Simple Is COOL: Graded Dispersal and Its Applications for Byzantine Fault Tolerance

Ittai Abraham
Intel Labs, Petah Tikva, Israel

Gilad Asharov
Bar-Ilan University, Ramat-Gan, Israel

Anirudh Chandramouli
Bar-Ilan University, Ramat-Gan, Israel

Abstract

The COOL protocol of Chen (DISC'21) is a major advance that enables perfect security for various tasks (in particular, Byzantine Agreement in Synchrony and Reliable Broadcast in Asynchrony). For an input of size L bits, its communication complexity is $O(nL + n^2 \log n)$, which is optimal up to a $\log n$ factor. Unfortunately, Chen's analysis is rather intricate and complex.

Our main contribution is a simple analysis of a new variant of COOL based on elementary counting arguments. Our main consistency proof takes less than two pages (instead of over 20 pages), making the COOL protocol much more accessible. In addition, the simple analysis allows us to improve the protocol by reducing one round of communication and reducing the communication complexity by 40%.

In addition, we suggest a new way of extracting the core properties of COOL as a new primitive, which we call Graded Dispersal. We show how Graded Dispersal can then be used to obtain efficient solutions for Byzantine Agreement, Verifiable Information Dispersal, Gradecast, and Reliable Broadcast (in both Synchrony and Asynchrony, where appropriate). Our improvement of COOL directly applies here, and we improve the state-of-the-art in all those primitives by reducing at least one round and 40% communication.

2012 ACM Subject Classification Security and privacy; Security and privacy → Cryptography; Security and privacy → Information-theoretic techniques; Security and privacy → Distributed systems security

Keywords and phrases Byzantine Agreement, Broadcast

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.1

Funding *Gilad Asharov*: Supported by the Israel Science Foundation (grant No. 2439/20), and by JPM Faculty Research Award.
Anirudh Chandramouli: Same as Gilad Asharov.

1 Introduction

Byzantine agreement (BA), introduced by [27, 22], is a fundamental problem in distributed protocol design. It involves n parties, each holding an initial input v_i, who must agree on a common value v. BA requires satisfying two key properties: **Agreement**, where all honest parties must output the same value, and **Validity**, which ensures that if all honest parties begin with the same initial value v, their output should also be v. These security properties must be guaranteed even in the presence of up to t corrupted parties, which may deviate arbitrarily from the protocol's specifications. This work focuses on perfect security with optimal resilience ($t < n/3$).

It is known that agreement on a single bit requires $\Omega(n^2)$ bits of communication when the protocol is deterministic [16], or even randomized, against strongly-adaptive adversaries [3] (for any $t = O(n)$). Moreover, agreeing on an L bit message requires $\Omega(nL)$ communication,

since we might have $O(n)$ parties that do not hold the agreed message as input and to receive that message. Furthermore, it has been shown that for any Byzantine agreement protocol with perfect security, there exists an execution that requires $t+1$ rounds [18]. These impossibilities imply that the best one can hope for is $O(nL + n^2)$ for agreeing on a message of size L. However, achieving this bound has been an elusive goal for extensive research.

The baseline protocol [10] for binary Byzantine agreement has a communication complexity of $O(n^2)$ and runs in $\Theta(n)$ rounds. For messages of size L, this implies a communication complexity of $O(n^2L)$. Liang and Vaidya [23] proposed a protocol with a communication complexity of $O(nL + n^4\sqrt{L} + n^6)$ communication; Ganesh and Patra [19], and Loveless, Dreslinski, and Kasikci [24] introduced protocols with a complexity of $O(nL + n^4 \log n)$ communication; Subsequently, Nayak, Ren, Shi, Vaidya, and Xiang [25] improved on this with a protocol that achieves $O(nL + n^3 \log n)$ communication.

The COOL protocol. The paper of Chen [13] further advanced the state of the art by presenting a *deterministic* byzantine agreement protocol, named COOL (stands for **co**ded BA protoc**ol**), that achieves $O(nL + n^2 \log n)$ communication complexity, which is only a logarithmic factor away from the best possible one can hope for. He shows:

▶ **Theorem 1.1** (Byzantine agreement [13]). *There exists a perfectly secure and deterministic Byzantine agreement protocol over a synchronous network for messages of size L bits with communication complexity of $O(nL+n^2 \log n)$ with $\Theta(n)$ rounds, resilient to $t < n/3$ malicious faults.*

Unfortunately, the analysis in [13] is complex and lengthy, making it difficult to understand. Our goal is to provide a simplified analysis of the protocol, offering new insights into the technique and its limitations. The COOL approach has already been applied to related primitives, such as broadcast and gradecast [30] in the synchronous settings, and reliable broadcast in asynchronous settings [6], and holds promise for broader applications as well.

1.1 Our Contributions

In this paper, we provide the following contributions:

Graded dispersal. The COOL protocol of [13] is tailored for Byzantine agreement, and its core was later adopted for reliable broadcast [6] and gradecast by [30]. We show that this core of COOL is essentially a new variant of dispersal (a part of a Verifiable Information Dispersal protocol [12]). We call this new variant *graded dispersal*. We note that dispersal typically needs a designated sender; in our abstraction, we assume the parties have already received the message from some "virtual sender" and wish to verify that they have enough information to retrieve and agree on it later. Defining this graded dispersal variant allows us to distill the novelty of COOL's protocol consistency check and build applications that are based on COOL in a more modular fashion.

Simple analysis. The analysis in [13] is rather involved and relies on various tools; citing [13] "the proof borrows tools from coding theory, together with graph theory and linear algebra". As mentioned above, this also makes the result hard to verify. We provide a new analysis of our graded dispersal protocol. Notably, our analysis is significantly more concise, much shorter than the original COOL analysis, and uses only simple counting. Given the centrality of the result and its simple analysis, we believe that this protocol will be a good candidate for being part of a curriculum in distributed computing classes as a prime example of the centrality of polynomials in obtaining efficient perfect security.

Applications. We show how to apply graded dispersal to achieve the following (1) gradecast (synchrony); (2) multi-valued Byzantine agreement; (3) broadcast; and (4) reliable broadcast (asynchrony).

Gradecast introduced by Feldman and Micali [17] ("graded-broadcast") is a fundamental primitive used for various consensus algorithms.

Bracha's Reliable Broadcast [11] is a useful building block in asynchronous protocols. The recent paper of Das, Xiang, and Ren [15] has a detailed survey of the usefulness of communication-efficient Reliable Broadcast. In particular, it implies Asynchronous Verifiable Secret Sharing and Asynchronous Distributed Key Generation [4].

One round less. Our new protocol reaches agreement faster than the COOL protocol. Specifically, while COOL requires three rounds of reporting agreements to one another (see more in-depth in Section 2), we show that two rounds suffice. This improvement becomes particularly significant when utilizing the COOL technique in constant round protocols such as reliable broadcast and gradecast - and improves the state-of-the-art for those primitives.

Less communication. Our new protocol improves the communication complexity by 40%. Specifically, the protocol of COOL divides the messages into blocks of polynomials of degree $d = t/5$. Our protocol works when dividing the messages into blocks of degree $d = t/3$. Changing the degree enhances the state-of-the-art in all our applications – multi-valued byzantine agreement, broadcast, gradecast, and reliable broadcast.

We consolidate the costs of gradecast and reliable broadcast in Table 1.

Table 1 The communication complexity (in bits) and round complexity of our gradecast and reliable broadcast compared to previous works. We compare here protocols with perfect security and optimal resilience only (see Section 2.3 for other settings). The reported numbers of rounds are for non-balanced protocols; Making the protocols balanced increases the number of rounds by one (except for [17, 11], which remain the same).

Gradecast (synchrony)

	Communication	Rounds
[17]	$O(n^2L)$	2
[2]	$O(nL + n^3 \log n)$	9
[30]	$O(nL + n^2 \log n)$	8
This work	$O(nL + n^2 \log n)$	5

Reliable Broadcast (asynchrony)

	Communication	Rounds
[11]	$O(n^2L)$	4
[26]	$O(nL + n^4 \log n)$	11
[25]	$O(nL + n^3 \log n)$	7
[6]	$O(nL + n^2 \log n)$	9
This work	$O(nL + n^2 \log n)$	6

2 Technical Overview

In this section, we provide a technical overview of the different primitives. We denote by n the number of parties and by t the number of corrupted parties. We assume a finite field $\mathbb{F}$ of size at least $n + 1$ (and for simplicity, $\mathsf{poly}(n)$). We let $t < n/3$ bound the number of corrupted parties (corresponding to optimal resilience in the plain model, that is, in the absence of a PKI).

We consider both synchronous communication, in which protocols proceed in rounds (Section 2.1), and asynchronous communication networks (Section 2.2). Messages between honest parties in the asynchronous network are guaranteed to be delivered, but the adversary can introduce arbitrary delays.

2.1 Synchrony

The COOL protocol in a nutshell. The protocol of COOL divides the value v_i of size L into blocks, where each block is a polynomial of degree-d over a finite field $\mathbb{F}$. For simplicity of exposition, assume that $L = (d+1)\log|\mathbb{F}|$ (i.e., we have a single block – one polynomial).

Two parties P_i and P_j that wish to check whether their polynomials $f_i(x), f_j(x)$ are identical can do so efficiently using randomization and with a statistical error based on the Schwartz-Zippel lemma [29, 31]. Each pair evaluates its respective polynomials on a random point $r \in \mathbb{F}$ and verify that $f_i(r) = f_j(r)$. This approach was taken in [1] for gradecast and reliable broadcast. The novelty of COOL protocol is that it achieves a similar effect of equality check of polynomials, *deterministically*, and *error free*. Each pair of parties exchange just $O(1)$ points, as in the statistical case. Specifically, P_i sends to P_j the points $f_i(i)$ and $f_i(j)$ only, and P_j verifies that those two points agree with its polynomial $f_j(x)$. If a party agrees with $n-t$ parties - then the party is happy. The protocol proceeds in a few rounds, during which each party reports to others whether it is happy. A party remains happy if at least $n-t$ parties in its agreement set report they are happy. Interestingly, the proof of COOL shows that after three rounds of repeating this process, all parties that report that they are happy must hold the same polynomial.

Graded dispersal. We formalize a variant of dispersal, which, as mentioned, distills the consistency property achieved by COOL. We require the following properties from a graded dispersal protocol. Each party holds some polynomial $f_i(x)$ as input (of degree-d). The parties output $(f'_i(x), g_i)$ where $f'_i(x)$ is a polynomial of degree-d (might be $\perp$), and g_i is a grade in $\{0,1,2\}$. The guarantees are: (1) Validity: If all honest parties start with the same polynomial $f(x)$, then they all must output $(f(x), 2)$; (2) Graded agreement: If some honest party outputs $(f(x), 2)$ then at least $t+1$ honest party output $f(x)$ with a grade at least 1, and all other honest parties output $(\perp, 0)$. Note that we have no guarantee in the case where no honest party outputs grade 2, i.e., in that case, different honest parties might have different polynomials as output (with grade 1). Furthermore, our formalization of graded dispersal allows two honest parties to output grades of 2 and 0, respectively. This is different from the typical definition of graded primitives (such as gradecast), where if an honest party outputs grade 2, then all honest parties output a grade at least 1. However, looking ahead, we show that this weak formalization of graded dispersal suffices for all the applications we consider, including reaching a full byzantine agreement.

We implement graded-dispersal based on the COOL protocol. The crux of our contribution is in providing a simple proof for this construction. Moreover, we show we can achieve it in one round of reporting happiness less than COOL. Specifically, we show that:

1. Initially, only two sets of parties, A and B, holding polynomials $f_A(x)$ and $f_B(x)$, respectively, can report that they are happy after the initial pairwise checks. Since two polynomials might agree on only d common points, three parties holding distinct polynomials cannot have enough agreements and cannot all be happy. See Claim 3.4.
2. We show that if there is one set of parties A, all holding the same polynomial $f_A(x)$ and $|A| \geq t+1$, then only parties from A can output grade at least 1, and all other parties output $(\perp, 0)$. Intuitively, if another set of parties B with $|B| \leq t$ holds polynomial $f_B(x)$, then those parties cannot remain happy due to inconsistencies with the larger set A. See Claim 3.5. Importantly, a party outputs grade 2 if it is happy and it received at least $2t+1$ reports from its agreement set in the last round. This implies that it received at least $t+1$ reports from honest parties. In that case, a party can output grade 2, and the other happy parties must also hold $f_A(x)$, and we have at least $t+1$ parties with the same (and only) output polynomial, $f_A(x)$.

3. If no large set exists, no party outputs grade 2 (no party is happy and received $2t+1$ reports of happiness from its agreed set in the last round). See Claim 3.6.

Our graded dispersal works in three rounds and requires $O(nL+n^2\log n)$ for inputs of size L.

Data dissemination. In Section 3.2, we show a protocol that allows parties to reconstruct the output $f(x)$ once there are at least $t+1$ honest parties that hold the same input $f(x)$ (and other honest parties hold $\perp$). Essentially, this is precisely the guarantee we have once one party outputs 2 in the graded dispersal protocol. Simply, each party that has an input $f(x)$ sends to each party P_i its point $f(i)$. P_i then takes as its point the majority value that it received, and since $t+1$ honest parties that hold $f(x)$ and there are at most t corrupted parties, we are guaranteed that the majority of P_i is $f(i)$. Once each party P_i has a point $f(i)$, the parties reconstruct f by sending their points to one another and using Reed-Solomon decoding.

Given graded dispersal and data dissemination, we provide three applications in synchrony.

Application I: Gradecast. Gradecast is a relaxation of broadcast and involves a sender that holds a message M, and all parties output (M_i, g_i) with $g_i \in \{0,1,2\}$. If the sender is honest, all parties must output $(M,2)$. The guarantee is that if one honest party outputs $(M,2)$, all honest parties must output (M,g_i) with $g_i \geq 1$. We remark that the definition that we provide is a relaxation of Gradecast that was formalized by Katz and Koo [21], which allows the parties to output different messages when there is no honest party with grade 2 (just as in our graded dispersal). We are unaware of any application of gradecast in which the relaxed definition does not suffice.

We show how to implement gradecast using graded dispersal and data dissemination. Specifically, the sender sends its message to all parties, and the parties then run graded-dispersal. After that, they run data dissemination and output the message they received in the data dissemination with their grade in the graded dispersal. It is easy to see that the requirements for gradecast are met.

The last round of graded dispersal can be run parallel to the first round of data dissemination, so we can get a protocol that runs in a total of 5 rounds. This results in an improvement of 3 rounds compared to the state-of-the-art [30]; yet, our guarantee is slightly weaker. We also show how to satisfy the [17] definition at the expense of adding one more round. We then get a strict improvement of 2 rounds with the same security definition (and reduce communication by 40%). The improvement of 2 rounds is due to some redundancy in the protocol of [30].

We emphasize that the "relaxed" gradecast suffice for all the applications we are aware of, specifically to achieve broadcast with an expected constant number of rounds (see Section 2.3). We therefore advocate using the relaxed version.

Application II: Multi-Valued Byzantine Agreement. We compose graded-dispersal, with binary agreement (on whether a party has grade 2), and data dissemination to achieve multi-valued byzantine agreement. As mentioned, our protocol has one less round than the state-of-the-art and 40% less communication. The protocol runs in $O(nL+n^2\log n)$ with $\Theta(n)$ rounds (due to the binary agreement).

Application III: Broadcast. Again, by composing graded-dispersal, binary agreement, and data dissemination, we obtain a broadcast protocol that runs in $O(nL+n^2\log n)$ communication and $\Theta(n)$ rounds.

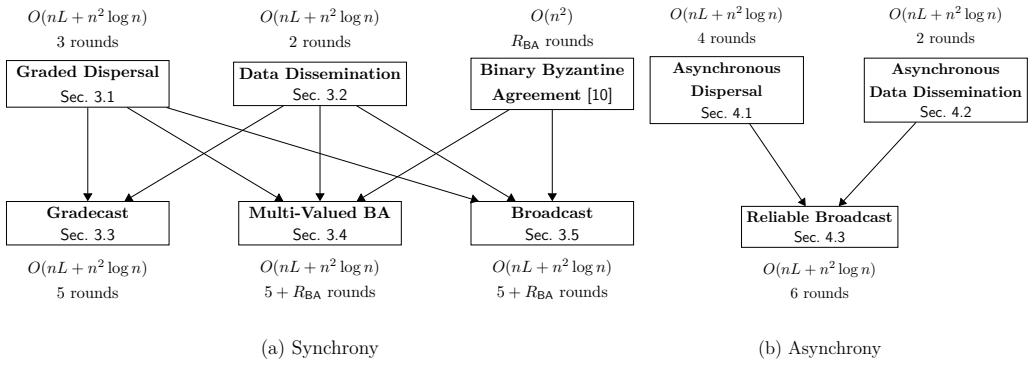

Figure 1 A roadmap of the paper, and the different primitives we construct. Next to each primitive, we report the round complexity and the communication complexity. When reporting the number of rounds – we report for a non-balanced protocol variation. Making each protocol balanced requires one more round of communication.

For gradecast and broadcast, we also show a variant where the protocol is balanced: each party sends or receives $O(L + n \log n)$, and there is no party whose communication load is higher than the others. This comes at the expense of adding an extra round.

2.2 Asynchrony

In reliable broadcast, there is a sender who holds a message M. If the sender is honest, then the protocol always terminates, and all honest parties must output M. If the sender is corrupted and the honest parties terminate, then they must all output the same message M'. Note that the protocol might not terminate if the sender is not honest.

Based upon graded dispersal in synchrony, we show a dispersal protocol in asynchrony (Section 4.1). We show perfect asynchronous data dissemination which has a weak agreement property. Our protocol and formalization is identical to that of [14] and we present it here for completeness of our result.

We have to add one more round than synchrony to guarantee the termination of honest parties for the honest sender case. We also provide an asynchronous variant of data dissemination (Section 4.2). Combining those primitives, we obtain a reliable broadcast.

Our reliable broadcast is deterministic and runs in 6 rounds. This is an improvement of 3 rounds, in addition to 40% reduced communication, compared to [6] that is based on COOL. See Section 4.3 for the protocol, and Table 1 for the improvements over previous protocols.

2.3 Related Work

All our protocols are deterministic and error-free. The applications we are considering (broadcast, multi-valued BA, gradecast, and reliable broadcast) were also studied in different settings.

Protocols with an expected number of rounds. As mentioned, broadcast and byzantine agreement in that setting must incur $\Omega(n)$ rounds [18]. The broadcast and byzantine agreement presented in our paper achieves $O(n)$ rounds in the worst case. Rabin [28] and Ben-Or [8] studied the effect of randomization on round complexity, and Feldman and Micali [17] gave the first protocol with an expected constant round protocol for byzantine agreement with optimal resilience. Many works have improved the communication complexity

of expected constant round Byzantine agreement protocols with optimal resilience [20, 25, 2, 7] where the current state of the art is $O(nL + n^3 \log^2 n)$ with expected constant number of rounds. We remark that in all those protocols, the sender first gradecasts its message, and therefore using our improved gradecast (whose relaxed notion suffices for this application) improves the constant in the "expected constant" of these works.

Gradecast. Gradecast is an important primitive used for various consensus algorithms, e.g., multi-consensus, approximate agreement [9], or Phase-King [5]. Our round complexity is the most efficient in the perfect setting. However, the protocol of Abraham and Asharov [1] is one round better at the expense of introducing some error (i.e., statistical security).

Reliable broadcast. In the computational setting, the protocol of Cachin and Tessaro [12] achieves $O(nL+\kappa n^2 \log n)$ where κ is the computational security parameter. This is improved by Alhaddad et al.[6] to $O(nL + n^2\kappa)$ with 4 rounds of communication. The recent work by Locher and Shoup looks into the constants and obtains $1.5nL + O(\kappa n^2 \log n)$.

Organization

The rest of the paper is organized as follows. The synchronous protocols appear in Section 3, and the asynchronous protocols appear in Section 4. See also Figure 1 for the previous primitives and their interplay.

3 Synchrony

In this section, we provide the protocols for synchronous communication.

Preliminaries. In the analysis, we sometimes use $H \subseteq [n]$ to denote the set of honest parties and $I \subseteq [n]$ to denote the set of corrupt parties. Our protocols also use Reed Solomon decoding. Let $R = \{(i, v_i)\}$ denote a set of evaluations of a degree d polynomial over $\mathbb{F}$ where up to t of the points may be incorrect. If $|R| > 2t + d$, then there exists an efficient algorithm that upon input R, d and t, can recover a degree-d polynomial $f(x)$ such that for $|R| - t$ points $(i, v_i) \in R$, it holds that $f(i) = v_i$. We denote this algorithm as $\mathsf{RSDec}(R)$.

Communication complexity. In each of our protocols, we assume that the input (either of the designated sender or each one of the parties) is a polynomial of degree at most d over the field $\mathbb{F}$. For the general case of an L-bit input, the parties segment the L-bit message into $\ell = \left\lceil \frac{L}{(d+1)\log|\mathbb{F}|} \right\rceil$ blocks of $(d+1)\log|\mathbb{F}|$ bits each (with appropriate padding for the last block if L is not a multiple of $(d+1)\log|\mathbb{F}|$). The parties then execute ℓ instances of the same protocol, where the input for the mth instance is the mth block interpreted as a polynomial of degree at most d. Aggregation over the ℓ different executions is performed in a natural way, e.g., a party P_i agrees with P_j if and only if they agree in all the ℓ instances. We, therefore, omit such details. We report the communication complexity of our protocols over inputs of size L bits while we present the protocols assuming $|L| = (d+1)\log|\mathbb{F}|$. Additionally, note that $\log|\mathbb{F}| \in O(\log n)$ and our communication complexity is reported in that light.

The degree d. We set the degree of our polynomials to be $d = \lfloor t/3 \rfloor$. Looking ahead, we remark that the only place this is necessary (except for requiring $d \leq t$) is in the proof of Claim 3.4.

3.1 Building Block I: Graded Dispersal

We introduce the notion of graded dispersal:

▶ **Definition 3.1** (Graded Dispersal). *A protocol for parties $P_1, \ldots, P_n$ where*

- **Input:** *Each party P_i holds as input some polynomial $f_i(x)$ of degree d over $\mathbb{F}$;.*
- **Output:** *An output $(f_i(x), g_i)$ with grade 2 or 1, or an output $(\perp, 0)$.*

is graded dispersal protocol tolerating t malicious parties *if the following properties hold:*

- **Validity:** *If all honest parties start with the same polynomial $f(x)$, then all output $(f(x), 2)$.*
- **Weak Graded Agreement:** *If an honest party outputs $(f(x), 2)$, then at least $t+1$ honest parties output $(f(x), g_i)$ with $g_i \geq 1$, and all other honest parties output $(\perp, 0)$.*

Note that if there is no honest party with grade 2 then honest parties with non-$\perp$ output and grade 1, may disagree on their output value. Moreover, grade 2 does not imply full agreement among the honest parties.

Nevertheless, we show that this Weak Graded Agreement property is sufficient to work well with the Data Dissemination protocol (see Section 3.2).

Protocol 3.2: Graded Dispersal

Input: Each party P_i holds $f_i(x)$ of degree at most $d = \lfloor t/3 \rfloor$ over $\mathbb{F}$ to encode $d \log n$ bits.
The protocol:

1. **Exchange:**
   a. P_i sends $(f_i(i), f_i(j))$ to each P_j.
2. **Dynamic set A_i^1:**
   a. Let (u_j, v_j) be the two values that P_i receives from P_j. If $f_i(j) = u_j$ and $v_j = f_i(i)$ then add j to A_i^1.
   b. If $|\mathsf{A}_i^1| \geq n - t$ then send OK_1 to all parties.
3. **Dynamic set A_i^2:**
   a. If OK_1 is received from P_j for $j \in \mathsf{A}_i^1$ then add j to A_i^2.
   b. If $|\mathsf{A}_i^2| \geq n - t$ then send OK_2 to all parties.
4. **Output:**
   a. If sent OK_2, and received at least $2t+1$ messages OK_2, then output $(f_i(x), 2)$.
   b. If sent OK_2, but received $\leq 2t$ messages OK_2, then output $(f_i(x), 1)$.
   c. Otherwise, output $(\perp, 0)$.

In the COOL protocol, there is an additional round of OK_3 messages and the degree is $d = \lfloor t/5 \rfloor$. In our protocol, there are just two rounds of OK and the degree is $d = \lfloor t/3 \rfloor$. Hence our protocol reduces the number of rounds from 3 to 2 (by %33) and reduces the overhead per bit from 15 to 9 (by %40). Note that when $L > d \log n$, only the first round is parallelized.

▶ **Theorem 3.3.** *Protocol 3.2 is a graded-dispersal protocol for messages of size $L = d \log n$ bits where $d = \lfloor t/3 \rfloor$ that is resilient to $t = \lfloor (n-1)/3 \rfloor$ malicious parties. The protocol runs in 3 rounds and incurs a total communication of $\leq \binom{n}{2} \log n + 2\binom{n}{2} < n^2 \log n$ bits.*

Proof. We now show that all the properties hold as in Definition 3.1.

Validity. Assume that all honest parties have the same input $f(x)$. For each party P_i, the set A_i^1 will contain the set of all honest parties and therefore $|\mathsf{A}_i^1| \geq n - t$ and P_i sends OK_1. Moreover, it will receive OK_1 from all honest parties; therefore, P_i also sends OK_2. We get that all honest parties send OK_2, and consequently, all receive at least $2t + 1$ OK_2, so all output $(f(x), 2)$.

Weak graded agreement. If one party outputs $(f(x), 2)$, then there are at least $t+1$ honest parties that output $(f(x), g)$ with $g \geq 1$, and all the others output $(\perp, 0)$. This is done in 3 steps:

- Claim 3.4 proves that there cannot be three honest parties that hold different inputs, that all send OK_1. Hence there exists at most two inputs f_a, f_b such that each honest party that sends OK_1 must hold one of these two inputs.
- Given this, Claim 3.5 proves that if there is a set of at least $t + 1$ honest parties that all have the same input, then no honest party holding a different input will send OK_2.
- Claim 3.6 shows that if there is no set with at least $t + 1$ honest parties that have the same input, then no honest party outputs grade 2.

Proof of Weak graded agreement given the Claims 3.4, 3.5, 3.6. If an honest party outputs grade 2, then from Claim 3.6 we must be in the case where there are at least $t + 1$ honest parties as in Claim 3.5 and from that all honest parties that send OK_2 (hence output grade 1) must output the same value.

Proving Claims 3.4, 3.5, 3.6 concludes the proof of the Theorem.

▷ Claim 3.4. There can be at most two distinct inputs, such that there exist honest parties that hold these inputs and send OK_1 messages.

Proof. Seeking a contradiction, assume there exist honest parties P_a, P_b, P_c that hold distinct input polynomials f_a, f_b, and f_c, such that P_a, P_b, P_c all send OK_1 messages.

Let H be the set of all honest parties and let $I = [n] \setminus H$ be the malicious parties.

For any $x \in \{a, b, c\}$ define S_x as the set of all the honest parties that agree with P_x:

$$S_x := \mathsf{A}_x^1 \cap H \ .$$

Observe that:

- For any $x \in \{a, b, c\}, |S_x| \geq n - t - |I|$. Because if P_x sent OK_1, it must agree with at least $n - t$ parties, and thus with at least $n - t - |I|$ honest parties.
- For any $x, y \in \{a, b, c\}, |S_x \cap S_y| \leq d$. Because for any $j \in S_x \cap S_y$ then $f_x(j) = f_j(j) = f_y(j)$. So if $|S_x \cap S_y| \geq d+1$ then f_x and f_y agree on at least $d+1$ points, and therefore, must be identical.

So from inclusion-exclusion:

$$\begin{aligned} |S_a \cup S_b \cup S_c| &= |S_a| + |S_b| + |S_c| - |S_a \cap S_b| - |S_a \cap S_c| - |S_b \cap S_c| + |S_a \cap S_b \cap S_c| \\ &\geq 3(n - t - |I|) - 3d + 0 \ . \end{aligned}$$

Since $d = \lfloor t/3 \rfloor$ so $3d \leq t$:

$$|S_a \cup S_b \cup S_c| \geq 3n - 3t - 3|I| - t = 3n - 4t - 3|I| \ .$$

On the other hand, $(S_a \cup S_b \cup S_c) \subseteq H$, and therefore $|S_a \cup S_b \cup S_c| \leq |H| = n - |I|$:

$$3n - 4t - 3|I| \leq |S_a \cup S_b \cup S_c| \leq n - |I| \ .$$

Using $n \geq 3t+1$ and $|I| \leq t$:

$$\begin{aligned} 2n-4t &\leq 2|I| \\ 2(3t+1)-4t \leq 2n-4t &\leq 2|I| \leq 2t \\ 2t+2 &\leq 2t\ , \end{aligned}$$

which is a contradiction. ◁

So let T_a and T_b be the at most two sets of honest parties with inputs f_a and f_b respectively such that only members in those sets send OK_1. Since $\mathsf{A}_j^2 \subseteq \mathsf{A}_j^1$ for each honest party P_j then only parties in T_a or T_b may send OK_2.

Assume wlog that $|T_a| \geq |T_b|$.

Define:

$$T_{a+} = \{i \in T_a \mid f_a(i) = f_b(i)\}\ , \qquad T_{a-} = T_a \setminus T_{a+} \quad (= \{i \in T_a \mid f_a(i) \neq f_b(i)\})\ ,$$
$$T_{b+} = \{i \in T_b \mid f_a(i) = f_b(i)\}\ , \qquad T_{b-} = T_b \setminus T_{b+} \quad (= \{i \in T_b \mid f_a(i) \neq f_b(i)\})\ .$$

▷ Claim 3.5. If $|T_a| \geq t+1$ then no party in $H \setminus T_a$ sends OK_2.

Proof. From Claim 3.4, honest parties not in $T_a \cup T_b$ do not send OK_1.

Observe that parties in T_{b-} do not send OK_1. This is because $|T_a| \geq t+1$, all $j \in T_{b-}$ have $f_a(j) \neq f_b(j)$ by definition. Therefore, any $j \in T_{b-}$ cannot receive support from T_a and thus

$$|\mathsf{A}_j^1| \leq n - |T_a| \leq n - t - 1 < n - t\ .$$

Given this, parties in in T_{b+} do not send OK_2. This is because the only honest parties that can send OK_1 that will be accepted by T_{b+} are from $T_{a+} \cup T_{b+}$, but $|T_{a+} \cup T_{b+}| \leq d$ because otherwise $f_a = f_b$. So for any party $j \in T_{b+}$ must have

$$|\mathsf{A}_j^2| \leq |T_{a+}| + |T_{b+}| + |I| \leq d + t \leq 2t < n - t\ .$$ ◁

▷ Claim 3.6. If $|T_a| \leq t$ then no party outputs grade 2.

Proof. Observe that the parties in T_{a-} and T_{b-} do not send OK_2. This is because all parties not in T_a, T_b do not send OK_1, and for $x \in \{a, b\}$, a party in T_{x^-} can receive OK_1 only from T_x and the corrupted parties. Thus,

$$|\mathsf{A}_x^2| \leq |T_x| + |I| \leq t + t = 2t < n - t\ .$$

Given this, only parties in T_{a+}, T_{b+} might send OK_2. However, no party can receive $2t+1$ OK_2 messages. This is because $|T_{a+} \cup T_{b+}| \leq d$ so

$$|\mathsf{A}_x^2| \leq |T_{a+}| + |T_{b+}| + |I| \leq d + t \leq 2t < n - t\ .$$ ◁

This concludes the proof of Theorem 3.3. ◀

Note: adding one more round. We are unaware of an application that actually requires this a property, where grade 2 implies that all parties output the same value with grade > 2 (so graded agreement not just weak graded agreement as in Definition 3.1). Nevertheless, we comment here that can be obtained by one more round.

To achieve this, parties that receive $n - t$ OK_2s send OK_3, and then: (1) if P_i sent OK_3 and received $2t+1$ OK_3 it outputs $(f(x), 2)$; (2) otherwise, if it sent OK_3 it outputs $(f(x), 1)$; (3) otherwise, (it did not send OK_3) it outputs $(\perp, 0)$.

If all parties start with the same input $f(x)$ then they all send and receive $n-t$ OK_3s and all output $(f(x), 2)$. In addition, if there is no set $|T_a| \geq t+1$, then all honest parties must output $(\perp, 0)$. In this case, Claim 3.6 shows that no party receives $2t+1$ OK_2; therefore, no honest party sends OK_3.

3.2 Building Block II: Data Dissemination

▶ **Definition 3.7.** *A protocol for parties $P_1, \ldots, P_n$ is where each party P_i has an input $f_i(x)$ (of degree d over field $\mathbb{F}$, might be $\perp$) and an output $f'_i(x)$, is* data dissemination protocol tolerating t malicious parties *if it satisfies the following property:*

- **Output consistency:** *If there exists a polynomial $f^*(x)$ such that there are at least $t+1$ honest parties with the input $f^*(x)$, and all other honest parties start with input $\perp$, then all honest parties output $f^*(x)$.*

Protocol 3.8: Data Dissemination

Input: Each party P_i holds $f_i(x)$ as input, of degree at most d. Some P_i might have input $\perp$.
The protocol:

1. **Round I – Exchange:** If $f_i(x) \neq \perp$, send to each P_j its point $f_i(j)$.
2. **Round II – Set and send your point:** Each party P_i receives messages $(u_1, \ldots, u_n)$. Let u_i be the value received at least $t+1$ times. Send u_i to all parties.

Output: Let $(y_1, \ldots, y_n)$ be the received values. Run Reed Solomon decoding RSDec on $(y_1, \ldots, y_n)$ to find a unique polynomial of degree-d $f'_i(x)$ with at most t errors. Output $f'_i(x)$. If there is no unique decoding, output $\perp$.

▶ **Theorem 3.9.** *Protocol 3.8 is a data-dissemination protocol tolerating $t < n/3$ malicious parties (as per Definition 3.7). The protocol takes two rounds of communication and $O(nL + n^2 \log n)$ communication, where each party starts with an input of size L.*

Proof. Assume that $t+1$ honest parties start with the same input $f(x)$. Each honest party P_j receives $f(j)$ at least $t+1$ times. All other honest parties that hold $\perp$ send nothing in the first message. As such, the majority value of P_j must be $f(j)$, and it sends it to all other parties in the second round. Therefore, all honest parties must receive $(y_1, \ldots, y_n)$ that is of distance at most t from $(f(1), \ldots, f(n))$. Since $n > 2t + d$, the Reed Solomon decoding guarantees that all output $f(x)$.

Protocol 3.8 as described above incurs a total communication of $O(n^2 \log n)$ bits and this corresponds to the case when each party starts with an input of size $(d+1)\log|\mathbb{F}|$ bits. Hence, the communication complexity of Protocol 3.8, where each party starts with inputs of size L bits, is $O(nL + n^2 \log n)$ bits. ◀

3.3 Application I: Gradecast

We now show how the above protocols can be utilized to achieve gradecast.

Gradecast is a relaxation of broadcast introduced by Feldman and Micali in 1988 [17]. Our protocol follows a relaxed notion of Gradecast formalized by Katz and Koo [21].

▶ **Definition 3.10.** *A protocol for parties $P_1, \ldots, P_n$, where a distinguished sender $P^* \in \{P_1, \ldots, P_n\}$ holds an initial input M is* a gradecast protocol tolerating t malicious parties *if the following conditions hold for any adversary controlling at most t parties:*

- *Each honest party P_i outputs a message m_i and a grade $g_i \in \{0, 1, 2\}$.*
- *If the sender is honest, then the output of every honest party P_i satisfies $m_i = M$ and $g_i = 2$.*
- *If there exists an honest party P_i which outputs a message m_i and the grade $g_i = 2$, then the output of every honest party P_j satisfies $m_j = m_i$ and $g_j \geq 1$.*

Protocol 3.11: Gradecast from Graded-Dispersal and Data Dissemination

Input: The sender P^* holds $f(x)$ of degree at most d. Other parties have no input.
The protocol:

1. **The sender:** The sender sends $f(x)$ to all parties.
2. **Each party P_i:** The parties invoke the graded-dispersal protocol (Protocol 3.2) where P_i inputs $f_i^{(1)}(x)$. Let $(f_i^{(2)}(x), g_i)$ be the output of the graded-dispersal protocol.
3. **Each party P_i:** The parties run the data-dissemination protocol (Protocol 3.8) with input $f_i^{(2)}(x)$. Let $f_i^{(3)}(x)$ be the output.
4. **Output:** If $g_i = 2$, then output $(f_i^{(3)}(x), 2)$. Otherwise, if $f_i^{(3)}(x) \neq \perp$, then output $(f_i^{(3)}(x), 1)$. Otherwise, output $(\perp, 0)$.

▶ **Theorem 3.12.** *Protocol 3.11 is a Gradecast protocol tolerating up to $t < n/3$ malicious parties (as per Definition 3.10). The protocol takes 5 rounds and $O(nL + n^2 \log n)$ communication for gradecasting a message of size L.*

Proof. We prove the case of an honest sender and a corrupted sender.

An honest sender. We first show that if the sender is honest and has an input $f(x)$, then all honest parties output $(f(x), 2)$. Specifically, all honest parties start the graded dispersal with the same input $f(x)$. Therefore, from the validity of graded-dispersal, all honest parties output $(f(x), 2)$. Moreover, since more than $t + 1$ honest parties hold the same input $f(x)$ in the data-dissemination protocol, all honest parties output $f(x)$ in that protocol, and output $(f(x), 2)$ in the gradecast protocol.

An honest party with output grade 2. We next consider the case where there exists an honest party P_j with an output grade 2. In that case, P_j must have grade 2 in the graded-dispersal protocol. As such, from the graded agreement, there are at least $t + 1$ honest parties that output the same polynomial $f(x)$ with grade $g_i \geq 1$ from graded dispersal. Moreover, from the properties of graded-dispersal, parties with output that is not $f(x)$ use $\perp$ as their input in the data dissemination protocol. Thus, there are at least $t + 1$ honest parties with input $f(x)$ in the data dissemination protocol, and therefore all honest parties must output $f(x)$ in the data dissemination. All honest parties output $f(x)$ with a grade of at least 1. ◀

Reducing one-round. We can reduce one round of interaction by sending the first round of the data-dissemination protocol together with the last round of the graded dispersal protocol. Specifically, each party that sends OK_2 message to P_j (the last round of the weak dispersal) sends together with it the point $f_i(j)$ to party P_j (the first message of the data dissemination).

We, therefore, get that the total round complexity of the protocol is five rounds (Theorem 3.12 is reported in that light).

Making the protocol balanced. In the protocol as described, the communication complexity of the sender is $O(n(d+1)\log n)$ while the communication complexity of every other party is just $O(n \log n)$. To reduce the communication complexity for the sender, the sender can first send to each party P_i its point $f(i)$. Each party forwards the point it received to all others, and the parties use Reed Solomon decoding to reconstruct the polynomial $f(x)$.

Stronger property. As mentioned, we use the relaxed definition of gradecast as defined in [20]. The definition by [17] requires that in the absence of a party with grade 2, all parties with grade 1 must agree on their message as well. This can be achieved at the expense of adding one round to the graded-dispersal, as discussed in Section 3.1.

3.4 Application II: Multi-Valued Byzantine Agreement

▶ **Definition 3.13** ((Binary/Multi-valued) Byzantine Agreement). *A protocol for parties $P_1, \ldots, P_n$ where each party P_i holds initial input v_i, is a* Byzantine agreement protocol tolerating t malicious parties *if the following conditions hold for any adversary controlling at most t parties:*

- **Agreement:** *All honest parties must output the same value.*
- **Validity:** *If all honest parties begin with the same input value v, then all honest parties output v.*
- **Strong consistency:** *If not all honest parties begin with the same input v, then all honest parties output the same value, which was an input of some honest party.*

For brevity, if the values $\{v_i\}$ are restricted to binary values, then we call the protocol a binary Byzantine agreement.

Protocol 3.14: Multi-Valued Byzantine Agreement from Graded Dispersal, Binary Byzantine Agreement, and Data Dissemination

- **Input:** Each party P_i holds a polynomial $f_i^{(1)}(x)$ of degree at most d over $\mathbb{F}$.
- **The protocol:**
  1. The parties invoke the graded dispersal protocol (Protocol 3.2), where each party P_i inputs $f_i^{(1)}(x)$. Let $(f_i^{(2)}(x), b_i)$ be the output.
  2. The parties run the binary Byzantine agreement protocol (e.g., [10]), where the input of each P_i is the bit b_i. Let b be the output of the Byzantine agreement.
  3. If $b = 1$, then the parties invoke the data dissemination protocol (Protocol 3.8) where each party P_i inputs $f_i^{(2)}(x)$. Let $f_i^{(3)}(x)$ be the output of P_i.
     If $b = 0$, then P_i sets $f_i^{(3)}(x) = \perp$.
- **Output:** Each P_i outputs $f_i^{(3)}(x)$.

We recall that the binary Byzantine agreement of [10] requires $O(n^2)$ communication and $\Theta(n)$ rounds for agreeing on a single bit. We get:

▶ **Theorem 3.15.** *When instantiated with a binary Byzantine agreement protocol with $O(n^2)$ communication and $\Theta(n)$ rounds ([10]), protocol 3.14 is a Byzantine Agreement protocol tolerating $t < n/3$ malicious parties (as per Definition 3.13). The protocol uses $O(nL + n^2 \log n)$ communication and $\Theta(n)$ rounds where each party starts with an input of size L.*

Proof. We first show validity, then agreement.

Validity. If all honest parties have the same input $f(x)$, then from the graded agreement property of graded dispersal (Definition 3.1), we get that all honest parties output $(f(x), 2)$. All honest parties then input 1 to the Binary Byzantine agreement, and then all honest parties receive 1 as output. The parties then invoke the data-dissemination protocol, and since there are more than $t+1$ honest parties with the input $f(x)$, all honest parties receive $f(x)$ as output and output it.

Agreement. The output of the parties is determined by the output of the binary byzantine agreement, b, which is the same for all parties. There are two cases to consider:

- If $b = 0$, then all honest parties output $\perp$, and agreement holds.
- If $b = 1$, then there must exists an honest party P_j with an input $b_j = 1$ to the Binary Byzantine Agreement. By the properties of weak-dispersal (Definition 3.1), this implies that at least $t+1$ honest parties have the same output $f(x)$ from the weak-dispersal protocol, i.e., the input of each honest party to the data-dissemination protocol (Protocol 3.8) is either $\perp$ or $f(x)$, and there are at least $t+1$ honest parties with input $f(x)$. The data-dissemination protocol then guarantees that all parties output $f(x)$. The parties output that value in our Byzantine Agreement protocol, and we again have an agreement.

Strong consistency. The only case that is interesting is when $b = 1$. The graded dispersal guarantees that output was also an input of some honest party. ◀

3.5 Application III: Broadcast

▶ **Definition 3.16** (Broadcast). *A protocol for parties $P_1, \ldots, P_n$ where a distinguished sender P^* holds an initial input M, is a* broadcast protocol tolerating t malicious parties *if the following conditions hold for any adversary controlling at most t parties:*

- **Agreement:** *All honest parties output the same value.*
- **Validity:** *If the sender is honest, then all honest parties output M.*

Protocol 3.17: Broadcast from Graded-Dispersal, Data Dissemination, and Binary Agreement

- **Input:** The sender holds a polynomial $f(x)$ of degree at most d over $\mathbb{F}$.
- **The protocol:**
  1. **The sender:** The sender sends $f(x)$ to all parties.
  2. **Each party P_i:** Let $f_i^{(1)}(x)$ be the received polynomial. The parties invoke graded-dispersal protocol (Protocol 3.2), where each party P_i inputs $f_i^{(1)}(x)$. Let $(f_i^{(2)}(x), g_i)$ be the output.
  3. **Each party P_i:** Run Binary byzantine agreement where each party P_i inputs 1 iff $g_i = 2$. Let b be the output.
  4. **Each party P_i:** If $b = 1$, then the parties run the data-dissemination protocol (Protocol 3.8), where each party P_i inputs $f_i^{(2)}(x)$. Let $f_i^{(3)}(x)$ be the output of the data dissemination protocol.
- **Output:** If $b = 1$, then output $f_i^{(3)}(x)$. If $b = 0$ then output $\perp$.

▶ **Theorem 3.18.** *Protocol 3.17 is a broadcast protocol tolerating $t < n/3$ malicious parties (as per Definition 3.16). The protocol uses $\Theta(n)$ rounds and $O(nL + n^2 \log n)$ communication for broadcasting a message of size L.*

Proof. We show agreement and validity.

Agreement. By the binary byzantine agreement, all parties receive the same bit b. If $b = 0$, then all honest parties output $\perp$ and agreement holds; If $b = 1$, then it must be that some honest party received grade 2 in the graded dispersal. As follows from the guarantees of graded dispersal, this implies that there are at least $t + 1$ honest parties with the same output $f(x)$. Those honest parties input $f(x)$ to the data dissemination, and the output of all honest parties is then $f(x)$. Agreement again holds.

Validity. If the sender is honest and starts with input $f(x)$ of degree-d over $\mathbb{F}$, then from the guarantees of graded-dispersal, all honest parties output $(f(x), 2)$. As such, all honest parties input 1 to the binary byzantine agremeent, and must agree on $b = 1$. Therefore, all run the data-dissemination protocol, and all output $f(x)$. ◀

Making the protocol balanced. In the broadcast protocol as described in Protocol 3.17, the communication complexity of the sender is $n(d + 1) \log n$ bits, whereas the communication complexity of every other party is just $O(n \log n)$ bits. As before, to reduce the communication complexity for the sender, the sender can first send to each party P_i its point $f(i)$. The parties then forward the received point to all others and use Reed Solomon decoding to reconstruct the polynomial $f(x)$.

4 Reliable Broadcast in Asynchrony

We now move to asynchronous communication. Here, when an honest party sends a message it is guaranteed to be delivered, but the adversary can introduce arbitrarily delays.

4.1 Dispersal

▶ **Definition 4.1** (Dispersal). *A protocol for parties $P_1, \ldots, P_n$ where the input of each party P_i is some $v_i \in \mathbb{F}$, is an* asynchronous dispersal protocol tolerating t corrupted parties *if the following properties hold:*

- **Termination:** *If one honest party terminates, then all honest parties terminate.*
- **Weak agreement:** *If one honest party terminates with output $f(x)$, then at least $t + 1$ other honest parties output also $f(x)$. Furthermore, all honest parties that terminate with a non-$\perp$ output, terminate with the same polynomial $f(x)$.*
- **Weak Validity:** *If all honest parties start with the same polynomial $f(x)$ of degree at most d, then termination and weak agreement hold with respect to $f(x)$.*

Our asynchronous dispersal protocol is fairly identical to the synchronous counterpart. However, when executed under asynchrony, a key distinction is in the construction of the dynamic sets, $\mathsf{A}_i^1, \mathsf{A}_i^2$. As the adversary may introduce arbitrary delays, the OK_2 messages may not be delivered after the OK_1 messages. The parties construct the A_i^1 and A_i^2 sets and update them in an online fashion upon receiving any message (OK_1 or OK_2).

Protocol 4.2: Dispersal

Input: Each party P_i holds $f_i(x)$ of degree at most d over $\mathbb{F}$.
The protocol:

1. Initialization: Each party initializes $S_i = \emptyset, \mathsf{A}_i^1 = \mathsf{A}_i^2 = \emptyset$, and $f_i(x) = \perp$.
2. **Exchange:**
   a. P_i sends $(\texttt{Exchange}, f_i(i), f_i(j))$ to each P_j.
3. **Dynamic set A_i^1:**
   a. **Upon** receiving $(\texttt{Exchange}, u_j, v_j)$ from P_j, if $f_i(j) = u_j$ and $f_i(i) = v_j$ then add j to A_i^1 .
   b. **Upon** $|\mathsf{A}_i^1| \geq n - t$, send OK_1 to all.
4. **Dynamic set A_i^2:**
   a. **Upon** receiving message OK_1 from P_j for which $j \in \mathsf{A}_i^1$, then add j to A_i^2.
   b. **Upon** $|\mathsf{A}_i^2| \geq n - t$, send OK_2 to all.
5. **Sending** `Done`**:**
   a. If P_i sent OK_2, and **upon** receiving $2t + 1$ OK_2 messages, send `Done` message to everyone.
   b. **Upon** receiving $t + 1$ `Done` messages from distinct parties, send `Done` to everyone.
   c. **Upon** receiving $2t + 1$ `Done` messages, if P_i sent OK_2 message, then terminate and output $f_i(x)$. If P_i did not send OK_2 message, then terminate and output $\perp$.

▶ **Theorem 4.3.** *Protocol 4.2 is an asynchronous dispersal protocol tolerating $t < n/3$ malicious parties (as per Definition 4.1). The protocol takes 4 rounds and a total communication of $O(nL + n^2 \log n)$ bits where each party starts with an input of size L.*

Proof. We show each one of the properties separately.

Termination. An honest party terminates only after it receives $2t + 1$ `Done` messages. This implies that it received at least $t + 1$ messages `Done` from honest parties (and in particular, it also sent the `Done` message). Eventually, all honest parties will receive those $t + 1$ `Done` messages and, in response, will also send a `Done` message. We obtain that eventually, all honest parties send `Done` message, and therefore each honest party will eventually receive $2t + 1$ `Done` messages, regardless of the messages of the adversary. Therefore, all honest parties must terminate.

Weak agreement. An honest party that terminates with an output $f^*(x) \neq \perp$ must have sent an OK_2 message. Moreover, since it terminated, it must have received at least $2t + 1$ `Done` messages, i.e., it received `Done` messages from at least $t + 1$ honest parties. An honest party sends a `Done` message if it either received $2t + 1$ OK_2 messages or received $t + 1$ `Done` messages. The latter implies that at least one honest party sent a `Done` message due to receiving $2t+1$ OK_2 messages, and at least $t+1$ honest parties sent OK_2 message. As follows from Claim 3.4, there are at most two sets of parties (with the same polynomial) that could have sent OK_1. We have two cases to consider:

- As follows from Claim 3.6, if there is no large set of $t+1$ parties with the same polynomial, then no party receives $2t + 1$ OK_2. This implies that no honest party would have sent the `Done` message, and no party terminates.
- If there is a large set of $t+1$ parties with the same polynomial $f^*(x)$, then no other party sends OK_2.

This implies that if some honest party terminates, then we must have a large set of $t+1$ parties with the same polynomial $f^*(x)$. This implies that at least $t+1$ honest parties terminate with that polynomial $f^*(x)$. Any other honest party that terminates with non-$\perp$ must also output $f^*(x)$.

Weak Validity. If all honest parties start with the same polynomial $f^*(x)$, then eventually, all honest parties will appear in the agreed sets of one another. As such, eventually, all honest parties will send the OK_2 message, send `Done` messages, and terminate. Parties might terminate earlier; as mentioned, if one honest party terminates, it outputs its polynomial, and we are also guaranteed that at least $t+1$ other honest parties terminate with their input polynomials. Therefore, weak validity holds. ◀

4.2 Asynchronous Data Dissemination

▶ **Definition 4.4** (Asynchronous Data Dissemination). *A protocol for parties $P_1, \ldots, P_n$ where each P_i has input $f_i(x)$ (of degree d over $\mathbb{F}$, might be $\perp$) is an* asynchronous data dissemination protocol tolerating t malicious parties *if it satisfies the following property:*

- **Agreement and termination:** *If there exists a polynomial $f(x)$ such that at least $t+1$ honest parties start with the input $f(x)$, and all other honest parties start with input $\perp$, then all honest parties terminate and output $f(x)$.*

Protocol 4.5: Asynchronous Data Dissemination

Input: Each party P_i holds $f_i(x)$ as input, of degree at most d. Some P_i might have input $\perp$.
The protocol:

1. Initialize a multi-set $M_i = \emptyset$ and $S_i = \emptyset$. If $f_i(x) \neq \perp$, send to each P_j its point $(\texttt{YourPoint}, f_i(j))$.
2. **Upon** receiving $(\texttt{YourPoint}, u_j)$ from P_j, add u_j to M_i.
3. **Upon** some u_i appearing $t+1$ times in M_i, send $(\texttt{MyPoint}, u_i)$ to all parties.
4. **Upon** receiving $(\texttt{MyPoint}, u_j)$ from P_j, add (j, u_j) to S_i. **Upon** $|S_i| \geq d+t+1$ execute the following:
   a. Run $\mathsf{RSDec}(S_i)$ and try to decode a polynomial $f_i(x)$ of degree at most d that agrees with S_i on at at least $d+t+1$ values.
   b. If no such polynomial exists, then wait to receive more points in S_i and retry.
   c. If such a polynomial $f_i(x)$ is computed, set $f_i(x)$ to be the resultant value
5. **Upon** unique decoding of $f_i(x)$, terminate and output $f_i(x)$.

▶ **Theorem 4.6.** *Protocol 4.5 is an asynchronous data-dissemination protocol tolerating $t < n/3$ malicious parties. The protocol takes 2 rounds and $O(nL + n^2 \log n)$ communication, where each party starts with an input of size L.*

Proof. Assuming that $t+1$ honest parties hold the input $f(x)$ and the others hold $\perp$, we get that each honest party P_j eventually receives $f(j)$ at least $t+1$ times and therefore sends to everyone the message $(\texttt{MyInput}, f(j))$. As a result, all honest parties eventually send that message. Therefore, all honest parties will eventually succeed in their Reed Solomon decoding and terminate with the polynomial $f(x)$. ◀

4.3 Reliable Broadcast

▶ **Definition 4.7** (Reliable Broadcast). *A protocol for parties $P_1, \ldots, P_n$ where a distinguished party P^* has an input $f(x)$ of degree at most d over $\mathbb{F}$ is a* reliable broadcast tolerating t malicious parties *if the following properties hold:*

- **Validity:** *If the sender P^* is honest then all honest parties terminate and output $f(x)$.*
- **Agreement:** *If two honest parties terminate, their output is the same.*
- **Termination:** *If an honest party terminates, then all honest parties will eventually terminate.*

Protocol 4.8: Reliable Broadcast

Input: The sender holds a polynomial $f(x)$ of degree at most d.
The protocol:

1. **The sender:** Send $f(x)$ to each party P_i.
2. **Each party P_i: Upon** receiving $f_i^{(1)}(x)$ from the sender, participate in Dispersal (Protocol 4.2) with input $f_i^{(1)}(x)$.
3. **Each party P_i: Upon** receiving an output $f_i^{(2)}(x)$ from the dispersal protocol, participate in asynchronous data dissemination protocol (Protocol 4.5) with input $f_i^{(2)}(x)$.
4. **Upon** receiving an output $f_i^{(3)}(x)$ from the data dissemination protocol, terminate and output $f_i^{(3)}(x)$.

▶ **Theorem 4.9.** *Protocol 4.8 is a reliable broadcast protocol tolerating $t < n/3$ malicious parties (as per Definition 4.7). The protocol takes 6 rounds and $O(nL + n^2 \log n)$ bits for reliably broadcasting a message of size L.*

Proof. We show each one of the properties separately.

Validity. If the sender is honest and starts with the input $f(x)$, then eventually all honest parties start the Dispersal protocol with the input $f(x)$. From the weak-validity property of the dispersal protocol (see Definition 4.1), all honest parties terminate with an output which is in $\{\perp, f(x)\}$, and at least $t+1$ honest parties terminate with $f(x)$. All honest parties then eventually continue to the asynchronous data dissemination. According to that protocol, if at least $t+1$ honest parties start with the same input $f(x)$, and all other honest parties start with $\perp$, then all honest parties must terminate, and with output $f(x)$. Therefore, all honest parties eventually terminate the reliable broadcast protocol, with an output $f(x)$.

Agreement and termination. Assume that some honest party terminates. This implies that both the dispersal and the data dissemination protocols must terminate (an honest party does not participate in the data dissemination before terminating the dispersal). If the dispersal protocol terminates, then from its security properties, we must have that at least $t+1$ honest parties terminate with the same polynomial $f(x)$, and all the other honest parties terminate with an output $\perp$. The parties then input those inputs to the data-dissemination protocol, which guarantees that under this exact input – all honest parties terminate with the same output $f(x)$. We conclude that all honest parties terminate with the same output $f(x)$. ◀

Reducing one-round. As in the case of synchrony, we can reduce one round of interaction by sending the first round of the data-dissemination protocol together with the last round of the data-dispersal protocol. Specifically, each party that sends `Done` message to P_j (the last round of the weak dispersal) sends, together with it, the point $f_i(j)$ to party P_j (the first message of the data dissemination).

Making the protocol balanced. In the reliable broadcast protocol as described in Protocol 4.8, the communication complexity of the sender is $n(d+1)\log n$ bits, whereas the communication complexity of every other party is just $O(n \log n)$ bits. As in the case of synchrony, to balance the communication complexity of the sender, the sender can first send to each party P_i its point $f(i)$. The parties then send it to all other parties and execute Reed Solomon decoding repeatedly to reconstruct the polynomial $f(x)$. Specifically, each party waits to receive at least $d+t+1$ points and executes Reed Solomon decoding to recover a polynomial that agrees with at least $d+t+1$ points.

References

1 Ittai Abraham and Gilad Asharov. Gradecast in synchrony and reliable broadcast in asynchrony with optimal resilience, efficiency, and unconditional security. In *PODC '22: ACM Symposium on Principles of Distributed Computing, 2022*, pages 392–398. ACM, 2022. doi:10.1145/3519270.3538451.

2 Ittai Abraham, Gilad Asharov, Shravani Patil, and Arpita Patra. Asymptotically free broadcast in constant expected time via packed VSS. In *Theory of Cryptography - 20th International Conference, TCC 2022*, volume 13747 of *Lecture Notes in Computer Science*, pages 384–414. Springer, 2022. doi:10.1007/978-3-031-22318-1_14.

3 Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. *Distributed Comput.*, 36(1):3–28, 2023. doi:10.1007/S00446-022-00428-8.

4 Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *PODC '21: ACM Symposium on Principles of Distributed Computing, 2021*, pages 363–373. ACM, 2021. doi:10.1145/3465084.3467914.

5 Ittai Abraham and Andrew Lewis-Pye. Phase-king through the lens of gradecast: A simple unauthenticated synchronous byzantine agreement protocol. *Decentralized Thoughts, Blog Post*, 2022. URL: https://decentralizedthoughts.github.io/2022-06-09-phase-king-via-gradecast/.

6 Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *PODC '22: ACM Symposium on Principles of Distributed Computing*, pages 399–417. ACM, 2022. doi:10.1145/3519270.3538475.

7 Gilad Asharov and Anirudh Chandramouli. Perfect (parallel) broadcast in constant expected rounds via statistical VSS. In *Advances in Cryptology - EUROCRYPT 2024*, volume 14655 of *Lecture Notes in Computer Science*, pages 310–339. Springer, 2024. doi:10.1007/978-3-031-58740-5_11.

8 Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proc. of the Annual Symposium on Principles of Distributed Computing (PODC)*, 1983.

9 Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Brief announcement: Simple gradecast based algorithms. In *Distributed Computing, 24th International Symposium, DISC 2010.*, volume 6343 of *Lecture Notes in Computer Science*, pages 194–197. Springer, 2010. doi:10.1007/978-3-642-15763-9_18.

10 Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science*. 1992.

11 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987. doi:10.1016/0890-5401(87)90054-X.

12 Christian Cachin and Stefano Tessaro. Asynchronous veri.able information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS 2005), 2005*, pages 191–202. IEEE Computer Society, 2005. doi:10.1109/RELDIS.2005.9.

13 Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *DISC 2021*, volume 209 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.DISC.2021.17.

14 Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721. ACM, 2021. doi:10.1145/3460120.3484808.

15 Sourav Das, Zhuolun Xiang, and Ling Ren. Balanced quadratic reliable broadcast and improved asynchronous verifiable information dispersal. *IACR Cryptol. ePrint Arch.*, page 52, 2022. URL: https://eprint.iacr.org/2022/052.

16 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, 1982*, pages 132–140. ACM, 1982. doi:10.1145/800220.806690.

17 Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 148–161. ACM, 1988. doi:10.1145/62212.62225.

18 Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 1982.

19 Chaya Ganesh and Arpita Patra. Optimal extension protocols for byzantine broadcast and agreement. *Distributed Comput.*, 34(1):59–77, 2021. doi:10.1007/S00446-020-00384-1.

20 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference*, 2006.

21 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, 2009. doi:10.1016/J.JCSS.2008.08.001.

22 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

23 Guanfeng Liang and Nitin H. Vaidya. Error-free multi-valued consensus with byzantine failures. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011*, pages 11–20. ACM, 2011. doi:10.1145/1993806.1993809.

24 Andrew D. Loveless, Ronald G. Dreslinski, and Baris Kasikci. Optimal and error-free multi-valued byzantine consensus through parallel execution. *IACR Cryptol. ePrint Arch.*, page 322, 2020. URL: https://eprint.iacr.org/2020/322.

25 Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *34th International Symposium on Distributed Computing, DISC 2020*, volume 179 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.DISC.2020.28.

26 Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011*, volume 7109 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2011. doi:10.1007/978-3-642-25873-2_4.

27 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

28 M. O. Rabin. Randomized byzantine generals. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 1983.

29 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

30 Jianjun Zhu, Fan Li, and Jinyuan Chen. Communication-efficient and error-free gradecast with optimal resilience. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 108–113, 2023. doi:10.1109/ISIT54713.2023.10206579.

31 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.