims, but are implemented by [9] with non-interactive commitment schemes and non-interactive zero knowledge proofs, which require a trusted dealer to generate common random strings and valid key pairs for each party.

To remove the strong assumption of trusted dealers, we need to ensure the secure use of VRFs with maliciously generated keys. To this end, parties are required to evaluate and prove their VRF on a random seed, revealed only after the parties post their public keys on the bulletin-PKI, such that malicious parties cannot adaptively choose a secret key that passes a validation predicate on their posted public key and also manage to bias the VRF output.

Therefore, parties will have to share and "agree" on unpredictable and unbiased random strings, which should act as random beacons. Importantly, parties cannot just share and use a local random seed as their output random strings; if they did, dishonest parties could observe and decide their inputs accordingly, biasing the final result. Instead, we assume the existence of delay functions, which ensure that even with access to parallel computation, an adversary cannot evaluate a function faster than a fixed time difficulty parameter; and so it does not have enough time to conveniently update its inputs. Therefore, parties obtain

fresh randomness via verifiable delayed functions (VDFs). We use Wesolowski's VDF [39] which does not require a trusted setup.

The four main challenges preventing us from using efficient and/or permissionless random beacon generators such as [35,33,12], even ones based on VDFs or time-lock puzzles, are the following: (i) lack of trusted setup, which limits the cryptographic tools that can be used, (ii) lack of broadcast channels, which does not guarantee agreement on the messages communicated between parties, (iii) dishonest majority, and (iv) requirement of sublinear number of rounds, which rules out each party having to forcibly evaluate the other parties' VDFs.

Parties cannot reach agreement in the sense of Byzantine agreement (which would be a circular problem), but rather in the sense of graded agreement, where parties obtain a grade associated to the output message. The grades indicate the confidence honest parties have that they all hold the same message. Notably, honest parties can disagree on their messages if they have the lowest two grades.

We bootstrap a gradecast protocol—which ensures graded agreement on a sender's message—to a parallel moderated gradecast protocol, where each party act as moderator for all the seeds it received via gradecast. We show how to aggregate both the seeds and the grades for each moderator, such that the graded agreement properties still hold, with the extra property that the aggregated seeds are unpredictable to the adversary at the time it would have to start evaluating the VDF in order to bias the output. At the end, each party will have a VDF evaluation (guaranteed to be unpredictable despite a dishonest majority) and proof, as well as aggregated seeds and grades for the other parties. We call this protocol a verifiable graded consensus on random strings.

These shared unpredictable random strings are used to augment the bulletin-PKI in a way that allows for securely implementing a committee election procedure. With this augmented setup in place, parties participate in the committee-based Chan *et al.* protocol, where they collect votes for validly signed bits by the sender across a sublinear number of rounds. In our case, a vote consists of a proof that the VRF output is lower than a prespecified bound, as well as a proof that the input on which the VRF was evaluated is truly the VDF evaluation of the relevant output of the verifiable graded consensus protocol. The design is such that parties can check votes without evaluating the VDF locally. In each round, parties accept batches of votes based on the number of valid votes contained, but also based on the confidence they have in the voter, given by their grade. Therefore, although not necessarily intuitive, the total number of rounds dictates the maximum grade we can have in the verifiable graded consensus protocol and the confidence threshold accepted depends on the current round. This is necessary because of the potential difference of one in the grades of honest parties, that can cascade to the lowest grade, where parties can disagree on their strings.

Put together, this builds up to a broadcast protocol with an online setup that emulates random beacons, that has a sublinear round complexity. However, the verifiable graded consensus protocol requires running a quadratic number of gradecasts in parallel, which brings the total communication complexity to quartic in the number of parties.

We reduce the communication complexity for the verifiable graded consensus (and thus, for an instance of broadcast as well) to cubic via gossiping instead of all-to-all transmissions. However, despite previous work on utilizing gossiping for e.g., parallel broadcast [36], adapting the results for our parallel moderated gradecast on non-binary messages is far from trivial. Parties have to drop conflicting 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. The number of rounds will increase slightly but remain polylogarithmic in the number of parties.

Finally, the online setup can be utilized to obtain random seeds for multiple broadcast instances via a random oracle. Therefore, we can amortize the cost of the verifiable graded consensus over multiple instances of broadcast to obtain the communication cost for a broadcast instance to be quadratic (when amortized over n instances) or linear (when amortized over n runs of parallel broadcast, so n^2 instances in total). Verifiable secret sharing or secure multiparty computation involve at least n uses of broadcast, and applications requiring a linear number of rounds are a good use case for the latter result.

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

1.2 Related work

We start with the related literature for broadcast protocols in the setup-free case apart from a bulletin-PKI. Dolev and Strong [15] 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.* [26,25] and Boyle *et al.* [7] proposed protocols with reduced communication complexity of $\tilde{O}(n^{3/2})$ but only for honest super majority $t < n/3$. Momose and Ren [29] 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.

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.* [19] achieved $O((2t-n)^2)$ rounds, and Fitzi and Nielsen [17] achieved $O(2t - n)$ rounds, against a strongly adaptive adversary.

Chan *et al.* [9] 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.* [38] 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. The solution requires building a trust graph which allows honest parties to identify the corrupted parties.

Wan *et al.* [37] tolerates stronger adversaries that can also erase messages. In their solution, 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 will have been solved and their solutions 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 why this solution cannot be used in our case of emulating random beacons, where an adversary can bias the result by observing the intermediate opened puzzles and deciding to not allow some corrupt puzzles to be opened.

Hou *et al.* [24] describe a blockchain that tolerates dishonest majority, loosely based on the Chan *et al.* broadcast protocol [9], proof of stake, proof of work in the random oracle model (ROM).

Time-lock puzzles (TLP) [32], timed commitments [6] and verifiable delay functions [4] 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 [4,30,39,13]. VDFs are currently used or intended to be used in blockchain applications such as Ethereum and Chia [8,11].

In the context of timing assumptions and broadcast, Das *et al.* [12] describe a Byzantine Agreement protocol with VDF in ROM, which achieves an expected constant round complexity in a setting without trusted or authenticated setup, tolerating fewer than $n/2$ adaptive corruptions. Although also based on graded agreement and VDFs, their construction differs conceptually from ours. Das *et al.* [12] generate a graded PKI with only two grades, then use VDFs both in this construction and in order 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 in [27]. Constructing randomness from VDF/TLP with transparent setup in ROM, but assuming broadcast, is also addressed by Bhat *et al.* [3], who tolerate $t < n$ only if the adversary is covert, and by Thyagarajan *et al.* [35] who tolerate $t < n$. Freitag *et al.* [18] propose a fair coin flipping protocol that assumes a public bulletin board and a partially trusted setup called all-but-one

model (non-interactive but where parties need to solve all TLPs) or with trusted setup and ROM (interactive but publicly verifiable).

2 Preliminaries

2.1 Model

Network. We consider n parties $P_1, \ldots, P_n$ that have access to a bulletin public key infrastructure (PKI). Every party generates a pair of keys (for e.g., signing, verifiable random function, verifiable delay 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: parties start executing round r at time $(r-1) \cdot \Delta_r$. We assume that parties perform *atomic send operations*, i.e., they can send a message to multiple parties simultaneously in such a way that the adversary cannot corrupt them in between individual sends.

Security parameters. We denote the computational security parameter κ and the statistical security parameter by λ. We assume all cryptographic hash functions are modeled as *random oracles*. This means that for any input x in the domain, a hash function H returns $H(x)$ if it was queried before on x and otherwise returns a random value from the codomain. We assume that the hash function output length is $O(\kappa)$, for the security parameter κ. Moreover, all signatures, verifiable delay function outputs and verifiable random function outputs are also of size $O(\kappa)$. We also use a failure probability $\delta \in (0,\ 1)$ that is negligible in the statistical security parameter λ but independent of n, i.e., $\log(1/\delta) = \mathsf{polylog}(\lambda)$.

For simplicity, since the computational security parameter is generally larger than the statistical security parameter, we use κ for both. We assume a regime where the $\kappa \leq \mathsf{polylog}(n)$, such that $\kappa \cdot n \leq \tilde{O}(n)$.

Threat model. We consider a *Byzantine fault* model, in which some fraction of the parties may be corrupted by an adversary $f \leq (1-\epsilon)n$, for $\epsilon \in (0,\ 1)$. The adversary controls the local computations, 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*. In the rest of the paper, whenever we make a statement about a honest party's actions or views, we refer to the specific moment when the party is honest if it is not clear from the context. For example, the validity of broadcast (Definition 1) refers to the sender remaining honest until the end of the protocol.

The adversary cannot perform *after the fact removals*: the adversary 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 has access to a probabilistic polynomial-time machine. Looking ahead, the adversary is Δ-limited, i.e., evaluating a VDF with difficulty parameter Δ on the adversary's machine takes parallel time Δ with $\mathsf{poly}(\Delta, \kappa)$ processors (see Def. 9). Honest users might have weaker machines.

2.2 Definitions and primitives

Broadcast. Introduced in the seminal work by Lamport *et al.* [34], 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 v^*, is a broadcast protocol tolerating t malicious parties if the following notions hold:*

(Validity): If P^* is honest, then every honest party outputs v^*.
(Consistency): Every honest party outputs the same value v.

Gradecast and Moderated Gradecast. Gradecast, introduced by Feldman and Micali in [16] and generalized for an arbitrary grade in [19], 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 v^*, is a g^*-gradecast protocol tolerating t malicious parties if the following notions hold:*

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

We also define a *Moderated Gradecast* protocol, where a moderator P_M re-gradecasts the value it received from P^*, the sender in gradecast. The goal is for the honest parties to use the two pieces of information, coming from the two gradecasts with different senders, to obtain outputs on which they disagree by a "small amount" 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 v^*, and a moderator $P_M \in \{P_1, \ldots, P_n\}$ moderates this value, is a g^*-moderated gradecast protocol tolerating t malicious parties if the following notions hold:*

(Termination): Each honest party outputs (v, g) where $g \in \{0, \ldots, g^*\}$ and terminates.
(Validity): If P^* is honest and moderator P_M is also honest, then every honest party outputs (v^*, g^*).
(M-Validity): If moderator P_M is honest, then every honest party outputs (v, g) as their value from moderator P_M, where $g \in \{g^* - 1, g^*\}$.

(Soundness): Let P_i, P_j be two honest parties outputting (m_i, g_i) and (m_j, g_j), respectively. If $g_i \geq 2$, then $m_i = m_j$ and $|g_i - g_j| \leq 1$. If $g_i = 1$, then either $m_i = m_j$ or $g_j = 0$.

We are interested in the parallel version of moderated gradecast, where a party acts as a moderator for multiple initial senders, so its moderator grade is determined by all gradecasts, and in the doubly-parallel version of moderated gradecast, where each party acts as an initial sender and as a moderator for all other senders.

Verifiable Random Functions. Verifiable Random Functions (VRFs), introduced by Micali *et al.* [28] 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.

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. First, we require an efficient algorithm that checks a predicate $\mathsf{VRF.Validate}$ on the public key in order to verify that the public key corresponds to an admissible secret key. Second, we require another property, which we call *extended pseudorandomness*, which is different from *pseudorandomness*, where the adversary only chooses the input but not the key pair and thus can't evaluate the VRF.

Chen and Micali [10] and Gilad et al. [22] 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. [23] discuss this issue and propose VRF constructions based on RSA or elliptic curves (in the random oracle model) that have a validation predicate for the public key as in our definition.

Definition 4. *A function family $F_{(\cdot)}(\cdot) : \{0,1\}^{l(\kappa)} \to \{0,1\}^{m(\kappa)}$ is a family of verifiable random functions (VRFs) if there exists a probabilistic polynomial time algorithm* $\mathsf{VRF.Gen}$ *and deterministic algorithms* $\mathsf{VRF.Prove}$, $\mathsf{VRF.Verify}$, $\mathsf{VRF.Validate}$ *such that:*

- $\mathsf{VRF.Gen}(1^\kappa)$ *outputs a pair of keys* $(\mathsf{PK}, \mathsf{SK})$*;*
- $\mathsf{VRF.Prove}_{\mathsf{SK}}(x)$ *outputs a pair* $(F_{\mathsf{SK}}(x), \pi_{\mathsf{SK}}(x))$*, where* $F_{\mathsf{SK}}(x)$ *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* $y = F_{\mathsf{SK}}(x)$ *using* 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, Provability, Pseudorandomness *and* Extended Pseudorandomness.

The definitions of Uniqueness, Provability, Pseudorandomness and Extended Pseudorandomness are standard (with the exception of the latter) and are given in Appendix A.

In our constructions, we can either instantiate the VRF via RO and unique signatures, e.g., BLS [5] 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 in the ROM from [23] with the validity predicate described there.

Verifiable Delay Functions. Verifiable Delay Functions (VDFs), defined by [4], 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.

Definition 5. *A Verifiable Delay Function (VDF)* $V = (\mathsf{VDF.Setup}, \mathsf{VDF.Eval}, \mathsf{VDF.Verify})$ *is a triplet of algorithms:*

- $\mathsf{VDF.Setup}(1^\kappa, \Delta) \to pp = (\mathsf{EK}, \mathsf{VK})$ *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*;*
- $\mathsf{VDF.Eval}_{\mathsf{EK}}(x) \to (y, \pi)$ *outputs* $y \in \mathcal{Y}$*, the evaluation on input* $x \in \{0,1\}^{l(\kappa)}$ *with evaluation key* EK *and* π*, the proof of correctness of this evaluation;*
- $\mathsf{VDF.Verify}_{\mathsf{VK}}(x, y, \pi, \Delta)$ *outputs 1 if* y *is the correct output of input* x *associated to verification key* VK *and difficulty* Δ*, possibly using proof* π*, and 0 otherwise.*

A VDF should satisfy Correctness *(Definition 6),* Soundness *(Definition 7) and* Sequentiality *(Definition 9).*

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. Algorithm $\mathsf{VDF.Verify}$ must run in total time polynomial in $\log \Delta$ and κ.

Definition 6. *[Correctness] A VDF* V *is* correct *if for all* κ*,* Δ*,* $pp \leftarrow \mathsf{VDF.Setup}(1^\kappa, \Delta)$ *and all* $x \in \{0,1\}^{l(\kappa)}$*, if* $(y, \pi) \leftarrow \mathsf{VDF.Eval}_{\mathsf{EK}}(x)$ *then* $\mathsf{VDF.Verify}_{\mathsf{VK}}(x, y, \pi, \Delta) = 1$*.*

Definition 7. *[Soundness] A VDF* V *is* sound *if for all algorithms* $\mathcal{A}$ *that run in time* $O(poly(\Delta, \kappa))$*, it holds that:*

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

We adapt the sequentiality definition from Boneh *et al.* [4] and Wesolowski [39] for an honestly selected input concatenated with an input chosen by the adversary. The definition states that no adversary is able to compute an output for $\mathsf{VDF.Eval}$ 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.

Definition 8. *[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* $pp \leftarrow \mathsf{VDF.Setup}(\kappa, \Delta)$.
2. $\mathcal{A}$ *processes the public parameters and obtains* $L \leftarrow \mathcal{A}_0(\kappa, pp, \Delta)$.
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* $y_A \leftarrow \mathcal{A}_1(L, pp, x_1||x_2)$.

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

Definition 9. *[*(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 with probability greater than* $negl(\kappa)$*.*

Definition 9 implies the *unpredictability* of the VDF output, but not indistinguishability from random [4]. But in the ROM, any unpredictable string can be used to extract an unpredictable κ-bit uniform random string.

Wesolowsky [39] gives a construction for public coin VDF, which we adopt in our protocols. The underlying group of the VDF is a class group 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 a hash function $H_G : \{0,\ 1\}^* \to G$ modeled as a random oracle. The setup is transparent, and honest parties can check whether VK, EK are not valid.

Furthermore, the VDF in [39] 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 $\sigma(\Delta)$. The proof is given in Appendix D.

Lemma 1. *The VDF proposed in [39] achieves* σ*-Sequentiality in the ROM for any* p *and for any* $\sigma(\Delta) = (1-\xi)\cdot\Delta$.

Verifiable Graded Randomness. Our goal is to generate strings that act as random beacons [31]: they satisfy unpredictability (and randomness in ROM), bias resistance, termination and verifiability by the other parties in the system. We introduce and define the following protocol.

Definition 10. *A protocol executed by parties* $P_1, \ldots, P_n$*, where each honest party begins holding a random string* x_i *and PKI information, is a* verifiable graded consensus on random strings *composed of algorithms* Gen*,* $Process$*,* $Verify$ *and protocol* $Toss$*:*

- $Gen(1^\kappa, \Delta)$ *outputs public parameters* pp*;*
- $Toss(pp)$ *and outputs to each party* P_i n *pairs* $(x_i^{(j)}, g_i^{(j)})$*;*
- $Process(pp, x)$ *outputs* (y, π)*;*

- $\mathsf{Verify}(\mathsf{pp}, x, y, \pi)$ *outputs 1 if* $(y, \pi) = \mathsf{Process}(\mathsf{pp}, x)$ *and 0 otherwise.*

A verifiable graded consensus on random strings protocol should satisfy the following notions whenever there are at most t corrupted parties.

> *(σ-Indistinguishability) For $\sigma(\Delta) \leq \sigma'(\Delta)$, where $\mathsf{Process}$ 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-Indistinguishability game with probability more than $\mathsf{negl}(\kappa)$.*
> *(Soundness, M-Validity) As in Definition 3, where every party acts as a moderator.*
> *(Termination) An honest party P_i outputs $(x_i^{(j)}, g_i^{(j)})$ for every $j \neq i$ after Toss and (y_i, π_i) after $\mathsf{Process}$.*
> *(Verifiability) A honest party that output $(x_i^{(j)}, \cdot)$ after Toss for a party j and receives (y_j, π_j) as output of $\mathsf{Process}$ will not accept y_j as P_j's beacon unless $\mathsf{Verify}(\mathsf{pp}, x_i^{(j)}, y_j, \pi_i) = 1$.*

The Δ-indistinguishability game captures both the unpredictability and unbiasedness notions, inspired by [2,35], and is based on the σ-sequentiality definition. The Δ-indistinguishability game is provided in Appendix A.

In the ROM, the outputs $H(y_i)$ are random, not just unpredictable.

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 $\mathsf{ver}_i(s, m) = 1$.

3 Emulating a common random string

Consider that each party has an associated *slot*, which contains a random seed. Based on these values, each honest party needs to create and "agree" on aggregate fresh random strings, via a protocol we call *Moderated Gradecast*. Each random string is input to a VDF, such that the VDF evaluations produce an unpredictable and unbiased random string that serves as a "common random string for parties". However, parties might not all hold the same common value, because the adversary can send different seeds to the honest parties. Nevertheless, the grades that honest parties hold for the final strings quantify how much confidence they have that their strings for a party coincide. In particular, grades greater than 2 certify that honest parties have the same VDF input and hence, VDF evaluation.

In our protocols, we use the $\mathsf{Gradecast}$ protocol from [19], provided in Appendix B. Let g^* be the maximum grade used in $\mathsf{Gradecast}$. The $\mathsf{Gradecast}$ protocol takes $2g^* + 1$ rounds and has $O(g^* \cdot (\kappa + \ell) \cdot n^2)$ communication complexity.

We start by describing the parallel Moderated Gradecast protocol, introduced in Definition 3, which ensures the pairwise graded agreement of honest parties on random strings. Then, we describe how to utilize the VDFs to obtain graded agreement on verifiable unpredictable values, introduced in Definition 10.

3.1 Moderated Gradecast

A single run of parallel gradecast is sufficient for parties to have graded agreement on the seed from each party. Concretely, each party P_j has for each sender P_i a message and a grade $(m_{i,j}, g_{i,j})$. However, it is not sufficient to have both unpredictability (after evaluating the VDF), soundness and high grade for honest parties. For instance, P_j cannot set the final string for P_i to be $m_{i,j}$ because it would not be random if P_i is malicious. On the other hand, if P_j sets as its random string $\|_{i\in[n],g_{i,j}\geq 1} m_{i,j}$, 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 messages itself gradecasts, 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 in Step 1 will not be penalized by more than 1 in its final grade. Therefore, malicious parties cannot arbitrarily modify the final grades of honest parties.

We construct the moderated gradecast Mod-Gradecast protocol based on several instances of Gradecast. In Figure 1, we introduce directly the parallel Mod-Gradecast protocol for further use.

It is trivial to see that termination holds for Protocol 1; each honest party will generate a value and an accompanying grade for every moderator, by construction. Validity also easily follows from the composition of gradecast protocols that achieve validity themselves. To prove the remaining properties, we make the following observation, proved in Appendix D.

Observation 1 *Let* a_1, b_1, a_2, b_2, g *s.t.* $|a_1 - a_2| \leq 1$, $|b_1 - b_2| \leq 1$ *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,* $|G_1 - G_2| \leq 1$.

The following results hold for one instance of Mod-Gradecast (one sender and one moderator) from the parallel version presented in Figure 1, against an adaptive adversary controlling $t \leq (1 - \epsilon)n$ parties.

Lemma 2. Mod-Gradecast *satisfies* M-validity.

Proof. Let P_s be the honest moderator and let P_j be an honest party. P_s will gradecast every value it moderates correctly, thus $g_{i,j}^{(s)} = g^*$ for all i. If P_i is also honest, then $m_{i,j} = m_{i,j}^{(s)}$, so $G_{i,j}^{(s)} = g^*$. However, if P_i is dishonest, then it could

Parallel Mod-Gradecast(g^*)

Step 1: Each party gradecasts their input for their slot.

1. Each party P_i calls $\mathsf{Gradecast}(m_i, g^*)$.
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 $\mathsf{Gradecast}(m_i^{(s)}, g^*)$, for all slots $i \neq s$.
3. Denote the n pairs per moderator P_s held at party P_j as $(m_{s,j}, g_{s,j}), \{(m_{i,j}^{(s)}, g_{i,j}^{(s)}) : i \in [n] \setminus \{s\}\}$.

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 slot i, 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. Parallel Moderated Gradecast protocol.

be $m_{i,j} \neq m_{i,j}^{(s)}$. 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 $\mathsf{Gradecast}$ we have $g_{i,j} \leq 1$. Therefore, $G_{i,j}^{(s)} = \min\{g_{i,j}^{(s)}, g^* - g_{i,j}\} \geq g^* - 1$ and so $G_j^{(s)} \geq g^* - 1$. □

Lemma 3. Mod-Gradecast *satisfies* soundness.

Proof. For any moderator P_s and any slot 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. $m_{i,j}^{(s)} = m_{i,j}$ & $m_{i,k}^{(s)} = m_{i,k}$. 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 $|G_{i,j}^{(s)} - G_{i,k}^{(s)}| \leq 1$.

Case 2. $m_{i,j}^{(s)} = m_{i,j}$ & $m_{i,k}^{(s)} \neq m_{i,k}$. 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. Thus,

- if $g_{i,k}^{(s)} \leq 1$, then $G_{i,k}^{(s)} \leq 1$ and also $g_{i,j}^{(s)} = G_{i,j}^{(s)} \leq 1$. So $|G_{i,j}^{(s)} - G_{i,k}^{(s)}| \leq 1$.
- if $g_{i,k} \leq 1$, then either $g_{i,k}^{(s)} = g^*$, in which case $G_{i,k}^{(s)} \in \{g^* - 1, g^*\}$ and $G_{i,j}^{(s)} \in \{g^* - 1, g^*\}$, or $g_{i,k}^{(s)} \leq g^* - 1$, in which case $G_{i,k}^{(s)} = g_{i,k}^{(s)}$.

Case 3. $m_{i,j}^{(s)} \neq m_{i,j}$ & $m_{i,k}^{(s)} \neq m_{i,k}$. 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 Observation 1 to get the result. □

The Parallel Mod-Gradecast in Figure 1 satisfies the following property, with respect to the final grade each honest party associates with each moderator.

Lemma 4. *Let P_j, P_k be two honest parties. 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 $G_k^{(s)} = 0$.*

Proof. The result follows from Lemma 3. □

Communication and Round Complexity. Mod-Gradecast takes $4g^* + 2$ rounds, determined by running n-parallel instances of Gradecast followed by n^2-parallel instances of Gradecast. The total communication complexity for n parties with inputs of length ℓ and signature size κ is $n^2 \cdot \mathsf{CC}_{\mathsf{Gradecast}}(\ell, \kappa, g^*)$, 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, which limits the cryptographic tools that can be used, (ii) lack of broadcast channels, which does not guarantee agreement on the messages communicated between parties, (iii) dishonest majority, and (iv) requirement of sublinear number of rounds, which rules out each party having to forcibly evaluate the other parties VDFs.

To address point (i), we employ Wesolowski's VDF construction [39] based on class group of imaginary quadratic fields, which has transparent setup and does not have trapdoors. Point (iii) is addressed by the Mod-Gradecast construction in the previous subsection which is secure against a dishonest majority. Points (ii) and (iv) above limit the use of the homomorphic properties of the VDF evaluation. First, we cannot have parties gradecasting $(m_i, \mathsf{VDF.Eval}(m_i))$, 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: e.g., if it is even, do not submit anything, otherwise submit a valid tuple $(m, \mathsf{VDF.Eval}(m))$ with even output, which will cause the final output to be even. Using timed commitments solves this issue. However, the lack of broadcast causes in the worst case each honest party to have to open the timed commitment of every malicious party, which is not sublinear time.

We aim for each party to only have to evaluate one 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, will be the concatenation of the parties' seeds, as in the random beacon construction described in [27]. Strings that have grade greater than 1 have the guarantee that they contain seeds of honest parties, therefore the concatenation will be random. Moreover, 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 (y_i, π_i) will be used in a setting where it is multicast by the computing party P_i. A 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 $\mathsf{VDF.Verify}(M_j^{(i)}, y_i, \pi_j, \cdot)$ if $G_j^{(i)} \geq 1$.

To summarize, we employ Wesolowski's VDF construction in the parallel Mod-Gradecast protocol in Figure 1 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 twice the time required to run Gradecast (in terms of the length of synchronous rounds).

Parameters and technical results. 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 Mod-Gradecast protocol. After the end of Step 1 of Protocol 1, all honest parties P_j are guaranteed to have obtained their own $M_j^{(j)} = m_j^{(j)} := \|_{i \in [n], g_{i,j} \geq 1} m_{i,j}$. 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 rely their messages from the beginning.

Let $\Delta_G := (2g^* + 1) \cdot \Delta_r$ denote the duration of the gradecast protocol, where Δ_r is the duration of a round. The Parallel Mod-Gradecast 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 Mod-Gradecast protocol. The adversary can do damage even in the last round of Mod-Gradecast, i.e., change the values of the honest parties' strings for a malicious moderator. While a delay of exactly $2 \cdot \Delta_G$ is sufficient to ensure Protocol 1 ends before the adversary can complete a VDF evaluation on values obtained in round 1 of Protocol 1, because of slight speed-ups that the adversary's machine could have, we prefer to set the delay to take this speed-up in consideration.

Therefore, we set the puzzle difficulty parameter Δ_{VDF} of the VDF to be $\Delta_{\mathsf{VDF}} = 2 \cdot \Delta_G / (1 - \xi)$. Then, the VDF will be σ-sequential, for $\sigma(\Delta_{\mathsf{VDF}}) = (1 - \xi) \cdot \Delta_{\mathsf{VDF}} = 2 \cdot \Delta_G$.

At the end of Step 1, each party computes $\mathsf{VDF.Eval}_{\mathsf{EK}}(H(m_j^{(j)}), \Delta_{\mathsf{VDF}})$. Importantly, security holds only when we do not use trapdoor-VDFs, otherwise malicious parties would have an advantage in biasing the output.

We now instantiate the algorithms and protocols of VGC in Section 2.2. Gen will be the transparent setup required for VDF, Toss will be the Mod-Gradecast protocol with the computation of the local strings, Process will be the inherited VDF.Eval on the hash of the output and Verify will be the inherited VDF.Verify.

We give the following results for an adaptive Δ_{VDF}-limited adversary who runs in at most $(1 - \xi) \cdot \Delta_{\mathsf{VDF}} = 2 \cdot \Delta_G$ parallel time, and can adaptively corrupt up to $(1 - \epsilon)n$ parties. The proofs are provided in Appendix D.

Lemma 5. *Π_{VGC} is σ-indistinguishable, for $\sigma(\Delta_{VDF}) = (1 - \xi) \cdot \Delta_{VDF}$.*

Lemma 6. *Π_{VGC} achieves soundness and M-validity.*

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

1. $\mathsf{Gen}(1^\kappa, \Delta_{\mathsf{VDF}})$: Each party P_i calls $(\mathsf{EK}_i, \mathsf{VK}_i) \leftarrow \mathsf{VDF.Setup}(1^\kappa, \Delta_{\mathsf{VDF}})$, posts the keys on the bulletin-PKI and sets $\mathsf{pp} = (\Delta_{\mathsf{VDF}}, \mathsf{EK}_i, \mathsf{VK}_i, PKI)$.
2. $\mathsf{Toss}(\mathsf{pp})$: Each party P_i samples a random input seed m_i, then calls the Parallel Mod-Gradecast and 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 evaluates its own random string as $(y_i, \pi_i) = \mathsf{VDF.Eval}_{\mathsf{EK}_i}(H(M_i^{(i)}), \Delta_{\mathsf{VDF}})$ (they can start computing this value before the Parallel Mod-Gradecast terminated).
4. $\mathsf{Verify}(\mathsf{pp}, x, y, \pi)$: On receiving (y_j, π_j) from P_j, party P_i outputs $\mathsf{VDF.Verify}_{\mathsf{VK}_j}(M_i^{(j)}, y_j, \pi_j, \Delta_{\mathsf{VDF}})$.

Fig. 2. Random strings generating protocol

Lemma 7. *Π_{VGC} achieves termination and verifiability.*

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 (evaluating the VDF) which takes $\Delta_{\mathsf{VDF}} = (4g^*+2)/(1-\xi)$, for very small $\xi > 0$. This means that the total round complexity for Π_{VGC} is slightly over $6g^*+3$, i.e., $6g^*+3+O(1)$. The total communication complexity is the same as for Protocol 1, $O(g^* \cdot (\ell + \kappa) \cdot n^4)$.

4 Broadcast with ideal graded mining functionality

The goal of the verifiable graded consensus was to emulate a common random string given by a trusted setup, which should aid an adaptively-secure committee election against a dishonest majority. The key difference from existing approaches is that, since we do not really obtain the equivalent of a trusted setup, the only guarantees that parties have are related to the grades of these distributed random strings.

4.1 Overview

Let us review the Chan *et al.* broadcast protocol [9]. A party can check if it is in the committee for the bit b via a mining functionality, which also allows all other parties to verify the correctness of this statement. This mining functionality is instantiated via a verifiable random function, allowing 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 adaptive adversaries. 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 is it possible to achieve consistency with overwhelming probability even if the number of rounds is constant and the committee size is also constant, if the corrupted fraction is constant.

In our case, the main difference with respect to the ideal mining functionality is that the verification performed by other parties on the membership of one party will return a binary answer *and* a grade in $\{0, \ldots, g^*\}$. The 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 modify the definition of valid batches of signatures for given rounds and relate the grade with the round number.

Specifically, we set the maximum grade g^* that can be returned by the mining functionality to be equal to the number of rounds the Chan *et al.* protocol requires. We also say that a batch of r signatures will be valid only if there are r signatures from parties on which the verification call on the mining functionality returns 1 and a sufficiently large grade, i.e., 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 grades have to be at least $g^* - \rho$, but 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.

Recall that the mining functionality does not necessarily return the same grade to all parties, but they might differ by one. This means that an honest party P_i might accept a batch with r signatures, i.e., all grades for the signers that P_i received from the verify call to the mining functionality are at least $g^* - 2r + 1$, but another honest party P_j received from the verify call a lower grade $g^* - 2r$ for one of the same signers. Therefore, 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 $g^* - 2r - 1$, in order to be picked up in the distribution round of stage $r + 1$.

Graded mining functionality. Define $\mathcal{F}_{\mathsf{mine}}^{g^*}$ to be the idealized graded mining functionality, parameterized by a probability p_{mine} and a maximum grade g^*. Parties can call $\mathcal{F}_{\mathsf{mine}}^{g^*}$ to vote on b, $\mathcal{F}_{\mathsf{mine}}^{g^*}.\mathsf{Mine}(b)$, or to verify another vote on b, $\mathcal{F}_{\mathsf{mine}}^{g^*}.\mathsf{Verify}(b, \cdot)$. Mine and Verify take as input the bit b in order to help elect different committees for 0 and for 1. This ensures that an adaptive adversary cannot corrupt a party after seeing its vote and force it to give a valid vote for the other value.

Once $\mathcal{F}_{\mathsf{mine}}^{g^*}$ was called by a party, it will always return the same output to that party. The $\mathcal{F}_{\mathsf{mine}}^{g^*}$ functionality is given in Figure 3.

Functionality: $\mathcal{F}_{\mathsf{mine}}^{g^*}$

$\mathcal{F}_{\mathsf{mine}}$ is parameterized by parties $P_1, \ldots, P_n$, "mining" probability p_{mine} and maximum grade g^*. Let $s \in [n]$ and $b \in \{0, 1\}$. Let $call^b$ be vectors of n entries initialized with -1.

On input (Mine, b) from party i:
- If $call_i^b = -1$ flip a coin with probability p_{mine} and set $call_i^b = 1$ if the output is 1, otherwise set $call_i^b = \perp$;
- Else output $call_i^b$.

On input (Verify, b, j) from party i output 1 and a grade $G_i^{(j)} \in \{0, \ldots, g^*\}$ if $call_j^b = 1$ and 0 otherwise.

Fig. 3. Mining functionality $\mathcal{F}_{\mathsf{mine}}^{g^*}$ parameterized by a maximum grade g^*.

We select p_{mine} depending on g^* to be a constant-sized quantity such that the following two rules hold. These rules correspond to (Lemmata 10 and 11) for a specific selection of p_{mine} in Section 5.
Rule 1. **Honest Lower Bound:** At least one honest party self-elects in a bit committee with overwhelming probability in the security parameter.
Rule 2. **Dishonest Upper Bound:** Fewer than $g^*/2$ dishonest parties can self-elect in a committee with probability overwhelming in the security parameter.

Each party P_i maintains a set $\mathsf{Extracted}_i$, initialized to the empty set. A signature of party P_i on a bit b is denoted as $\mathsf{sig}_i(b)$. A party P_i will add a bit b to their $\mathsf{Extracted}_i$ set if it receives a valid batch on b, as described below.

Valid batches. The protocol consists of stages, where each stage is composed of two rounds. We denote the stage number by r for $r \in \{1, \ldots, g^*/2\}$, round 1 of stage r by $2r - 1$, and round 2 of stage r by $2r$. We prefer this notation because we want to have sets of r signatures in every stage r. To address the difference in grades in the two rounds, we will define $(r, 1)$-batches and $(r, 2)$-batches.

We say that a batch rb in stage r for a bit b is a valid $(r, 1)$-*batch* for party P_i if: (i) there are at least r distinct signatures, (ii) one of the signatures is from P_s and (iii) the rest of the signatures are valid signatures from parties P_j for which $\mathcal{F}_{\mathsf{mine}}^{g^*}.\mathsf{Verify}(b, j)$ returns $(1, G_i^{(j)})$ at stage r, where grade $G_i^{(j)} \geq g^* - 2r + 1$. Similarly, a $(r, 2)$-*batch* is a valid batch with at least r distinct signatures coming from parties in the b-committee, each with grade $G_i^{(j)} \geq g^* - 2r$.

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

We describe the Broadcast protocol when given the idealized graded $\mathcal{F}_{\mathsf{mine}}^{g^*}$ functionality in Figure 4. It is clear that Protocol 4 has round complexity g^*, where g^* is the maximum grade returned by $\mathcal{F}_{\mathsf{mine}}^{g^*}$.

Broadcast in the $\mathcal{F}_{\mathsf{mine}}^{g^*}$-hybrid world

Stage 0:

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

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

1. (Round $2r - 1$): Each party P_i accepts a message $b \notin \mathsf{Extracted}_i$ only if it is accompanied by a valid $(r, 1)$-batch rb: $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$. P_i then propagates (b, rb) 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 $(r, 2)$-batch. For each such b, P_i checks whether it has called $\mathcal{F}_{\mathsf{mine}}^{g^*}(b, i)$ before and calls it if not.
   If $\mathcal{F}_{\mathsf{mine}}^{g^*}(b, i) = 1$, then P_i sets $\mathsf{Extracted}_i \leftarrow \mathsf{Extracted}_i \cup \{b\}$, constructs a $(r + 1, 1)$-batch $rb' := rb \| \mathsf{sig}_i(b)$ and sends (b, rb') to all parties.

Stage $r = g^*/2$:

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

Fig. 4. Broadcast protocol in the $\mathcal{F}_{\mathsf{mine}}^{g^*}$-hybrid world for designated sender P_s and parties $P_1, \ldots, P_n$, run for a fixed number g^* of rounds.

We will prove that the protocol achieves validity and consistency in the $\mathcal{F}_{\mathsf{mine}}^{g^*}$-hybrid world and under the rules on p_{mine}.

Lemma 8. *Protocol 4 achieves validity.*

Proof. P_s at round 0 sends $[b_s, \mathsf{sig}_s(b_s)]$ to all parties. Thus, by round 1, all honest parties have a valid $(1, 1)$-batch for b_s and add it to their extracted sets. Since P_s is honest, the rest is derived from the security of the signature scheme: no malicious party can inject another validly signed message, so honest parties only see valid batches for the original message. □

Lemma 9. *Protocol 4 achieves consistency.*

Proof. Suppose that an honest party P_i adds message b to $\mathsf{Extracted}_i$ at some stage r. We shall 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 rb. So, for all signatures $\mathsf{sig}_k(b) \in rb$ it holds that $G_i^{(k)} \geq g^* - 2r + 1$. Thus, during the second round of stage r, all honest parties receive $[b, rb]$. Also, at least one honest party

is in the b-committee with overwhelming probability by Rule 1. Such a party, say P_j, adds b to its $\mathsf{Extracted}_j$ set and adds its own signature to the batch, creating a $(r+1,1)$-batch rb' (from M-Validity since P_j is honest, $G_k^{(j)} \geq g^* - 1$, for all other honest parties P_k). So, P_j sends (b, rb') to all parties. Thus, during the first round of stage $r+1$, all honest parties observe a valid $(r+1,1)$-batch for message b and 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 is in the committee for b, meaning that P_i holds a valid $(r,2)$-batch and also adds their own signature to the batch, creating a valid $(r+1,1)$-batch rb' (from M-Validity for P_i). So, P_i sends (b, rb') to all parties. Thus, during the first round of stage $r+1$, all honest parties observe a valid $(r+1,1)$-batch for message b 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 rb 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 $G_i^{(j)} \geq 1$. However, by Rule 2, at least one of the signatures comes from another honest party P_k, so every honest party received this valid $(g^*/2,1)$-batch rb and adds b to their $\mathsf{Extracted}$ set by this stage.

Now assume that at any point in the protocol, if the batch rb 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. With overwhelming probability, there cannot be $g^*/2$ malicious parties in the committee and there is at least an honest party in the b-committee, so limiting the size of the transmitted batches to $g^*/2$ does not impact consistency. □

5 Sublinear round Broadcast

We now describe the broadcast protocol where we replace the $\mathcal{F}_{\mathsf{mine}}^{g^*}$ functionality by cryptographic tools in the random oracle model. Specifically, we use:

1. A Verifiable Graded Consensus (VGC) for random strings satisfying the properties in Definition 10;
2. An adaptively secure Verifiable Random Function (VRF) that achieves the properties in Definition 4 even with maliciously generated keys;
3. A bound for the output of the VRFs such that the committees have at least one honest party and fewer malicious parties than the number of rounds, with overwhelming probability.

The VCG can be seen as an online setup phase that generates graded random strings and proofs of the validity of these strings. The VRF and the bound are used to verifiably and correctly elect bit-specific committees. This happens thanks to the VCG outputs which help validate the committee membership.

Parameters. Concretely, we set the mining probability p_{mine} 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 the failure probability δ which should be constant and negligible in the security parameter κ, $\delta = \exp(-\omega(\ln \kappa))$. We set the maximum grade and the number of rounds g^* to be $g^* = 2 \cdot \lceil \frac{2}{\epsilon} \ln\left(\frac{2}{\delta}\right) \rceil = O(\kappa/\epsilon)$. Finally, the bound for the VRF output check $\mathsf{bound}_{\epsilon,\delta} = p_{\text{mine}} \cdot \frac{2^{m(\kappa)}}{n}$, where the VRF output length is $m(\kappa)$.

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 VDF pair of keys $(\mathsf{EK}_i, \mathsf{VK}_i)$ and a VRF pair of keys $(\mathsf{SK}_i, \mathsf{PK}_i)$. Before the beginning the protocol, every party P_i has posted on the bulletin-board the public keys $\mathsf{pk}_i, \mathsf{EK}_i, \mathsf{VK}_i, \mathsf{PK}_i$, such that all parties have access to the bulletin-PKI. Moreover, each party has access to the values for the rest of the network output by VGC, with their accompanying grades, and the accompanying proofs for their own values.

The local string y 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 VDF has length polynomial in the security parameter κ, then the output of a hash function H modeled as a random oracle $H(b||y)$ is random.

Mine. We instantiate the Mine call from Functionality 3 as follows. Let $i, j \in [n]$ and $b \in \{0,1\}$, SK a VRF secret key, s a VDF output and $\mathsf{bound}_{\epsilon,\delta}$ the appropriate bound for the VRF. Let $call^b$ be two vectors of n entries initialized with -1. Given $(\mathsf{SK}, b||s)$ from party i, a party checks if $call_i^b = -1$ and computes $(y, \pi) = \mathsf{VRF.Prove}_{\mathsf{SK}}(H(b||s))$. If $y < \mathsf{bound}_{\epsilon,\delta}$, set $call_i^b = 1$, else $call_i^b = \perp$. The output is given as $call_i^b$.

For our choice of parameters, the instantiation of $\mathcal{F}_{\mathsf{mine}}.\mathsf{Mine}$ and the generated bit-specific committees satisfy the following results. The proofs of Lemma 10 and of Lemma 11 are given in Appendix D.

Lemma 10 (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 overwhelming probability $\delta/2$ in the security parameter.*

Lemma 11 (Dishonest Upper Bound). *For $p_{mine} = \min\left\{1, \frac{1}{\epsilon n} \log\left(\frac{2}{\delta}\right)\right\}$, at most $g^*/2 = \lceil \frac{2}{\epsilon} \cdot \log\left(\frac{2}{\delta}\right) \rceil$ dishonest parties can self-elect in a committee for a bit b with overwhelming probability $\delta/2$ in the security parameter, for any $\epsilon \in (0,\ 1)$.*

We defined valid batches of signatures in Section 4. Here, parties (with the exception of the sender) will also send proofs of their self-election in the committee for bit b. These proofs will be submitted and echoed by the parties in certificates called cert.

Verify and valid certificates. The instantiation of the Verify call from Functionality 3 is required when checking the certificates' validity. A certificate cert consists of tuples $(y_j^{\mathsf{VDF}}, \pi^{\mathsf{VDF}}, Y_j, \pi_j)$ for every j-signature in batch. A certificate cert is a valid certificate for an $(r, \cdot)$-batch batch if for all signatures sig_j in batch not coming from the sender, and the messages $M^{(j)}$ held at the verifying party, it holds that:

- $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||y_j^{\mathsf{VDF}}), Y_j, \pi_j) = 1$ and

- $\mathsf{VDF.Verify}_{\mathsf{VK}_j}(M^{(j)}, y_j^{\mathsf{VDF}}, \pi_j^{\mathsf{VDF}}, \Delta_{\mathsf{VDF}}) = 1$.

The updated protocol that uses VDFs and VRFs to securely elect committees is described in Figure 5. 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 $6g^* + 3 + O(1)$ rounds.

We follow with the steps outlined in Protocol 4, but with the calls to $\mathcal{F}_{\mathsf{mine}}^{g^*}$ replaced with the real instantiations. As before, we implicitly assume that honest parties only send at most $g^*/2$ values in batch, including the sender's signature, and the corresponding $g^*/2 - 1$ values in cert.

Theorem 1. *Consider a Δ_{VDF}-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 5 is a Broadcast protocol with probability $1-\delta-\mathsf{negl}(\kappa)$.*

Proof. We sketch the main ideas for showing that we have a secure instantiation of $\mathcal{F}_{\mathsf{mine}}^{g^*}$ with cryptographic primitives.

The VGC protocol achieves indistinguishability for a difficulty parameter of Δ_{VDF}, M-Validity, soundness termination and verifiability: Lemmata 5, 6 and 7. Therefore, we are guaranteed that by the start of Stage 1, each party P_i has a random string y_i that they will 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, the local strings with grades greater than 2 are the same among honest parties. Then, given the string y_j^{VDF} and the proof π_j^{VDF} inside a vote in the certificate cert_b, a honest party P_i can verify that this random string was correctly generated by running $\mathsf{VDF.Verify}_{\mathsf{VK}_j}(M_i^{(j)}, y_j^{\mathsf{VDF}}, \pi_j^{\mathsf{VDF}}, \Delta_{\mathsf{VDF}})$.

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. Specifically, given malicious strings Y_j, y_j^{VDF} and malicious proofs $\pi_j, \pi_j^{\mathsf{VDF}}$ inside a vote in the certificate cert_b, a honest party P_i will not consider cert_b as valid if $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||y_j^{\mathsf{VDF}}), Y_j) = 0$ or $\mathsf{VDF.Verify}_{\mathsf{VK}_j}(M_i^{(j)}, y_j^{\mathsf{VDF}}, \pi_j^{\mathsf{VDF}}, \Delta_{\mathsf{VDF}}) = 0$.

The Δ_{VDF}-indistinguishability of the VGC guarantees that malicious parties cannot bias the output of the VDF while having grade greater than 0 in the views of honest parties (see the proof of Lemma 5). As a result, to have its vote taken into consideration, the adversary has to verifiably compute the VRF by applying a hash function to the unbiased VDF output (with grade greater than 1). Therefore, the adversary can validly self-elect with negligible probability.

Finally, since the described instantiation with the above properties enforces the rules in Section 4, validity with probability $1-\mathsf{negl}(\kappa)$ follows from Lemma 8 and consistency with probability $1-\delta-\mathsf{negl}(\kappa)$ follows from Lemma 9 that use the committees' properties from Lemmata 10 and 11. □

Broadcast

Rounds $-6g^* - 3 - O(1)$ to 0:

1. Parties run Π_{VGC}. Each party P_i obtains the following quantities $M_i^{(j)}, G_i^{(j)}$, for all $j \in [n]$, as well as $(y_i^{\mathsf{VDF}}, \pi_i^{\mathsf{VDF}}) \leftarrow \mathsf{VDF.Eval}_{\mathsf{EK}_i}(M_i^{(i)})$.

Stage 0:

1. (Round 0) Each party P_i initializes $\mathsf{Extracted}_i = \emptyset$ and $call^0 = call^1 = [-1, \ldots, -1]$ vectors of length n.
   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 ballot $(\mathsf{batch}_b, \mathsf{cert}_b)$, where batch batch_b is a valid $(r, 1)$-batch and certificate 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 some ballot $(\mathsf{batch}_b, \mathsf{cert}_b)$, where batch batch_b is a valid $(r, 2)$-batch and certificate 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 $(Y_i, \pi_i) \leftarrow \mathsf{VRF.Prove}_{\mathsf{SK}_i}(H(b||y_i^{\mathsf{VDF}}))$. 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 $\mathsf{batch}'_b := \mathsf{batch}_b || \mathsf{sig}_i(b)$, $\mathsf{cert}'_b := \mathsf{cert}_b || (y_i^{\mathsf{VDF}}, \pi_i^{\mathsf{VDF}}, Y_i, \pi_i)$.

   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. 5. Updated Broadcast protocol for designated sender P_s and parties $P_1, \ldots, P_n$.

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 $\delta = \exp(-\omega(\log \kappa))$ negligible in the security parameter, we obtain a round complexity of $O(\kappa/\epsilon)$ and a total communication complexity of $O(\kappa^2 \cdot n^4/\epsilon)$.

6 Communication reduction for parallel gradecast

The communication of sharing and agreeing on random strings is the dominating term in the Broadcast communication. To improve upon that, we leverage parallelization and randomization. We take inspiration from the recent work in [36] and use gossiping to lower the communication cost by a factor of $O(n/\mathsf{polylog}(n))$, improving the communication of Π_{VGC} (and thus of $\mathsf{Broadcast}$) to $\tilde{O}(n^3)$.

Tsimos *et al.* [36] formulates the notion of honest parties disseminating messages via gossiping in a communication-efficient way that is adaptively secure. There, the messages 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 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 [36] to meet our needs, which uses a function that takes a set and outputs only one message per each k-bit prefix. In our case, we define the set to be dispersed to be the set of all possible messages $m_{j,s}$ for each pair of sender P_j and moderator P_s. In this set, any two pairs of two messages for the same j, s are considered 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 call this type of protocol $\mathcal{M}$-Converge* and formalize it below.

6.1 The $\mathcal{M}$-Converge* Protocol

Definition 11 ($\mathsf{couples}_k$ function). *For any set M, $\mathsf{couples}_k(M)$ 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 $\mathsf{couples}_k(M)$ contains exactly those messages, and if there are more than two messages with prefix PR, then $\mathsf{couples}_k(M)$ contains only two of them.*

For example, for $M = \{00101, 01000, 01100, 11001, 11010, 11111\}$ we have that $\mathsf{couples}_2(M) = \{00101, 01000, 01100, 11001, 11111\}$, but since $\mathsf{couples}_k$ is an one-to-many function, thus $\mathsf{couples}_2(M)$ might also output $\{00101, 01000, 01100, 11010, 11111\}$. Now we present the $\mathcal{M}$-Converge* problem and protocol.

Definition 12 ($\mathcal{M}$-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, with probability $1 - \mathsf{negl}(\kappa)$, output a set $S_i \supseteq \mathsf{couples}_k\left(\bigcup_{j\in\mathcal{H}} M_j - \bigcup_{j\in\mathcal{H}} \mathcal{C}_j\right)$, when at most t parties are corrupted and where $\mathcal{H}$ is the set of honest parties in the beginning of the protocol.*

Let $p_{prop} = (10/\epsilon + \kappa)/n$. We consider the ideal functionality $\mathcal{F}_{\mathsf{prop}}$ from [36], 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 building block behind gossiping against an adaptive adversary; it is called by our $\mathcal{M}$-Converge* 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 E.

$\mathcal{M}$-Converge$^*{}_i(M_i, \mathcal{C}_i, k)$

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. 6. The $\mathcal{M}$-Converge$^*{}_i$ 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.

We present protocol $\mathcal{M}$-Converge* in Figure 6 and we prove its properties in Lemma 12. The proof is given in Appendix D.

Lemma 12. *Let $\kappa > 0$. Protocol $\mathcal{M}$-Converge* from Figure 6 is an adaptively secure $\mathcal{M}$-Converge* protocol for all $t \leq (1-\epsilon) \cdot n$ and fixed $\epsilon \in (0,1)$. The total number of bits sent by all parties is $O(n \log(\epsilon n) \cdot \max\{n, |\mathsf{couples}_k(\mathcal{M})|\} \cdot m \cdot s)$.*

6.2 Gradecast using $\mathcal{M}$-Converge*

We propose a new protocol for parallel gradecast with one instance per pair of $(j \in \text{senders}, k \in \text{casts})$. Each party P_j, for $j \in$ senders is expected to gradecast |casts| many messages. The trivial (and less efficient) way to do so would be for each such party P_j to call $\mathsf{Gradecast}(m_k^j)$, for each $k \in$ casts. Instead, by calling $\mathcal{M}$-Converge*, all P_j's can gradecast simultaneously all their |casts|

many messages, while using less communication, with a small additional cost of multiplicative $O(\log \epsilon n)$ rounds.

Parallel Vector Gradecast is a secure gradecast protocol, for each separate value each sender sends. We prove the next Lemma in Appendix D.

Lemma 13. *Protocol Parallel Vector Gradecast$_i^{(\cdot,\cdot)}(\cdot, g^*)$ is a g^*-gradecast protocol with round complexity $2g^* \cdot \lceil \log(\epsilon n) \rceil + 1$. The total communication complexity for all parties is $O(n \log \epsilon n \cdot g^* \cdot \max\{n, |senders| \cdot |casts|\} \cdot m \cdot s)$.*

Parallel Vector Gradecast$_i^{(\mathbf{senders},\mathbf{casts})}(M_i, g^*)$

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 $j \in$ senders, $k \in$ casts.
2. If $i \in$ senders, then P_i computes a signature $\sigma_k = \mathsf{sig}_i(m_k \in M_i)$, for each $k \in$ 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 $\mathcal{M}$-Converge$^*(\mathbf{M}, \emptyset, k_{sc})$, where k_{sc} is the bit-length needed to express the prefix for every pair of ($j \in$ senders, $k \in$ 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. 7. Gradecast protocol with maximum grade g^* using gossiping.

6.3 Moderated Gradecast with cubic communication

Consider our previous Protocol 1 for Parallel Mod-Gradecast. Let each party P_i call Parallel Vector Gradecast$_i^{([n],n-1)}(\{m_i^{(s)}\}_{s \in [n]-\{i\}}, g^*)$ from Figure 7 during Step 2.2, instead of Gradecast to propagate its $n-1$ received values. This allows parties to moderate the n random strings they each received during Step 1 with $\tilde{O}(g^* \cdot (\ell + \kappa)\kappa \cdot n^3)$ 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 Mod-Gradecast has the same communication and round complexity as the updated Mod-Gradecast.

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

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

7 Amortization

Notice that in Protocol 5, we use the online setup, Π_{VGC}, for a single run of 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 of Broadcast over multiple instances.

Concretely, instead of feeding $(b||y_i^{\mathsf{VDF}})$ in broadcast instance $\mathsf{br_{id}}$ to the VRF, P_i we can instead feed $(b||H(y_i^{\mathsf{VDF}}||\mathsf{br_{id}}))$, which is also guaranteed to be random, as long as y_i^{VDF} has length polynomial in the security parameter κ. Therefore, the adversary can not predict the committee membership for 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 y_i^{VDF}, their composition is secure.

The $\mathcal{F}_{\mathsf{mine}}^{g^*}$ functionality instantiation changes accordingly to compute $(y, \pi) = \mathsf{VRF.Prove}_{\mathsf{SK}}(H(b||y||\mathsf{br_{id}}))$ in Mine and $\mathsf{VRF.Verify}_{\mathsf{PK}_j}(H(b||y_j^{\mathsf{VDF}}||\mathsf{br_{id}}), Y_j, \pi_j)$ in Verify. Define Π_{BC} to be the subprotocol run between stages 0 and $g^*/2$ in the Broadcast protocol from Figure 5, with the new instantiation of the Mine and Verify calls. Then, multiple secure Broadcast instances are obtained by running the Π_{VGC} protocol (with any given instantiation) and then run $\Pi_{\mathsf{BC}}(\mathsf{br_{id}})$, for $\mathsf{br_{id}} = 1, \ldots, \mathsf{br_{max}}$.

Theorem 2. *We obtain Broadcast protocols secure against an adaptive dishonest majority of $t \leq (1-\epsilon)n$ with overwhelming probability in the security parameter κ with the amortized cost of:*

1. *$\tilde{O}(\kappa)$ rounds and $\tilde{O}(n^2)$ communication complexity over n instances;*
2. *$O(\kappa)$ rounds and $\tilde{O}(n^2)$ communication complexity over n^2 instances;*
3. *$\tilde{O}(\kappa)$ rounds and $\tilde{O}(n)$ communication complexity over n^2 instances.*

Proof. For 1, we use one instance of Π_{VGC} to "generate" random strings for $\mathsf{br_{id}} \in \{1, \ldots, n\}$. Using the protocol Π_{VGC} where we instantiate the parallel Gradecast via Protocol 7, and $\mathsf{br_{id}} \in \{1, \ldots, n\}$, yields an amortized communication complexity of Broadcast of $\tilde{O}(n^2)$, and still $\tilde{O}(\kappa)$ round complexity.

For 2, using the online setup Π_{VGC} without gossiping (using the Gradecast from Appendix B) and $\mathsf{br_{id}} \in \{1, \ldots, n^2\}$ yields an amortized communication complexity of Broadcast of $\tilde{O}(n^2)$, and $O(\kappa)$ round complexity.

For 3, we amortize the communication cost of parallel broadcast using the techniques of Tsimos *et al.* [36], who show how to achieve parallel broadcast with trusted setup based on the broadcast protocol of Chan *et al.* [9] that has a total communication of $\tilde{O}(n^2)$ and $\tilde{O}(\kappa)$ rounds. Specifically, if we use the Π_{VGC} protocol with communication cost $\tilde{O}(n^3)$ to bootstrap randomness for n^2 broadcast instances, and we run sequentially n parallel broadcast instances where we apply the gossiping techniques in Tsimos *et al.* on Protocol 5, we obtain an amortized cost of $\tilde{O}(n)$ per broadcast instance, with $\tilde{O}(\kappa)$ round complexity and without trusted setup. Note that there is no "gossiping composition" since it is applied separately for the VGC and the Broadcast.

References

1. 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.
2. 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`.
3. 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.
4. 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.
5. 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.
6. D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, Heidelberg, Aug. 2000.
7. 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.
8. V. Buterin. Ethereum white paper. White paper, Ethereum, 2013.
9. 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.
10. J. Chen and S. Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.
11. B. Cohen and K. Pietrzak. The chia network blockchain. Technical report, Chia Network, 2019.
12. 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`.
13. 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.

14. 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.
15. D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
16. P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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`.
23. S. Goldberg, J. Vcelak, D. Papadopoulos, and L. Reyzin. Verifiable random functions (VRFs), 2018.
24. R. Hou, H. Yu, and P. Saxena. Using throughput-centric byzantine broadcast to tolerate malicious majority in blockchains. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1263–1280. IEEE, 2022.
25. 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.
26. V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *17th SODA*, pages 990–999. ACM-SIAM, Jan. 2006.
27. A. K. Lenstra and B. Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017.
28. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999.
29. 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.
30. K. Pietrzak. Simple verifiable delay functions. In A. Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, Jan. 2019.
31. M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27:256–267, 1983.
32. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.

33. 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.
34. R. Shostak, M. Pease, and L. Lamport. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
35. 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.
36. G. Tsimos, J. Loss, and C. Papamanthou. Gossiping for communication-efficient broadcast. *CRYPTO 2022*, 2022.
37. 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.
38. 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.
39. 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.

A More preliminaries

Verifiable Random Functions. The extended pseudorandomness game works as follows. The adversary generates the key pair and an input, but a challenger samples a random string that should be concatenated with the input and a random input. The adversary will 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.

Definition 13. *The VRF properties are:*

(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 $(y, \pi) = \mathsf{VRF.Prove}_{\mathsf{SK}}(x)$ *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:*

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

(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:*

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

Verifiable Graded Randomness. Recall from the σ-sequentiality definition that the adversary cannot run in more time than $\sigma(\Delta)$.

Definition 14. *[*Δ*-Indistinguishability Game] An adversary* $\mathcal{A} := (\mathcal{A}_0, \mathcal{A}_1)$ *and a challeger* $\mathcal{C}$ *play the following 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 $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa, \Delta)$ 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 consistent with Δ.*
6. *$\mathcal{C}$ and $\mathcal{A}$ execute the protocol $\mathcal{A}_0^{\mathsf{Toss}(\mathsf{pp})}$; $\mathcal{A}$ can corrupt additional parties.*
7. *The protocol ends when all remaining honest parties $P_i \in \mathsf{Honest}$ have generated output $(M_i^{(j)}, G_i^{(j)})$. Denote by st the state of $\mathcal{A}$ obtained so far.*
8. *For each j, $\mathcal{A}$ computes $y_A^{(j)} \leftarrow \mathcal{A}_1^{\mathsf{Process}(\mathsf{pp},\cdot)}(\mathsf{pp}, st)$.*

The adversary $\mathcal{A}$ wins the game if for at least one index j, $y_A^{(j)} = y^{(j)}$ for $(y^{(j)}, \pi) \leftarrow \mathsf{Process}(\mathsf{pp}, M_i^{(j)})$ for $G_i^{(j)} \geq 1$ for any honest party $P_i \in \mathsf{Honest}$.

B Gradecast

Gradecast(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. 8. Gradecast protocol with maximum grade g^*.

Lemma 14. *Protocol 8 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 [19]. 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 a round apart. However,

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 $O(g^* \cdot (\kappa + \ell) \cdot n^2)$.

C Inequalities

Lemma 15 (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 $\mu := \mathbb{E}[X] = \sum_{i=1}^{n} p_i$:*

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

Lemma 16 (Bernoulli's inequality). *For every $x \in \mathbb{R}$ and any positive exponent $r > 0$, it holds that:*

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

D Postponed proofs

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

Since H_G is a random oracle, $H_G(x_1||x_2)$ is random, and for large enough polynomial $n(\kappa)$, 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 it performs to abort. In particular, instead of aborting if $x \xleftarrow{\$} \{0,1\}^{l(\kappa)}$ is already queried by the oracle H_G, $\mathcal{C}$ now aborts if $x_1 \| x_2 : x_1 \xleftarrow{\$} \{0,1\}^{l(\kappa)}$ 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. □

Proof of Observation 1. 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. □

Proof of Lemma 5. Recall the Δ-indistinguishability game in Definition 14. The adversary $\mathcal{A}$ receives Δ_{VDF} and the public keys of the honest users from the challenger $\mathcal{C}$, and sets the public keys of the currently corrupted parties $\mathcal{M}$. Every time the adversary corrupts a new party, it will add its identity to $\mathcal{M}$, but cannot change the public keys. For each $j \in \mathcal{M}$, EK_j and VK_j are checked by the challenger to be valid (possible because the VDF used has a transparent

setup). The adversary can start evaluating $\mathsf{VDF}_{\mathsf{EK}_j}$ on any string it wishes for any j.

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 $\mathsf{Gradecast}$. 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.

The adversary can start evaluating $\mathsf{VDF.Eval}$ on any combination of strings m_i of the honest parties and any other strings. By the fact that 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_{VDF}.

The adversary participates in the $\mathsf{Gradecast}$ 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 $(M_i^{(j)}, G_i^{(j)})$. 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 and starts $\mathsf{Process}$ using behavior $\mathcal{A}_1$ to obtain $y_A^{(j)}$.

By the fact that $\mathcal{A}$ is $\sigma(\Delta_{\mathsf{VDF}})$-parallel time limited and the VDF is σ-sequential, it holds from Lemma 1 that for any value $M_j^{(j)}$ obtained after the start of Toss as a function of the remaining honest parties, the advantage of the adversary of guessing $y^{(j)} = \mathsf{VDF.Eval}(\mathsf{pp}, M_j^{(j)})$ is $p_1 = \mathsf{negl}(\kappa)$.

We now show that each value $M_j^{(j)}$ for which $\mathcal{A}$ could have guessed $y^{(j)}$ has to have grade $G_i^{(j)} = 0$ by all other honest parties. To see that, recall that the final grade for a party P_j is set by party P_i as $G_i^{(j)} = \min_{k\in[n]}\{G_{k,i}^{(j)}\}$. To obtain a biased output y_j^{VDF}, by the Δ_{VDF}-sequentiality of the VDF, the adversary must have computed it on values not depending on the honest parties' random seeds. This guarantees that at least for one index k corresponding to an honest party P_k, P_i has not observed $m_{k,i}^{(j)} = m_{k,i}$ and since $g_{k,i} = g^*$, by validity of $\mathsf{Gradecast}$, the rule from Step 3 in Protocol 1 specifies that $G_{k,i}^{(j)} = 0 = G_i^{(j)}$.

Therefore, the advantage of the adversary $\mathcal{A}$ in winning the Δ_{VDF}-indistinguishability game is at most $p_0 + p_1 = \mathsf{negl}(\kappa)$, so VGC is σ_Δ-indistinguishable cf. Definition 5. □

Proof of Lemma 6. The soundness and M-validity follow immediately from the soundness and M-validity of $\mathsf{Mod\text{-}Gradecast}$ (Lemmata 3 and 2), and from Lemma 4. □

Proof of Lemma 7. The termination of Toss is inherited by the termination of $\mathsf{Mod\text{-}Gradecast}$, and termination of $\mathsf{Process}$ is guaranteed by the property of $\mathsf{VDF.Eval}$ that an output is generated after time Δ_{VDF}. Verifiability is inherited from the soundness of the VDF. □

Proof of Lemma 10. There are at least ϵn parties that are forever honest by assumption. Let X denote the number of honest parties managing to elect themselves in a committee for a given by 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 (see (2) in Appendix C).

Then, the statement in the Lemma, $\mathbb{P}[X \geq 1] \geq 1 - \delta/2$ holds because δ is negligible in the security parameter. □

Proof of Lemma 11. 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 by b. Even if nodes are corrupted adaptively, the election probability does not change. The expected value of X is:

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

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 (see (1) in Appendix C). 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:

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

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

$$R \geq \left(1 + \frac{\epsilon + \sqrt{8\epsilon - 7\epsilon^2}}{2(1-\epsilon)}\right) \cdot \mathbb{E}[X] = \frac{2 - \epsilon + \sqrt{8\epsilon - 7\epsilon^2}}{2(1-\epsilon)} \cdot \frac{1-\epsilon}{\epsilon} \cdot \log\left(\frac{2}{\delta}\right)$$
$$= \frac{2 - \epsilon + \sqrt{8\epsilon - 7\epsilon^2}}{2\epsilon} \cdot \log\left(\frac{2}{\delta}\right) \leq \frac{3.1}{2\epsilon} \cdot \log\left(\frac{2}{\delta}\right).$$

For a slightly simpler expression, we set $R := \lceil \frac{2}{\epsilon} \cdot \log\left(\frac{2}{\delta}\right) \rceil$, obtaining the $g^*/2$ value in the statement.

The value we obtain for $g^*/2$ is slightly tighter than the one in Chan *et al.* because we use the tighter version of the Chernoff inequality. □

Proof of Lemma 12. The proof can be adapted from the proof of Theorem 2 in [36]. 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 one party is, with probability $1 - \mathsf{negl}(\kappa)$:

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

Proof of Lemma 13. We prove validity and consistency separately.

Validity: Let an honest sender P_s 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 is $g^i_{(s,k)} = 2g^*$ 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 $g^i_{(s,k)} \geq 1$. This means that $\bar{g}^i_{(s,k)} \geq 2g^i_{(s,k)}$ 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 $S^i_{(s,k)}$. Then, it holds 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 it doesn't hold, then $r's \leq r + 2g^i_{(s,k)} - 3$. P_j is honest, thus it calls $\mathcal{M}$-Converge$^*(\mathbf{M}_j, \emptyset, \cdot)$ during round $r' + 1$, with $m^j_{(s,k)} \in \mathbf{M}_j$. From Lemma 12, by the end of round $r'+1$ all honest parties have 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, $r' > r+2g^i_{(s,k)}-3$. P_i, therefore calls $\mathcal{M}$-Converge$^*(\mathbf{M}_i, \emptyset, \cdot)$ 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 from how $\mathcal{M}$-Converge* works, each honest party received at least two distinct messages for (s,k). Then, by the previous claim, $2g^i_{(s,k)} < 3$, 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 $r-1$. Since by the end of round r it is $|S^j_{(s,k)}| \geq 2$, then $\bar{g}^j_{(s,k)} \leq 2$, i.e. $g^j_{(s,k)} \leq 1$.

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 $\mathcal{M}$-Converge* 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. □

E Secure instantiation of $\mathcal{M}$-Converge*

The protocol $\mathcal{M}$-Converge* was introduced in Section 6 in the $\mathcal{F}_{\mathsf{prop}}$-hybrid world, for $\mathcal{F}_{\mathsf{prop}}$ defined in Functionality 9. We instantiate $\mathcal{M}$-Converge* via the protocol Propagate, presented in Figure 10 as it appears in [36].

The instantiation requires a one-time use of a CPA-secure public key encryption scheme, defined next.

Definition 15 (PKE). *A public key encryption (PKE) scheme is a 3-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.*

PROPAGATE is a secure instantiation of $\mathcal{F}_{\mathsf{prop}}$, assuming a CPA-secure public key encryption scheme and erasures, cf. Lemma 8 in [36].

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

Let $p_{prop} = (10/\epsilon + \kappa)/n$. 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. 9. Functionality $\mathcal{F}_{\mathsf{prop}}$ for parties $P_1, \ldots, P_n$.

PROPAGATE(SendRandom, M_i)

Let $\Lambda_i = 2m \left\lceil \frac{|M_i|}{n} \right\rceil$, $O_i = \emptyset$ 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. 10. An instantiation of $\mathcal{F}_{\mathsf{prop}}$