Communication and Round Efficient Parallel Broadcast Protocols

Ittai Abraham[1], Kartik Nayak[2], and Nibesh Shrestha*[3]

[1]Intel Labs, ittai.abraham@intel.com
[2]Duke University, kartik@cs.duke.edu
[3]Supra Research, nibeshrestha2@gmail.com

Abstract

This work focuses on the *parallel broadcast* primitive, where each of the n parties wish to broadcast their ℓ-bit input in parallel. We consider the *authenticated* model with PKI and digital signatures that is secure against $t < n/2$ Byzantine faults under a *synchronous* network.

We show a generic reduction from parallel broadcast to a new primitive called graded parallel broadcast and a single instance of validated Byzantine agreement. Using our reduction, we obtain parallel broadcast protocols with $O(n^2\ell + \kappa n^3)$ communication (κ denotes a security parameter) and expected constant rounds. Thus, for inputs of size $\ell = \Omega(n)$ bits, our protocols are asymptotically free.

Our graded parallel broadcast uses a novel gradecast protocol with multiple grades with asymptotically optimal communication complexity of $O(n\ell + \kappa n^2)$ for inputs of size ℓ bits. We also present a multi-valued validated Byzantine agreement protocol with asymptotically optimal communication complexity of $O(n\ell + \kappa n^2)$ for inputs of size ℓ bits in expectation and expected constant rounds. Both of these primitives are of independent interest.

1 Introduction

Parallel broadcast is a primitive where all parties wish to broadcast ℓ bit messages in parallel. This is an essential building block, central to many cryptographic protocols like verifiable secret sharing (VSS), multi-party computation (MPC) [6, 8, 2], distributed key generation (DKG) [14] where all parties broadcast messages in parallel in the same round. For example, in MPC and DKG applications, each party broadcasts $O(1)$ VSSs in parallel to share secrets. In VSS itself, each party broadcasts a commitment [17] to a secret in parallel. Design of efficient protocols for parallel broadcast is therefore of paramount importance as any improvements for parallel broadcast also results in improvement of these primitives. In this work, we focus on improving the communication complexity (i.e., reducing the number of bits honest parties exchange) and the round complexity (i.e., the time required to reach a decision) of parallel broadcast in the synchronous authenticated model with PKI and digital signatures tolerating $t < n/2$ Byzantine failures under various setup assumptions.

While existing works on parallel broadcast can tolerate a dishonest Byzantine majority, they either naïvely run n instances of Byzantine agreement (or Byzantine broadcast) primitives (increasing the communication complexity by an undesirable factor of n) [7, 1, 16] or incur a high round complexity along with strong cryptographic assumptions [33, 28]. For example, the broadcast extension protocol[1] due to Nayak et al. [28] can be used to achieve a parallel broadcast protocol tolerating $t < (1-\varepsilon)n$ Byzantine faults (where $\varepsilon > 0$ is a small constant) with a communication complexity of $O(n^2\ell + \kappa n^3 + n^4)$ (where κ is a security parameter) to propagate ℓ bit input and $O(t)$ round complexity. Similarly, protocols due to Tsimos et al. [33] tolerate $t < (1-\varepsilon)n$ Byzantine faults with a communication complexity of $O(\kappa^2 n^3 \ell)$ and $O(t \log t)$ round complexity.

*Lead author. Work done while the author was a student at Rochester Institute of Technology.

[1]By extension protocol, we mean reduction of reduction of BA/BB on long input to a BA/BB on a smaller input [28]

Table 1: **Comparison of related works on MVBA with ℓ-bit input**

		Network	Resilience	Communication	Latency	Adversary	Assumption
Cachin et al.	[13]	async.	$t < n/3$	$O(n^2\ell) + E(O(\kappa n^2 + n^3))$	$E(O(1))$	adaptive	threshold sigs.
VABA	[5]	async.	$t < n/3$	$O(n^2\ell) + E(O(\kappa n^2))$	$E(O(1))$	adaptive	threshold sigs.
DUMBO-MVBA	[26]	async.	$t < n/3$	$E(O(n\ell + \kappa n^2))$	$E(O(1))$	adaptive	threshold sigs.
Shrestha et al.	[32]	sync.	$t < n/2$	$E(O(n^2\ell + \kappa n^3))$	$E(O(1))$	static	PKI
This work		sync.	$t < n/2$	$E(O(n\ell + \kappa n^2))$	$E(O(1))$	adaptive	threshold sigs.

$E(.)$ implies "in expectation".

Thus, this work investigates the communication complexity and round complexity of parallel broadcast protocol when the fault tolerance is $t < n/2$. To be specific, we ask the following question:

> *Can we design a parallel broadcast protocol with a good communication complexity and a good round complexity while tolerating $t < n/2$ Byzantine faults?*

We answer this question affirmatively by showing two parallel broadcast protocols each with $O(n^2\ell + \kappa n^3)$ communication for inputs of size ℓ bits and termination in constant expected rounds. Thus, for inputs for size $\ell = \Omega(n)$ bits, our protocols have no asymptotic overhead. Our first protocol works in the authenticated model with PKI and digital signatures and is secure against a static adversary. Our second protocol relies on threshold setup assumption to obtain security against a (strongly rushing) adaptive adversary.

1.1 Key Technical Ideas and Results

Parallel broadcast is a primitive where all parties wish to broadcast ℓ bit messages in parallel [30]. It can be implemented naïvely by invoking n instances of Byzantine broadcast [16] or Byzantine agreement [4] primitives in parallel in a black box manner. However, this technique increases the communication complexity by an undesirable factor of n. Moreover, invoking n concurrent instances of randomized Byzantine agreement protocol [4] (that terminates in expected $O(1)$ rounds) terminates in expected $O(\log n)$ rounds [7]; thus increasing the round complexity. Our work focuses on improving communication complexity while keeping a constant expected round complexity.

Towards communication efficient parallel broadcast. Instead of relying on n instances of expensive Byzantine Broadcast (or Byzantine Agreement) primitive, we obtain parallel broadcast using a combination of n instances of a gradecast primitive [23, 32] and only one instance of (validated) agreement protocol. To ensure an overall communication complexity of $O(n^2\ell + \kappa n^3)$ for inputs of size ℓ bits, we improve the communication complexity of gradecast to $O(n\ell + \kappa n^2)$ and the validated agreement protocol to $O(n\ell + \kappa n^2)$ in expectation. In the following, we will first describe our improvements to each of the primitives, before describing parallel broadcast.

Gradecast with multiple grades. Gradecast is a relaxed version of broadcast introduced by Feldman and Micali [18] where parties output a value along with a grade. Basic versions of gradecast [23, 32] have grades in the range of $\{0, 1, 2\}$. We rely on a version of gradecast that supports grades in the range $\{0, 1, 2, 3, 4\}$ (we will explain later the need for this version of gradecast). At a high level, our gradecast with multiple grades provides the following guarantees: (i) the grades of all honest parties are maximum i.e., 4 when the sender is honest, (ii) honest parties may output different grades when the sender is Byzantine; but the grades of any two honest parties differ by at most 1, (iii) when an honest party outputs a value with a grade of 2, all honest parties output the same value with grade of ≥ 1, (iv) two honest parties may output different values with a grade of 1 when no honest party has a grade of 2. While gradecast with grades up to 4 suffices for our purpose, we generalize it to support arbitrary number of grades $\{0, 1, \ldots, g^*\}$ where g^* is the maximum supported grade.

We give a construction tolerating $t < (1-\varepsilon)n$ Byzantine faults with a communication complexity of $O(n\frac{\ell}{\varepsilon} + \kappa n^2)$. The key technique we employ to design communication efficient gradecast is to have parties multicast smaller chunks of messages (via Reed-Solomon erasure codes [31]) only once and then "silently" waiting to detect any conflicting messages while simultaneously increasing the grades when no conflicting messages are detected. In particular, we obtain the following result:

Theorem 1. *Assuming a public-key infrastructure, digital signatures and a universal structured reference string under q-SDH assumption, there exists a g^*-gradecast protocol tolerating $t < (1-\varepsilon)n$ Byzantine faults with a communication complexity of $O(n\frac{\ell}{\varepsilon}+\kappa n^2)$ for an input of size ℓ bits and a round complexity of $3g^*-2$.*

When $\varepsilon > 0$ is a small constant, our protocol has a communication complexity of $O(n\ell + \kappa n^2)$. This communication complexity is achieved when we assume q-Strong Diffe Hellman (q-SDH) [11] setup assumption (this setup can be achieved via distributed protocols [9, 12]). We can alternatively make use of Merkle tree [27] to avoid q-SDH assumption at the expense of $O(\log n)$ multiplication communication complexity.

Graded parallel broadcast: Composing n instances of gradecast with multiple grades and ensuring validated output. Parties invoke gradecast with multiple grades with each party as a sender to propagate their input and output an n-element list of grades (GradeList) corresponding to each party as sender. When the sender is honest, all honest parties output a common value with the highest possible grade. When the sender is Byzantine, honest parties may output different values with different grades. Thus, GradeList of two honest parties may be different; especially the grades corresponding to Byzantine senders.

Looking ahead, our aim is to agree on a common GradeList using a validated Byzantine agreement protocol and compute the final output vector based on the grades in the agreed GradeList to solve the parallel broadcast problem. The agreed GradeList can be the input of even a Byzantine party who may set arbitrary grades in its GradeList. In order to restrict a Byzantine party from setting arbitrary grades in its GradeList, we define the notion of *valid* GradeList and ensure the validated Byzantine agreement protocol outputs only valid GradeList. A *valid* GradeList is one that has been verified by at least one honest party. An honest party verifies a given GradeList by checking against its own GradeList and ensuring that the grades corresponding to a sender differ by at most 1. This restricts a Byzantine party to set arbitrary grades in a *valid* GradeList.

Given this notion of valid GradeList, let us see why we need a gradecast that supports grades in the range $\{0, 1, 2, 3, 4\}$ where honest parties output a common value with the highest grade of 4 when the sender is honest. Consider a Byzantine party who may set arbitrary grades in its GradeList. For its GradeList to be valid, it must set a grade of at least 3 corresponding to honest senders for its GradeList to be verified by an honest party (since gradecast ensures that the grades output by any two parties differ by at most 1). Then we can compute the final output vector (to solve parallel broadcast) by considering values that have grades at least 3 in the agreed valid GradeList. This ensures honest inputs are always included in the final output vector. Note that the Byzantine party may also set a grade of at least 3 corresponding to a Byzantine sender in its GradeList. The GradeList will be verified as long as one honest party has a grade of at least 2 corresponding to this Byzantine sender. Note that our gradecast protocol ensures that all honest parties have output the same value when an honest party sets a grade of at least 2. This ensures consistency in the final output vector.

To see why gradecast protocol that supports fewer grades does not work, let us consider a gradecast where the maximum grade is 3. We consider a GradeList of a Byzantine party who may set a grade of 2 corresponding to an honest sender (to ensure the GradeList is valid). In this version, in order to ensure honest inputs are included in the final vector, we need to output values with grades of at least 2 in the agreed GradeList. However, the Byzantine party may also set a grade of 2 corresponding to Byzantine sender for which no honest party has a grade of 2; different honest parties may have different values in this case. Thus, this violates consistency.

We formally define the process of invoking n parallel instances of gradecast with multiple grades and obtaining (possibly different) valid GradeList as *graded parallel broadcast*. We obtain the following result,

Theorem 2. *Assuming a public-key infrastructure, digital signatures and a universal structured reference string under q-SDH assumption, there exists a graded parallel broadcast protocol tolerating $t < n/2$ Byzantine faults with $O(n^2\ell + \kappa n^3)$ communication for an input of size ℓ bits and constant rounds.*

Agreeing on a common valid GradeList using efficient multi-value validated Byzantine agreement. We make use of a single instance of multi-valued validated Byzantine agreement (MVBA) to agree on a common GradeList. In MVBA, each party starts with a different externally valid input (possibly large) and outputs a common value; the output value can be input of any party as long as it is externally valid. The best-known prior work is the MVBA protocol of Shrestha et al. [32] which works in the authenticated model with PKI and digital signatures. Their protocol is secure against a static adversary tolerating $t < n/2$ faults with $O(n^2\ell + \kappa n^3)$ communication in expectation and expected $O(1)$ rounds.

In order to improve communication complexity and provide adaptive security, we design an MVBA protocol secure against a (strongly rushing) adaptive adversary tolerating $t < n/2$ Byzantine faults. Our MVBA protocol incurs $O(n\ell + \kappa n^2)$ communication in expectation and terminates in expected constant rounds but assumes threshold setup and relies on adaptively-secure threshold signature scheme [25]. Following the communication lower bound results of Abraham et al. [3] and Fitzi et al. [20], our MVBA protocol has asymptotically optimal communication complexity. Specifically, we show the following result:

Theorem 3. *Assuming a public-key infrastructure, digital signatures, threshold setup and a universal structured reference string under q-SDH assumption, there exists a multi-valued validated Byzantine agreement protocol tolerating $t < n/2$ Byzantine faults with $O(n\ell+\kappa n^2)$ communication in expectation for inputs of size ℓ bits, termination in expected $O(1)$ rounds and security against a (strongly rushing) adaptive adversary.*

The starting point of our MVBA construction is the Byzantine synod protocol of Abraham et al. [4] which is secure against a (strongly rushing) adaptive adversary and incurs $O((\ell + \kappa)n^2)$ communication in expectation and terminates in expected constant rounds. We present a brief overview of their protocol to understand $O(n^2\ell)$ term.

In their protocol, parties first multicast their ℓ-bit proposals and collect acknowledgements from at least $t + 1$ parties. A proposal is said to be "prepared" if it collects acknowledgements from $t + 1$ parties. Each of the parties then proposes these prepared proposals. This is followed by a leader election phase where they always obtain a common leader. With probability at least 1/2, this leader is honest. Once the leader is elected, parties only consider the prepared proposal of the leader. If such a proposal exists and there are no equivocating prepared proposals from the leader, the proposal is committed; otherwise parties perform a "view-change" to restart the process. Having parties create prepared proposals before a leader election prevents an adaptive adversary from corrupting the elected party and creating equivocating proposals. For proposals of size ℓ bits each, a multicast of n proposals trivially incurs $O(n^2\ell)$ communication as each party needs to receive $n\ell$ bits.

Our MVBA protocol inherits the underlying consensus mechanism of their protocol and improves the dissemination of the proposals to obtain $O(n\ell + \kappa n^2)$ communication. Our solution uses Reed-Solomon erasure codes [31] to encode large messages into n code words and cryptographic accumulators [29] to verify the correctness of the code words.

In our protocol, each party P_i encodes its ℓ bit proposal to n code words $(s_{i,1}, \ldots, s_{i,n})$ via Reed-Solomon erasure codes and sends a code word $s_{i,j}$ to party P_j $\forall j \in [n]$ along with a cryptographic witness to verify the correctness of the code word $s_{i,j}$. Each party P_j, upon receiving a valid code word $s_{i,j}$, sends an acknowledgment to party P_i. Party P_i considers its proposal "prepared" once it receives $t + 1$ acknowledgments. We stress that a party receives a single valid code word corresponding to the proposal; and not the full proposal. This differs from extension techniques [28] where all parties receive the full proposal. For all n proposals of size ℓ bits each, this process only incurs $O(n\ell + \kappa n^2)$ communication. Having proposals prepared in this manner still gives that same advantages against an adaptive adversary with reduced communication.

Later in the protocol, when the prepared proposal is selected during leader election phase, the full proposal needs to be retrieved before committing it. An original proposal can be decoded with $t+1$ valid code words for the proposal. Note, however that a "prepared" proposal does not imply sufficient code words required to decode the proposal will be available. A Byzantine party may send a valid code word corresponding to its proposal to a single honest party and collect t acknowledgements from Byzantine parties to have its proposal prepared. Thus, having a proposal prepared does not guarantee its availability. We consider such proposals

Table 2: **Comparison of related parallel broadcast protocols**

	Model	Resilience	Communication		Latency	Adversary
			Expected	Worst		
Nayak et al.[28]	PKI	$t < (1-\varepsilon)n$	$O(n^2\ell + \kappa n^3 + n^4)^*$		$O(t)$	adaptive
Tsimos et al. [33]	PKI	$t < (1-\varepsilon)n$	$\tilde{O}(\kappa^2 n^3 \ell)^*$		$O(t \log t)$	adaptive
Tsimos et al. [33]	trusted PKI	$t < (1-\varepsilon)n$	$\tilde{O}(\kappa^4 n^2 \ell)^*$		$O(\kappa \log t)$	adaptive
Abraham et al. [1]	unauthenticated	$t < n/3$	$O(n^2\ell) + E(O(n^4 \log n))$	$O(n^2\ell + n^5 \log n)$	$E(O(1))$	static
Nayak et al. [28] + [4]	threshold sigs.	$t < n/2$	$O(n^2\ell) + E(O(\kappa n^3))$	$O(n^2\ell + \kappa n^4)$	$E(\log n)$	adaptive
This work + [32]	PKI	$t < n/2$	$O(n^2\ell) + E(O(\kappa n^3))$	$O(n^2\ell + \kappa n^4)$	$E(O(1))$	static
This work	threshold sigs.	$t < n/2$	$O(n^2\ell + \kappa n^3) + E(O(\kappa n^2))$	$O(n^2\ell + \kappa n^3)$	$E(O(1))$	adaptive

Tsimos et al. [33] and Abraham et al. [1] do not assume q-SDH assumption. Tsimos et al. [33] has $\tilde{O}$ in the communication complexity which hides a $\log n$ factor unrelated to the q-SDH assumption. Without q-SDH setup assumption, our protocols would have $\log n$ multiplicative factor in the communication complexity. $E(.)$ implies "in expectation". * This is the best communication complexity as these protocols execute for a fixed number of rounds.

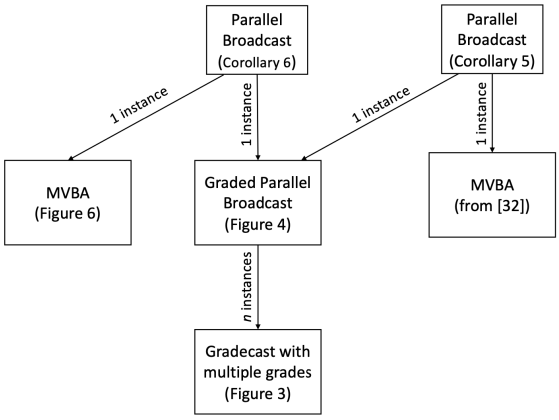

Figure 1: Overview of sub-protocols and their dependencies

as "bad". When such a bad proposal is selected during the leader election phase, we "wait" for a few rounds to detect recoverability of the proposal and perform view-change when we are unable to decode the selected proposal, i.e., we rely on synchrony to detect and filter out bad proposals. Once an honest leader is elected, its prepared proposal can be decoded and committed.

Efficient parallel broadcast. Finally, we obtain efficient protocols for parallel broadcast using the above primitives. In particular, we use the graded parallel broadcast and MVBA protocol to achieve parallel broadcast protocol. Specifically, we obtain the following main result:

Theorem 4. *Assuming a public key infrastructure and digital signatures, if we have a graded parallel broadcast tolerating $t < n/2$ Byzantine faults with a communication complexity of x and round complexity of y, and a MVBA protocol tolerating $t < n/2$ Byzantine faults with a communication complexity of a and a round complexity of b, we can have a parallel broadcast protocol tolerating $t < n/2$ Byzantine faults with a communication complexity of $x + a$ and a round complexity of $y + b$.*

We obtain different results for parallel broadcast depending on the variant of the validated Byzantine agreement used. Our first parallel broadcast protocol uses the MVBA protocol of Shrestha et al. [32] which is a secure against a static adversary with $O(n^2\ell + \kappa n^3)$ communication in expectation and expected $O(1)$ rounds. Using their MVBA protocol, we obtain the following corollary:

Corollary 5. *Assuming a public-key infrastructure, digital signatures, and a universal structured reference*

string under q-SDH assumption there exists a protocol secure against static adversary that solves parallel broadcast tolerating $t < n/2$ Byzantine faults with $O(n^2\ell) + E(O(\kappa n^3))$ communication and expected $O(1)$ rounds.

Our second parallel broadcast protocol uses our MVBA protocol (Theorem 3). We obtain the following corollary:

Corollary 6. *Assuming a public-key infrastructure, digital signatures, threshold setup, and a universal structured reference string under q-SDH assumption there exists a protocol that solves parallel broadcast tolerating $t < n/2$ Byzantine faults with $O(n^2\ell + \kappa n^3) + E(O(\kappa n^2))$ communication, termination in expected $O(1)$ rounds and security against a (strongly rushing) adaptive adversary.*

Observe that our second parallel broadcast has $O(n^2\ell+\kappa n^3)+E(O(\kappa n^2))$ communication. In the common case, the protocol terminates in expected constant number of rounds with total communication complexity of $O(n^2\ell + \kappa n^3)$. In the worst case, when the protocol runs for linear number of rounds, this protocol still incurs $O(n^2\ell + \kappa n^3)$ communication; thus this protocol incurs $O(n^2\ell + \kappa n^3)$ communication even in the worst-case.

On simultaneous termination and sequential composition. Our protocols cannot provide simultaneous termination. This is similar to Feldman and Micali [18] and Katz and Koo [23]. However, we can use techniques introduced in Lindell, Lysyanskaya and Rabin [24], Katz and Koo [23] and Cohen et al. [15] for sequential composition of our protocols.

Related work. Table 1 and Table 2 presents comparisons with recent results in MVBA and parallel broadcast literature. We present a detailed discussion in Section 7.

2 Model and Preliminaries

We consider a system consisting of n parties $(P_1, \ldots, P_n)$ in a reliable, authenticated all-to-all network, where up to t parties can be Byzantine faulty. The Byzantine parties may behave arbitrarily. We consider two kinds of adversaries: (i) a static adversary, and (ii) a strongly rushing adaptive adversary. A static adversary corrupts parties before the start of the protocol execution whereas a strongly rushing adaptive adversary can adaptively decide which t parties to corrupt at any time during protocol execution. In addition, due to "strongly-rushing" nature of the adversary, the adversary is capable of corrupting a party P_h after observing message sent by party P_h in round r and remove round r messages sent by party P_h before they reach other honest parties and send round r messages after corrupting it [4]. A party that is not faulty throughout the execution is considered to be *honest* and executes the protocol as specified.

We assume a synchronous communication model. Thus, if an honest party sends a message at the beginning of some round, the recipient receives the message by the end of that round. We make use of digital signatures and a public-key infrastructure (PKI) to prevent spoofing and replays and to validate messages. Message x sent by a node P_i is digitally signed by P_i's private key and is denoted by $\langle x \rangle_i$. In addition, we use $H(x)$ to denote the invocation of the hash function H on input x.

2.1 Definitions

Definition 2.1 (Parallel Broadcast [30])**.** *In a parallel broadcast protocol, each party P_i has its input value v_i and each party P_i outputs a n-element vector $\mathcal{V}_i$ of values. A parallel broadcast protocol tolerating t Byzantine failures has the following properties:*

- ***Agreement.*** *All honest parties must agree on the same vector of values $\mathcal{V} = [v_1, \ldots, v_n]$.*
- ***Validity.*** *If the input of an honest party P_j is v_j, then $\mathcal{V}_i[j] = v_j$.*
- ***Termination.*** *All honest parties must eventually decide on a vector $\mathcal{V}$.*

Gradecast with multiple grades. Gradecast with multiples grades was originally introduced by Garay et al. [22] that supports arbitrary number of grades. We present a slightly weaker definition of gradecast with multiple grades.

Definition 2.2 (Gradecast with multiple grades)**.** *A protocol with a designated sender P_i holding an initial input v is a g^*-gradecast protocol tolerating t Byzantine faults if the following conditions hold:*

1. *Each honest party P_j outputs a value v_j with a grade $g_j \in \{0, 1, \ldots, g^*\}$.*
2. *If the sender is honest, each honest party P_j outputs v with a grade $g_j = g^*$.*
3. *If two honest parties P_j and P_k output values with grades g_j and g_k respectively, then $|g_j - g_k| \leq 1$.*
4. *If an honest party P_j outputs a value v with a grade $g_j > 1$, then all honest parties output value v.*

Our definition allows honest parties to output different values with a grade of 1 when no honest party outputs a grade > 1 while the definition of Garay et al. [22] restricts honest parties to output the same value with a grade of 1.

Multi-valued validated Byzantine agreement. In an MVBA protocol, there is an external validity function ex-validation that every party has access to. Each honest party starts with some externally valid input v_i, and on termination must output a value. An MVBA protocol tolerating t Byzantine failures has the following properties:

Definition 2.3 (Multi-valued Validated Byzantine Agreement [5, 26, 32])**.** *A protocol solves multi-valued validated Byzantine agreement if it satisfies the following properties except with negligible probability in the security parameter κ:*

- ***Agreement.*** *No two honest parties decide on different values.*
- ***Validity.*** *If an honest party decides a value v, then* ex-validation$(v) =$ true.
- ***Termination.*** *If all honest parties start with externally valid values, all honest parties eventually decide.*

2.2 Primitives

In this section, we present several primitives used in our protocols. Our protocols make use of standard coding schemes and cryptographic accumulators in the literature.

Linear erasure and error correcting codes. We use standard (n, b) Reed-Solomon (RS) codes [31]. This code encodes b data symbols into code words of n symbols using ENC function and can decode the b elements of code words to recover the original data using DEC function.

- ENC. Given inputs $m_1, \ldots, m_b$, an encoding function ENC computes $[s_1, \ldots, s_n] =$ ENC$(m_1, \ldots, m_b)$, where $[s_1, \ldots, s_n]$ are code words of length n. A combination of any b elements of n code words uniquely determines the input message and the remaining of the code word.
- DEC. The function DEC computes $[m_1, \ldots, m_b] =$ DEC$(s_1, ..., s_n)$, and is capable of tolerating up to c errors and d erasures in code words $[s_1, \ldots, s_n]$, if and only if $n - b \geq 2c + d$. In our protocol, we invoke DEC with $c = 0$ i.e., the input code words to DEC function do not contain errors; and only erasures. The DEC function will return correct output in the presence of up to $n - b$ erasures.

In our protocol, we use the (n, b) RS codes with n set to be the number of parties in the system and b set to be the number of honest parties i.e., $b = n - t$.

Cryptographic accumulators. A cryptographic accumulator scheme constructs an accumulation value for a set of values and produces a witness for each value in the set. Given the accumulation value and a witness, any party can verify if a value is indeed in the set. Formally, given a parameter k, and a set $\mathcal{D}$ of n values $d_1, \ldots, d_n$, an accumulator has the following components:

- $\mathsf{Gen}(1^k, n)$: This algorithm takes a parameter k represented in unary form 1^k and an accumulation threshold n (an upper bound on the number of values that can be accumulated securely), returns an accumulator key a_k. The accumulator key a_k is part of the q-SDH setup [11] and therefore is public to all parties.
- $\mathsf{Eval}(a_k, \mathcal{D})$: This algorithm takes an accumulator key a_k and a set $\mathcal{D}$ of values to be accumulated, returns an accumulation value z for the value set $\mathcal{D}$.
- $\mathsf{CreateWit}(a_k, z, d_i, \mathcal{D})$: This algorithm takes an accumulator key a_k, an accumulation value z for $\mathcal{D}$ and a value d_i, returns $\perp$ if $d_i \notin \mathcal{D}$, and a witness w_i if $d_i \in \mathcal{D}$.
- $\mathsf{Verify}(a_k, z, w_i, d_i)$: This algorithm takes an accumulator key a_k, an accumulation value z for $\mathcal{D}$, a witness w_i and a value d_i, returns true if w_i is the witness for $d_i \in \mathcal{D}$, and false otherwise.

In this paper, we use *collision free bilinear accumulators* from Nguyen [29] as cryptographic accumulators which generates constant sized witness, but requires q-SDH assumption [11]. Alternatively, we can use Merkle trees [27] (and avoid q-SDH assumption) at the expense of $O(\log n)$ multiplicative communication complexity.

Normalizing the length of cryptographic building blocks. Let λ denote the security parameter, $\kappa_h = \kappa_h(\lambda)$ denote the hash size, $\kappa_s = \kappa_s(\lambda)$ denote the size of the signature size, $\kappa_a = \kappa_a(\lambda)$ denote the size of the accumulation value and witness of the accumulator. Further, let $\kappa = \mathsf{max}(\kappa_h, k_s, \kappa_a)$; we assume $\kappa = \Theta(\kappa_h) = \Theta(\kappa_s) = \Theta(\kappa_a) = \Theta(\lambda)$. Throughout the paper, we will use the same parameter κ to denote the hash size, signature size and accumulator size for convenience.

3 Gradecast with Multiple Grades

In this section, we present a communication efficient gradecast protocol that supports multiple grades. Gradecast (aka graded broadcast) is a relaxed version of broadcast introduced by Feldman and Micali [18]. In gradecast, parties output a value along with a grade. Informally, the grade output by a party is an indicator of the "confidence" in the output produced by it. Thus, when the grade output by an honest party is *high*, other honest parties are expected to output the same value (even though their grade may be lower). When the grades are *lower*, there may be some amount of disagreement between the output values of different honest parties too. This is in contrast to broadcast which requires honest parties to reach a unanimous decision.

The gradecast protocol of Feldman and Micali [18] supports three grades $\{0, 1, 2\}$ and their protocol tolerates $t < n/3$ Byzantine faults in the *plain* authenticated model without PKI. Later, Garay et al. [22] generalized the gradecast protocol to the case of an arbitrary number of grades $\{0, 1, \ldots, g^*\}$ where g^* is the maximum supported grade. They gave a protocol in the authenticated model with PKI and digital signatures tolerating $t < n$ Byzantine faults and a message complexity of $O(g^* n^2)$ (this corresponds to a communication complexity of $O(g^*(\ell + \kappa)n^2)$ for input of size ℓ bits) and a round complexity of $2g^* + 1$. In this work, we present a slightly weaker definition of the gradecast with multiple grades (see Definition 2.2) and show a construction that satisfies this definition with a communication complexity of $O(n\ell + \kappa n^2)$ for input of size ℓ bits and $t < (1 - \varepsilon)n$ resilience where $\varepsilon > 0$ is some constant. Our protocol also works in the authenticated model with PKI and digital signatures.

Equivocation. Two or more messages of the same *type* but with different payload sent by a party is considered an equivocation. In order to facilitate efficient equivocation checks, the sender sends the payload along with signed hash of the payload. When an equivocation is detected, broadcasting the signed hash suffices to prove equivocation by the sender.

Deliver. As a building block, we first present a $\mathsf{Deliver}$ function (refer Figure 2) used by an honest party to efficiently propagate long messages using erasure coding techniques and cryptographic accumulators. The input parameters to the function are long message m, accumulation value z corresponding to message m and the sender's signature on $(H(m), z)$. The sender is the party who originally sent message m.

$\mathsf{Deliver}(m, z, \sigma)$:

- Partition input m into b blocks. Encode the b blocks into n code words $[s_1, \ldots, s_n]$ using ENC function. Add an index j to each code word s_j to obtain $\mathcal{D} = [(1, s_1), \ldots, (n, s_n)]$. Compute accumulation value $z_i = \mathsf{Eval}(a_k, \mathcal{D})$. If $z \neq z_i$ or σ is not a valid signature on $(H(m), z_i)$ abort. Otherwise, compute witness w_j for each element $(j, s_j) \in \mathcal{D}$ using $\mathsf{CreateWit}$ function and send $\langle \mathsf{codeword}, s_j, w_j, H(m), z \rangle_i$ and σ to party P_j $\forall P_j \in \mathcal{P}$.
- If party P_j receives the first valid code word $\langle \mathsf{codeword}, s_j, w_j, H(m), z \rangle_*$ along with the sender's signature σ such that $\mathsf{Verify}(a_k, z, w_j, (j, s_j)) = \mathsf{true}$, forward the code word and the sender's signature σ to all the parties.
- Upon receiving b valid code words for the first accumulation value z it received, decode m using DEC function.

Figure 2: **Deliver function**

We consider the invocation of the $\mathsf{Deliver}$ function by an honest party P_i. When the function is invoked using the above input parameters, the long message m is first partitioned into b blocks. The b blocks are then encoded into n code words $[s_1, \ldots, s_n]$ using ENC function (defined in Section 2). An index j is added to each code word s_j to obtain $\mathcal{D} = [(1, s_1), \ldots, (n, s_n)]$ and the accumulation value z_i is computed from $\mathcal{D}$ using Eval function. If $z \neq z_i$ or the sender signature σ is not a valid signature on $(H(m), z_i)$, party P_i aborts further operations. If party P_i did not abort, it computes the cryptographic witness w_j for each element $(j, s_j) \in \mathcal{D}$ using $\mathsf{CreateWit}$ (defined in section 2). Then, the code word and witness pair (s_j, w_j) is sent to the node $P_j \in \mathcal{P}$ along with the sender's signature σ.

When a node P_j receives the first valid code word s_j for an accumulation value z such that the witness w_j verifies (j, s_j) (using Verify function defined in section 2), it forwards the code word and witness pair (s_j, w_j) along with the sender's signature σ to all parties. Note that party P_j forwards only the first valid code word and witness pair (s_j, w_j). Thus, it is required that all honest parties forward the code word and witness pair (s_j, w_j) for long message m; otherwise all honest nodes may not receive b code words required to decode the long message m.

When a party P_i receives b valid code words corresponding to the first accumulation value z (or the first valid code word) it receives, it reconstructs the object m.

The $\mathsf{Deliver}$ function incurs 2 rounds. Our $\mathsf{Deliver}$ function improves upon $\mathsf{Deliver}$ function of Rand-Piper [10] to tolerate dishonest majority Byzantine faults.

Set $o_i = \perp$ and $g_i = 0$. Each party P_i performs the following operations:

- **Round 1:** If party P_j is the designated sender, then it multicasts (v, z, σ) where v is the input value, z is the accumulation value of v and σ is the its signature on $(H(m), z))$.
- **Round** $2h$ **(**$h \in [1, g^* - 1]$**):** If party P_i receives (v, z, σ) for the first time, it invokes $\mathsf{Deliver}(v, z, \sigma)$.
- **Round** $2g^*$**:** If party P_i invoked $\mathsf{Deliver}$ for value v without aborting by round $2g^* - 2$ and no party P_j equivocation has been detected so far, set $o_i = v$ and $g_i = 2$. Let v_i be the first value received. If $v_i = \perp$, set $o_i = \perp$ and $g_i = 0$, else if $o_i = \perp$, set $o_i = v_i$ and $g_i = 1$.
- **Round** $2g^* + h$ **(**$h \in [1, g^* - 2]$**):** If party P_i invoked $\mathsf{Deliver}$ for value v by Round $2g^* - 2(h + 1)$ and no party P_j equivocation has been detected so far, set $g_i = g_i + 1$. At round $3g^* - 2$, output (o_i, g_i).

Figure 3: **M-Gradecast(v, g^*) with $O(n\ell + (\kappa + w)n^2)$ communication.**

3.1 Protocol details

We construct a protocol $\mathsf{M\text{-}Gradecast}(v, g^*)$ where v is the sender's value and g^* is the maximum supported grade. The $\mathsf{M\text{-}Gradecast}(v, g^*)$ protocol is presented in Figure 3. In round 1, the designated sender P_j multicasts (v, z, σ) where v is its input value, z is the accumulation value and σ is its signature on $(H(m), z)$. We note that the size of input value v can be large. In order to facilitate efficient equivocation checks, the sender P_j signs $(H(v), z)$ and sends the signature σ. Whenever an equivocation by the sender is detected, multicasting these signatures suffices to prove equivocation by the sender.

During rounds $2h$ for $h \in [1, g^* - 1]$, if party P_i receives (v, z, σ) for the first time, it invokes Deliver to propagate long message v. Note that if party P_i invoked Deliver in round 2, it does not invoke Deliver again in later rounds. Also, note that Deliver function requires 2 rounds. Rounds $2h + 1$ for $h \in [1, g^* - 1]$ accommodates steps of Deliver function invoked in rounds $2h$ for $h \in [1, g^* - 1]$. We note that although parties may invoke Deliver to propagate long message v in different rounds, they forward their code words only the first time. For example, if a party P_i invoked Deliver in round 2 and an honest party P_k received its first valid code word (s_k, w_k) in round 3 for accumulation value z, it forwards the code word to all parties in round 3. Later, if some other party (say party P_h) invokes Deliver in round 4 and party P_k receives code word (s_k, w_k) again in round 5, party P_k does not forward (s_k, w_k) again. This helps in keeping communication complexity to $O(n\ell + \kappa n^2)$.

In round $2g^*$, each party P_i sets its output value and initial grades. If party P_i invoked Deliver for value v at any prior rounds, and it did not detect any equivocation so far, it sets $o_i = v$ and $g_i = 2$. We note that an honest party decodes long messages corresponding to the first valid code word (or the first accumulatioin value z) they receive even though it detects equivocation as long as it receives b valid code words. We refer to this value as the first received value. Let v_i be the first value received. If $v_i = \perp$, it sets $o_i = \perp$ and $g_i = 0$. Otherwise if $o_i = \perp$, set $o_i = v_i$ and $g_i = 1$ irrespective of the equivocation i.e., if P_i did not invoke Deliver for any values but receives a value $v_i \neq \perp$, it sets $o_i = v_i$ and $g_i = 1$.

In round $2g^* + h$ for $h \in [1, g^* - 2]$, each party P_i updates their grade g_i based on when they invoked Deliver and if they have detected any equivocation so far.

Optimal communication complexity. Our M-Gradecast(v, g^*) incurs $O(n\ell + \kappa n^2)$ communication for input of ℓ bits while tolerating $t < (1 - \varepsilon)n$ Byzantine faults where $\varepsilon > 0$ is a constant. In a recent work, Shrestha et al. [32] designed a weak-gradecast protocol where grades are in the range $\{0, 1, 2\}$ with a communication complexity of $O(n\ell + \kappa n^2)$ for input of size ℓ bits in the same setting and gave a communication lower bound of $\Omega(n\ell + n^2)$ for weak-gradecast. The communication lowerbound of Shrestha et al. [32] can trivially be extended to show the optimal communication complexity of our M-Gradecast(v, g^*) protocol.

3.2 Security Analysis of Gradecast with Multiple grades

Claim 7. *Suppose party P_j is the designated sender. If an honest party invokes* Deliver *in round r without aborting for value v sent by party P_j and no honest party has detected a party P_j equivocation by round $r+1$, then all honest parties will receive value v by round $r + 2$.*

Proof. Suppose an honest party P_i invokes Deliver at round r without aborting for a value v sent by party P_j. This implies party P_j sent a valid signature σ on $(H(v), z)$ where z is the accumulation value of v. Moreover, party P_i must have sent valid code words and witness $\langle \mathsf{codeword}, s_k, w_k, H(v), z \rangle_i$ computed from value v to every party P_k $\forall P_k \in \mathcal{P}$ along with the party P_j's signature σ at round r. The code words, witness and signature σ arrive at all honest parties by round $r + 1$.

Since no honest party has detected a party P_j equivocation by round $r+1$, it must be that either honest parties will forward their code word $\langle \mathsf{codeword}, s_k, w_k, H(v), z \rangle$ when they receive the code words sent by party P_i or they already sent the corresponding code word when they either invoked Deliver for value v or received the code word from some other party. In any case, all honest parties will forward their code word corresponding to value v by round $r+1$. In addition, accumulation value z is the first received accumulation value for all honest parties. Thus, all honest parties will have received b valid code words for the first received accumulation value z by round $r + 2$ sufficient to decode value v. □

Theorem 8. *The protocol in Figure 3 is a gradecast protocol with multiple grades satisfying Definition 2.2.*

Proof. Suppose party P_j is the designated sender with its input value v. Let g^* be the maximum grade.

We first consider the case when an honest party P_i outputs value v with a grade $g_i = 2$ and no honest party outputs a value with a grade > 2. Honest party P_i must have invoked Deliver for value v by round $2g^* - 2$ and did not detect a party P_j by round $2g^*$. This implies no honest party detected a party P_j equivocation by round $2g^* - 1$. By Claim 7, all honest parties receive value v by round $2g^*$. In addition,

since party P_i invoked Deliver for value v by round $2g^* - 2$, all honest parties receive a code word for value v by round $2g^* - 1$. Thus, value v is the first value received by all honest parties. Since $v \neq \perp$, all honest parties will output value v with a grade ≥ 1.

Next, we consider the case when an honest party P_i outputs a value v with a grade $g_i > 2$. Without loss of generality, assume g_i is the highest grade output by any honest party. Let $h = g_i - 2$. Since, party P_i outputs value v with a grade $g_i > 2$, it must have invoked Deliver to propagate value v by round $2g^* - 2(h+1)$ and did not detect any party P_j equivocation by round $2g^* + h$. This implies no other honest party detected a party P_j equivocation by round $2g^* + h - 1$. With $h \geq 1$, $2g^* + h - 1 > 2g^* - 2(h+1) + 1$. Thus, by Claim 7, all other honest parties receive value v by round $2g^* - 2h$. The honest parties that did not invoke Deliver by round $2g^* - 2(h+1)$ will invoke Deliver for value v by round $2g^* - 2h$. Since no other honest party detected a party P_j equivocation by round $2g^* + h - 1$, all honest parties will set a grade of 2 in round $2g^*$. In addition, all honest parties will set a grade of $2 + h - 1 = g_i - 1$ by round $2g^* + h - 1$. Thus, all honest parties will output value v with a grade at least $g_i - 1$.

This also proves that if an honest party P_i outputs a value v with a grade $g_i > 1$, then all honest parties output value v.

Next, we consider the case when the designated sender is honest. Since, the sender is honest, it sends its input value v to all honest parties such that all honest parties receive value v in round 2. Thus, all honest parties invoke Deliver to propagate value v in round 2. Moreover, the honest sender does not equivocate. Thus, all honest parties set a grade of 2 in round $2g^*$ and set a grade of $2 + g^* - 2 = g^*$ in round $3g^* - 2$.

The case where each honest party outputs a value with a grade $\in \{0, 1, \ldots, g^*\}$ is trivial by design. □

Lemma 9 (Communication Complexity)**.** *Let ℓ be the size of the input, κ be the size of accumulator, and w be the size of witness. The communication complexity of the protocol in Figure 3 is $O(n\frac{\ell}{\varepsilon} + (\kappa + w)n^2)$.*

Proof. At the start of the protocol, the sender multicasts its value of size ℓ to all party P_j $\forall j \in [n]$ along with κ sized signed message containing accumulator and hash of large message. This step incurs $O(n\ell + \kappa n)$. An honest party invokes Deliver only on the first value it receives where it sends a code word of size $O(\ell/b)$, a witness of size w, an accumulator of size κ to each party and a singature of size κ bits. Moreover, each party multicasts a code word of size $O(\ell/b)$, a witness of size w, an accumulator of size κ and a singature of size κ bits. Thus, each party sends $O(n\ell/b + (\kappa + w)n) = O(n\frac{\ell}{n-t} + (\kappa + w)n) = O(\frac{\ell}{\varepsilon} + (\kappa + w)n)$ bits. Thus, the overall complexity is is $O(n\frac{\ell}{\varepsilon} + (\kappa + w)n^2)$ bits. □

4 Graded Parallel Broadcast

In this section, we present a new primitive that we call *Graded Parallel Broadcast* that is secure against $t < n/2$ Byzantine faults. Graded parallel broadcast is a relaxation of parallel broadcast [33] and uses gradecast with multiple grades to propagate its input. In this work, we consider an instance of gradecast with multiple grades where the grades can be in the range $\{0, 1, \ldots, 4\}$. In our construction, each party P_i uses M-Gradecast$(., 4)$ to propagate its input v_i and output an n-element list of values along with an n-element list of grades (GradeList$_i$). Looking ahead, our aim is to have each party P_i feed its output of graded parallel broadcast (i.e., GradeList$_i$) into a Byzantine consensus primitive to agree on a common GradeList$_h$. The agreed GradeList$_h$ can be a Byzantine parties' input too. However, a Byzantine party may set arbitrary grades in its GradeList corresponding to an honest sender and prevent honest input from appearing in the final output. In order to prevent this scenario, we restrict a Byzantine party from setting arbitrary grades and consider only a *valid* GradeList. A *valid* GradeList has (i) at least $n - t$ entries of grade 4, i.e., $|\{h \mid \text{GradeList}[h] = 4\}| \geq n - t$, (ii) GradeList$[i] \in \{3, 4\}$ corresponding to honest sender P_i. Note that for an honest sender P_k, each honest party P_i sets a grade GradeList$_i[k] = 4$. Thus, a valid GradeList must have at least $n - t$ entries of 4. Moreover, due to the properties of M-Gradecast$(., 4)$, the grades of two parties for the same sender can differ by at most 1. Since each honest party sets a grade of 4 for an honest sender P_k, a Byzantine party must set a grade of at least 3 corresponding to an honest sender P_k for its GradeList to be valid. In the final parallel broadcast protocol, we consider all values with grades in the range $\{3, 4\}$ corresponding to agreed GradeList.

In graded parallel broadcast, we ensure that a valid GradeList is *certified*, i.e., it is accompanied by a set of signatures from at least $t+1$ parties. A set of $t+1$ signatures on GradeList forms the certificate for GradeList and denoted as $\mathcal{AC}(\mathsf{GradeList})$.

Definition 4.1 (Graded Parallel Broadcast)**.** *Each party P_i, as a sender, sends its input v_i. Each honest party P_j outputs an n-element list of values along with a n-element list $\mathsf{GradeList}_j$ with an entry corresponding to each party as a sender such that $\mathsf{GradeList}_j[h] \in \{0,1,2,3,4\}\ \forall h \in [n]$. A graded parallel Broadcast protocol tolerating t Byzantine failures satisfies the following properties:*

1. *If sender P_i is honest, then each honest party P_j sets $\mathsf{GradeList}_j[i] = 4$.*
2. *A certified $\mathsf{GradeList}_k$ must have $|\{h \mid \mathsf{GradeList}_k[h] = 4\}| \geq n - t$.*
3. *If the sender P_i is honest and $\mathsf{GradeList}_k$ is certified, then $\mathsf{GradeList}_k[i] \in \{3,4\}$.*
4. *If $\mathsf{GradeList}_k$ is certified and $\mathsf{GradeList}_k[i] \in \{3,4\}$, then all honest parties have received a common value v_i.*

Each party P_i with its initial input v_i performs following operations:

1. **(Round 1) Propose.** Each party P_i invokes $\mathsf{M\text{-}Gradecast}(v_i, 4)$.
2. **(Round 10) Propose Grade.** Let $(o_{j,i}, g_{j,i})$ be the output of M-Gradecast of party P_i with party P_j as sender. Set $\mathsf{GradeList}_i[j] = g_{j,i}$. Multicast $\langle \mathsf{grade\text{-}list}, \mathsf{GradeList}_i \rangle_i$.
3. **(Round 11) Verify and Ack.** Upon receiving $\langle \mathsf{grade\text{-}list}, \mathsf{GradeList}_j \rangle_j$ from party P_j, if the following conditions hold send $\langle \mathsf{ack}, H(\mathsf{GradeList}_j) \rangle_i$ to party P_j.
   (a) $|\{h \mid \mathsf{GradeList}_j[h] = 4\}| \geq n - t$
   (b) $|\mathsf{GradeList}_j[h] - \mathsf{GradeList}_i[h]| < 2\ \forall h \in [n]$.

Figure 4: Graded Parallel Broadcast with $O(n^2\ell + (\kappa + w)n^3)$ communication

Protocol Details. Each party P_i uses $\mathsf{M\text{-}Gradecast}(., 4)$ to propagate its input v_i. At the end of $\mathsf{M\text{-}Gradecast}(., 4)$ invocation, each honest party P_i outputs an n element list of values along with n element list of grades, denoted by $\mathsf{GradeList}_i$, with an entry corresponding to each party as a sender.

Party P_i then multicasts its $\mathsf{GradeList}_i$ to all other parties. Party P_j then checks the validity of $\mathsf{GradeList}_i$ by checking if (i) $|\{h \mid \mathsf{GradeList}_i[h] = 4\}| \geq n - t$, and (ii) $|\mathsf{GradeList}_j[h] - \mathsf{GradeList}_i[h]| < 2\ \forall h \in [n]$. The first check ensures that $\mathsf{GradeList}_i$ contains at least $n - t$ entries with $\mathsf{GradeList}_i[h] = 4$. Note that for an honest sender P_k, each honest party P_i outputs a value with $\mathsf{GradeList}_i[k] = 4$. Thus, a valid GradeList must have at least $n - t$ entries of 4. In addition, due to the properties of $\mathsf{M\text{-}Gradecast}(., 4)$, the grades of any two parties corresponding to a sender differs by at most 1. Thus, a valid GradeList must satisfy $|\mathsf{GradeList}_j[h] - \mathsf{GradeList}_i[h] < 2|\ \forall h \in [n]$. This check also prevents a Byzantine party from setting too low grades corresponding to an honest sender; otherwise its GradeList would not be certified. Thus, a Byzantine party must set a grade of at least 3 corresponding to an honest sender for its GradeList to be certified.

If the checks pass, party P_j sends $\langle \mathsf{ack}, H(\mathsf{GradeList}_i) \rangle_j$ to party P_i. A set of $t+1$ ack (ack-cert) messages for $\mathsf{GradeList}_i$ (denoted by $\mathcal{AC}(\mathsf{GradeList}_i)$) implies at least one honest party has verified $\mathsf{GradeList}_i$.

4.1 Security Analysis of Graded Parallel Broadcast

Theorem 10. *The protocol in Figure 4 is a graded Parallel Broadcast protocol satisfying Definition 4.1.*

Proof. If the sender P_i is honest, it propagates its input v_i using M-Gradecast. By Theorem 8, each honest party P_j output $\mathsf{GradeList}_j$ with $\mathsf{GradeList}_j[i] = 4$.

Next, we consider a certified grade list $\mathsf{GradeList}_k$. The only way $\mathsf{GradeList}_k$ gets certified is if at least one honest party P_j sends an ack for it. If an honest party P_j sends an ack for a grade list $\mathsf{GradeList}_k$, then

it must be that (i) $|\{h \mid \mathsf{GradeList}_k[h] = 4\}| \geq n - t$ and (ii) $|\mathsf{GradeList}_k[h] - \mathsf{GradeList}_j[h] < 2| \ \forall h \in [n]$. Trivially, this implies a certified $\mathsf{GradeList}_k$ must have $|\{h \mid \mathsf{GradeList}_k[h] = 4\}| \geq n-t$. This also implies that if $\mathsf{GradeList}_k[h] \in \{3, 4\}$, $\mathsf{GradeList}_j[h]$ must be at least 2. By the properties of M-Gradecast (Definition 2.2), if an honest party outputs a value v_h with a grade of > 1, all honest parties output a common value v_h. Thus, all honest parties have common value v_h for all h such that $\mathsf{GradeList}_k[h] = \{3, 4\}$.

Next, we consider the grades in $\mathsf{GradeList}_k[j]$ for an honest sender P_j. We know from the fact that for an honest sender P_j, by the properties of M-Gradecast$(v, 4)$, all honest parties will set a grade of 4. An honest party P_i will send an ack for $\mathsf{GradeList}_k$ only if $|\mathsf{GradeList}_k[j] - \mathsf{GradeList}_i[j] < 2|$. This implies $\mathsf{GradeList}_k[j]$ must be at least 3 i.e. $\mathsf{GradeList}_k[j] \in \{3, 4\}$.

□

Lemma 11 (Communication Complexity)**.** *Let ℓ be the size of commitment* comm*, κ be the size of secret share and accumulator, and w be the size of witness. The communication complexity of the protocol is $O(n^2\ell + (\kappa + w)n^3)$ bits per epoch.*

Proof. In the Propose step, each party P_i invokes M-Gradecast$(., 4)$ protocol. By Lemma 9, the communication complexity of one invocation of M-Gradecast protocol is $O(n\ell + (\kappa + w)n^2)$. Thus, this step incurs $O(n^2\ell + (\kappa + w)n^3)$.

In the Propose grade step, each party multicast their GradeList of size $O(n)$. Multicast of $O(n)$-sized GradeList by n parties incurs $O(n^3)$ communication. In the Verify and Ack step, each party sends at most n ack messages. This step incurs $O(\kappa n^2)$ communication. Thus, the total communication complexity is $O(n^2\ell + (\kappa + w)n^3)$ bits. □

5 Multi-valued Validated Byzantine Agreement

In this section, we present an efficient protocol for multi-valued validated Byzantine agreement (MVBA) tolerating $t < n/2$ Byzantine faults with security against a strongly rushing adaptive adversary. MVBA protocol allows honest parties to agree on any externally valid input; the agreed value can be the input of a Byzantine party as long as it is externally valid. The problem of multi-valued validated Byzantine agreement has been extensively studied in the asynchronous model. In the synchronous model, Shrestha et al. [32] recently gave an MVBA protocol in the authenticated model with PKI and digital signatures tolerating $t < n/2$ Byzantine faults and secure against a static adversary. Their protocol incurs a communication complexity of $O(n^2\ell + \kappa n^3)$ communication in expectation for input of size ℓ bits and terminates in expected $O(1)$ rounds.

In this work, we improve upon their result by a linear factor in communication and also design an MVBA protocol secure against a strongly rushing adaptive adversary. We make threshold setup assumptions and rely on adaptively-secure threshold signature scheme due to Loss and Moran [25] to perform a perfect leader election where all honest parties obtain a common leader all the time. This assumption provides us with three major advantages (i) the leader election can be performed in $O(\kappa n^2)$ communication, (ii) we can obtain security against an adaptive adversary, and (iii) since the leader election is perfect (i.e, all honest parties observe a common leader), we only need to ensure the leader's proposal is propagated among all parties; this allows us to obtain $O(n\ell + \kappa n^2)$ communication in expectation and security against an adaptive adversary.

The starting point of our construction is the adaptively-secure Byzantine synod protocol of Abraham et al. [4] which has a communication complexity of $O((\ell + \kappa)n^2)$ for ℓ bit input values and termination in expected 16 rounds. Our MVBA protocol inherits the underlying consensus mechanism of their protocol and improves the dissemination of the proposals to obtain $O(n\ell + \kappa n^2)$ communication. Our solution uses Reed-Solomon erasure codes [31] to decode large messages into n code words and cryptographic accumulators [29] to verify the correctness of the code words. Section 1.1 presents the key ideas behind our improvement.

Epoch. Our protocol progresses through a series of numbered epochs. Each epoch lasts for 8 rounds.

Certified values and ranking. A certificate on a value v_i consists of $t + 1$ distinct signatures in an epoch e and is represented by $\mathcal{C}_e(v_i)$. Certificates are ranked by epochs, i.e., values certified in a higher epoch has a

Each party P_i with its input v_i performs following operations:

- **Round 1:** Each party P_i partitions its input v_i into $t+1$ data symbols and encode the $t+1$ data symbols into n code words $(s_{i,1}, \ldots, s_{i,n})$ using ENC function. Compute accumulation value z_{v_i} using Eval function and witness $w_{i,j}$ $\forall s_{i,j} \in (s_{i,1}, \ldots, s_{i,n})$ using CreateWit function. Send $\langle \mathsf{codeword}, s_{i,j}, w_{i,j}, z_{v_i} \rangle_i$ to party P_j $\forall j \in [n]$.
- **Round 2:** If party P_i receives the first valid code word $\langle \mathsf{codeword}, s_{j,i}, w_{j,i}, z_{v_j} \rangle_j$ for the accumulator z_{v_j}, send an $\langle \mathsf{ack}, z_{v_j} \rangle_i$.
- **Round 3:** Upon receiving $t+1$ distinct $\langle \mathsf{ack}, z_{v_i} \rangle_*$ message, create a threshold signature, denoted as $\mathcal{AC}(z_{v_i})$.

Figure 5: Proposal Dispersal with $O(n\ell + \kappa n^2)$ communication

higher rank. During the protocol execution, each party keeps track of all certified blocks and keeps updating the highest ranked certified block to its knowledge. Parties lock on the highest ranked certified values and do not vote for values other than the locked values to ensure safety of a commit.

5.1 Protocol Details

We first present a protocol used by all parties to efficiently distribute their long ℓ bit input v_i at the cost of $O(n\ell + \kappa n^2)$ communication. This protocol is executed before the MVBA protocol (refer Figure 6).

Proposal dispersal. In proposal dispersal protocol (refer Figure 5), each party makes use of erasure coding techniques and cryptographic accumulators to efficiently distribute its long message. Each party P_i partitions its input v_i into $t+1$ data symbols. The $t+1$ data symbols are then encoded into n code words $(s_{i,1}, \ldots, s_{i,n})$ using ENC function and a corresponding accumulation value z_{v_i} is computed. Then, the cryptographic witness $w_{i,j}$ is computed for each code word $s_{i,j} \in (s_{i,1}, \ldots, s_{i,n})$ using CreateWit. Then, the code word and witness pair $(s_{i,j}, w_{i,j})$ is sent to the partyP_j $\forall j \in [n]$ along with the accumulation value z_{v_i}.

When a party P_j receives the first valid code word $s_{i,j}$ for an accumulation value z_{v_i} such that the witness $w_{i,j}$ verifies the code word $s_{i,j}$, it sends an $\langle \mathsf{ack}, z_{v_i} \rangle_j$ to party P_i. When party P_i receives $t+1$ ack messages for z_{v_i}, it forms an ack-cert for value v_i, denoted as $\mathcal{AC}(z_{v_i})$. Note that an ack-cert for value v_i does not imply all honest parties have received valid code words corresponding to value v_i; this only implies at least one honest party has received a valid code word corresponding to value v_i. When the sender P_i is honest, then all honest parties will receive a valid code word corresponding to value v_i which is sufficient to decode value v_i. In the MVBA protocol that follows, each party P_i proposes accumulation value z_{v_i} along with $\mathcal{AC}(z_{v_i})$ and honest parties only consider proposals containing an ack-cert. Collecting an ack-cert for a proposal is similar to having a proposal prepared in the Byzantine synod protocol of Abraham et al. [4]. However, it does not guarantee that all honest parties will be able to decode the proposed value.

In the proposal dispersal protocol, each party P_j receives only a single code word $s_{i,j}$ corresponding to value v_i. For n proposals each of size ℓ bit, this protocol incurs $O(n\ell + (\kappa + w)n^2)$ bits where κ is the size of accumulator and w is the size of the accumulator *witness*.

MVBA Protocol. At the start of the MVBA protocol (refer Figure 6), no party has a certificate for any proposed value; thus each party P_i sends a status message with an empty certificate. Consequently, $\mathcal{CC}_i = \perp$ for each party P_i and each party P_i multicasts its own value $(z_{v_i}, \mathcal{AC}(z_{v_i}))$ in the propose step of the first epoch. In subsequent epochs, parties send proposals corresponding to the highest ranked certificate known to them. Note that a valid proposal is accompanied by an ack-cert which can only be formed during a proposal dispersal phase; this is because a ack-cert consists of at least $t+1$ ack for z_{v_i} and honest parties send ack for z_{v_i} only in the proposal dispersal phase. In the MVBA protocol, all parties send their proposals first and a leader is elected in a later round. This prevents an adaptive adversary from corrupting the elected party and sending valid equivocating proposals afterwards; this is because ack-cert for an equivocating proposal cannot form afterwards.

In round 3, parties participate in the adaptively-secure threshold coin tossing scheme due to Loss and Moran [25] to randomly select a common leader L_e for epoch e. The leaders are elected uniformly at random,

Each party P_i with its input v_i executes the proposal disperal protocol (refer Figure 5) and outputs $\mathcal{AC}(z_{v_i})$. Then, each party P_i performs the following operations for each epoch e:

1. **(Round 1) Status.** Multicast the highest ranked certificate known to party P_i in the form of $\langle \mathsf{status}, \mathcal{C}_{e'}(z_{v_h}), \mathcal{AC}(z_{v_h})\rangle_i$.
2. **(Round 2) Propose.** Let $\mathcal{CC}_i := \mathcal{C}_{e'}(z_{v_h})$ be the highest ranked certificate known to party P_i at the end of Status round. If $\mathcal{CC}_i \neq \perp$, set $\mathsf{val}_i = (z_{v_h}, \mathcal{AC}(z_{v_h}))$; otherwise set $\mathsf{val}_i = (z_{v_i}, \mathcal{AC}(z_{v_i}))$. Each party P_i multicasts $\langle \mathsf{propose}, \mathsf{val}_i, \mathcal{CC}_i, e\rangle_i$.
3. **(Round 3) Elect.** Each party P_i participates in threshold coin tossing scheme from [25]. Let L_e be leader of epoch e.
4. **(Round 4) Forward.** Upon receiving the first valid proposal $\langle \mathsf{propose}, (z_{v_h}, \mathcal{AC}(z_{v_h})), \mathcal{CC}_{L_e}, e\rangle_{L_e}$ forward the proposal. If $\mathcal{CC}_i \leq \mathcal{CC}_{L_e}$, party P_i forwards a valid code word $\langle \mathsf{codeword}, s_{h,i}, w_{h,i}, z_{v_h}, e\rangle_i$ consistent with accumulator z_{v_h} sent by party P_h during proposal dispersal phase (if party P_i received a code word for z_{v_h}).
5. **(Round 5) Decode.** Upon receiving $t+1$ valid code words for the accumulator z_{v_h}, decode v_h using DEC function if party P_i has not already received v_h in earlier epochs. Send $\langle \mathsf{codeword}, s_{h,j}, w_{h,j}, z_{v_h}, e\rangle_i$ to party P_j $\forall j \in [n]$.
6. **(Round 6) Forward2.** If party P_i receives the first valid code word $\langle \mathsf{codeword}, s_{h,i}, w_{h,i}, z_{v_h}, e\rangle_*$ for the accumulator z_{v_h}, forward the code word to all the parties.
7. **(Round 7) Vote.** If party P_i receives v_h by round 5, $\mathcal{CC}_i \leq \mathcal{CC}_{L_e}$, $\mathsf{ex\text{-}validation}(v_h) = \mathsf{true}$ and no equivocating proposal by L_e has been detected so far in epoch e, multicast a vote in the form of $\langle \mathsf{vote}, e, H(z_{v_h})\rangle_i$.
8. **(Round 8) Commit.** Upon receiving $t+1$ distinct vote for z_{v_h} (denoted by $\mathcal{C}_e(z_{v_h})$), multicast $\mathcal{C}_e(z_{v_h})$, commit v_h and multicast $\langle \mathsf{terminate}, e, H(v_h)\rangle_i$.
9. **(At any time) Terminate.** Upon receiving $t+1$ $\langle \mathsf{terminate}, e, H(v_h)\rangle_*$ messages, multicast it, output v_h and terminate.
10. **(At any time) Equivocation.** Multicast the equivocating proposals signed by L_e. Stop performing epoch e operations.

Figure 6: **MVBA** with $O(n\ell + \kappa n^2)$ bits communication per epoch and expected $O(1)$ epochs

a common honest leader is elected with probability at least $\frac{1}{2}$. Let $\langle \mathsf{propose}, (z_{v_h}, \mathcal{AC}(z_{v_h})), \mathcal{CC}_{L_e}, e\rangle$ be $L_e's$ proposal for epoch e. If $\mathcal{CC}_i \leq \mathcal{CC}_{L_e}$, party P_i forwards a code word $(s_{h,i}, w_{h,i})$ corresponding to the L_e's proposal for z_{v_h} if party P_i has received $(s_{h,i}, w_{h,i})$ either during the proposal dispersal phase or in earlier epochs. We note again that an $\mathsf{ack\text{-}cert}$ on accumulation value z_{v_h} (i.e., $\mathcal{AC}(z_{v_h})$) does not imply that all honest parties have received valid code words corresponding to value v_h during proposal dispersal phase. Thus, all honest parties may not forward their code word corresponding to value v_h in round 4.

In round 5, if party P_i receives $t+1$ valid code words for the accumulator z_h, it decodes value v_h using DEC function. Party P_i again encodes value v_h and sends code word $(s_{h,j}, w_{h,j})$ to party P_j $\forall j \in [n]$. In round 6, party P_j forwards the valid code word $(s_{h,j}, w_{h,j})$ to all parties if it has not already forwarded the code word $(s_{h,j}, w_{h,j})$ in round 4. Multicasting code words in round 5 and forwarding codewords in rounds 5 and 6 ensures that if an honest party successfully decodes v_h, all honest parties will receive value v_h by the end of round 6.

Note that the elected leader could be Byzantine and that leader might not have sent valid code words to all honest parties during proposal dispersal phase and all honest parties may not have received valid code words corresponding to value v_h although an $\mathcal{AC}(z_{v_h})$ exists. Thus, it is possible that no honest party receives $t+1$ valid code words for accumulator z_{v_h} required to decode value v_h in round 5. In such a case, we ensure no honest party commits value v_h. In our protocol, we require that an honest party be able to decode value v_h in timely manner before voting for value v_h and later commit it. In particular, we rely on synchrony assumption to detect "bad" proposals and prevent it from getting committed.

Thus, party P_i votes for value v_h only if it decodes value v_h by round 5. Party P_i also checks if it did not detect equivocating proposals made by leader L_e in epoch e. This check ensure that if an honest party votes for a value v_h in round 7, all honest parties receive value v_h by round 7. In addition, party P_i also checks the proposed value is externally valid (i.e., $\mathsf{ex\text{-}validation}(v_h) = \mathsf{true}$) and the leader L_e is proposing with the highest ranked certificate. This ensures the safety of a committed value in earlier epochs.

An honest party P_i commits value v_h if it receives $t+1$ distinct votes for v_h. It multicasts the vote certificate and $\langle \mathsf{terminate}, e, H(v_h)\rangle$. In the next round, all honest parties will receive the vote certificate and not vote for lower ranked certificates in future epochs. In addition, if an honest party receives $t+1$ distinct $\langle \mathsf{terminate}, e, H(z_{v_h})\rangle$ in a round, all honest parties receive the termination certificate, output value v_h and terminate in the next round.

Optimal communication complexity. Each party needs to learn ℓ bit input; thus, a protocol must incur $\Omega(n\ell)$ communication [20]. In Abraham et al. [3], they show $\Omega(n^2)$ communication is required even for a randomized Byzantine agreement protocol secure against a strongly rushing adaptive adversary. Thus, our MVBA protocol has optimal communication complexity of $O(n\ell + \kappa n^2)$ in expectation.

Round complexity. In an epoch, an honest leader is elected with probability at least $\frac{1}{2}$. All honest parties commit and terminate in the same epoch when an honest leader is elected. Thus, the protocol terminates in expected 2 epochs. The proposal dispersal phase requires 2 rounds. Multicast of the termination certificate requires one more additional round. Thus, the protocol terminates in 19 rounds in expectation.

5.2 Security Analysis of Multi-valued Validated Byzantine Agreement

Claim 12. *If an honest party votes for value v_h at round 7, then all honest parties receive value v_h by round 7.*

Proof. Suppose an honest party P_i votes for value v_h at round 7 in epoch e. Then party P_i must have decoded value v_h by round 5 and did not detect equivocating proposals by leader L_e by round 7. Party P_i must have sent valid code words and witness $\langle \mathsf{codeword}, \mathsf{mtype}, s_{h,k}, w_{h,k}, z_{v_h} \rangle_i$ computed from value v_h to every party P_k $\forall k \in [n]$ at round 5. The code words and witness arrive at all honest parties by round 6. In addition, no honest party detected an equivocating proposal by round 6 in epoch e.

Since no honest party detected an equivocating proposal by round 6 in epoch e., it must be that either honest parties will forward their code word $\langle \mathsf{codeword}, \mathsf{mtype}, s_{h,k}, w_{h,k}, z_{v_h} \rangle$ when they receive the code words sent by party P_i or they already sent the corresponding code word in round 5 or received the code word from some other party. In any case, all honest parties will forward their code word corresponding to value v_h by round 6. Thus, all honest parties will have received $t+1$ valid code words for a common accumulation value z_{v_h} by round 7 sufficient to decode value v_h. □

Lemma 13. *If an honest party commits value v_h in epoch e, then (i) an equivocating certificate does not exist in epoch e, and (ii) all honest parties receive $\mathcal{C}_e(z_{v_h})$ by the end of epoch e.*

Proof. Suppose an honest party P_i commits value v_h in epoch e. Party P_i must have received at least $t+1$ vote messages for value v_h at round 8 in epoch e. At least one honest party (say party P_j) must have voted for value v_h at round 7 in epoch e. Party P_j votes for value v_h when it receives value v_h by round 5, invokes $\mathsf{Deliver}$ for value v_h and does not detect any equivocating proposal by leader L_e by round 7. By Claim 12, all honest parties receive value v_h by round 7. Thus, no honest party votes for a conflicting value and an equivocating certificate does not exist in epoch e. This proves part (i) of the Lemma.

For part (ii), note that party P_i multicasts $\mathcal{C}_e(z_{v_h})$ when it commits value v_h in round 8. Thus, all honest parties receive $\mathcal{C}_e(z_{v_h})$ by end of round 8. By part (i) of the Lemma, an equivocating certificate does not exist. Thus, all honest parties will receive $\mathcal{C}_e(z_{v_h})$ by the end of epoch e. □

Theorem 14 (Safety)**.** *If two honest parties commit v and v', then $v = v'$.*

Proof. Suppose an honest party P_i commits value v in epoch e. By Lemma 13, all honest parties receive $\mathcal{C}_e(v)$ by the end of epoch e and no equivocating certificate exists in epoch e. Thus, no honest party votes for values other than v in any epoch $e' > e$ and an equivocating certificate cannot form in epochs higher than $e' > e$. Thus, it must be that if two honest party commits to v and v', then $v = v'$ □

Theorem 15 (Termination)**.** *If the leader L_e of epoch e is honest, all honest parties terminate by epoch e.*

Proof. Suppose the leader L_e of epoch e is honest. Leader L_e will send the same proposal $(z_{v_h}, \mathcal{AC}(z_{v_h}))$ to all parties by extending the highest ranked certificate known to all honest parties. Thus, each honest party P_i will forward valid code word (s_i, w_i) corresponding to value v_h to all parties in round 4 and all honest parties will receive $t+1$ valid code words sufficient to decode value v_h in round 5. Thus, each honest party P_i will vote for value v_h in epoch 7, receive $\mathcal{C}_e(z_{v_h})$ by round 8 and commit v_h. In addition each honest party P_i will multicast $\langle \mathsf{terminate}, e, H(v_h) \rangle_i$, receive $t+1$ distinct $\mathsf{terminate}$, terminate by the end of round 8 of epoch e. □

Theorem 16. *The protocol in Figure 6 is a multi-valued validated Byzantine agreement protocol satisfying Definition 2.3*

Proof. For a value v_h to be decided at least one honest party must vote for it. For an honest party to vote for value v_h it must be that ex-validation(v_h) = true. The proofs for safety and termination follows immediately from Theorem 14 and Theorem 15. □

Lemma 17 (Communication Complexity)**.** *Let ℓ be the size of the input, κ be the size of accumulator, and w be the size of witness. The communication complexity of the MVBA protocol is $O(n\ell + (\kappa + w)n^2))$ in expectation.*

Proof. In the proposal dispersal phase, each party sends a code word of size $O(\ell/n)$, a witness of size w and an accumulator of size κ to all other parties. In addition, each party sends κ-sized ack message to all other parties. Thus, this phase incurs $O(n\ell + (\kappa + w)n^2)$ communication.

In the protocol in Figure 6, the status step incurs $O(\kappa n^2)$ as each party sends $O(\kappa)$-sized threshold signature to all other parties. The propose step also incurs $O(\kappa n^2)$ communication. The leader election phase in round 3 incurs $O(\kappa n^2)$ communication. In the Forward step (round 4) each party multicasts code word of size $O(\ell/n)$, witness of size w bits, accumulator of size $O(\kappa)$ bits with a total communication complexity of $O(n\ell + (\kappa + w)n^2)$ bits. Similarly, in Decode step and Forward2, each party sends code word of size $O(\ell/n)$, witness of size w bits, accumulator of size $O(\kappa)$ bits with a total communication complexity of $O(n\ell + (\kappa + w)n^2)$ bits.

In Vote step, each party multicasts $O(\kappa)$-sized vote message to all other parties, this incurs $O(\kappa n^2)$ communication. In the commit step, each party multicasts $O(\kappa)$-sized vote certificate and $O(\kappa)$-sized terminate messages. All-to-all multicast of $O(\kappa)$-sized termination certificate also incurs $O(\kappa n^2)$ communication. Thus, the protocol incurs $O(n\ell + (\kappa + w)n^2))$ communication in an epoch.

Note that the protocol terminates in expected constant epochs. Thus, the communication complexity of the protocol is $O(n\ell + (\kappa + w)n^2))$ in expectation. □

6 Parallel Broadcast

Finally, we present two communication efficient parallel broadcast protocols tolerating $t < n/2$ Byzantine faults with a communication complexity of $O(n^2\ell + \kappa n^3)$ for input of size ℓ bits and expected $O(1)$ rounds under various setup assumptions. The first protocol is in the authenticated model with PKI and digital signatures. It is secure against a static adversary. The second protocol is secure against an adaptive adversary, but assumes threshold setup and uses adaptively-secure threshold signature scheme.

We present parallel broadcast protocols with expected constant rounds in Figure 7. In this protocol, each party P_i first uses graded parallel broadcast to propagate their input v_i and output a n-element list of values along with a n-element grade list $\mathsf{GradeList}_i$ accompanied by $\mathcal{AC}(\mathsf{GradeList}_i)$. The tuple ($\mathsf{GradeList}_i$, $\mathcal{AC}(\mathsf{GradeList}_i)$) is then input to an MVBA protocol to agree on a common certified $\mathsf{GradeList}_h$. The ack certificate on GradeList serves as the external validity function. Parties then output $\mathcal{V}$ with $\mathcal{V}[j] = v_j$ if $\mathsf{GradeList}_h[j] \in \{3, 4\}\ \forall j \in [n]$. We give two variants of the protocol depending upon the MVBA protocol being considered.

1. **Graded parallel broadcast.** Each party P_i invokes graded parallel broadcast protocol (refer Figure 4) with its input v_i and outputs an n element list of values along with ($\mathsf{GradeList}_i$, $\mathcal{AC}(\mathsf{GradeList}_i)$).
2. **MVBA.** Each party P_i participates in MVBA with input $\mathsf{GradeList}_i$ and $\mathcal{AC}(\mathsf{GradeList}_i)$. Let $\mathsf{GradeList}_h$ be the output of the MVBA protocol.
3. **Output.** Set $\mathcal{V}[j] = v_j$ if $\mathsf{GradeList}_h[j] \in \{3, 4\}\ \forall j \in [n]$. Output $\mathcal{V}$.

Figure 7: Parallel broadcast with $O(n^2\ell + \kappa n^3)$ communication and expected $O(1)$ rounds

Using MVBA protocol of Shrestha et al. [32]. In Shrestha et al. [32], they gave an MVBA protocol in the authenticated model with PKI and digital signatures with security against a static adversary. Their protocol incurs $O(\kappa n^3)$ communication in expectation when $\ell = O(n)$ (i.e., the size of $(\mathsf{GradeList}, \mathcal{AC}(\mathsf{GradeList}))$ and terminates in expected $O(1)$ rounds. The exact round complexity of their MVBA protocol is expected 36 rounds. We refer the readers to Shrestha et al. [32] for more details. Using this MVBA protocol, gives us a parallel broadcast protocol secure against static adversary in the authenticated model with PKI and digital signatures. The resulting parallel broadcast protocol has $O(n^2\ell) + E(O(\kappa n^3))$ communication and terminates in expected constant rounds. Concretely, this protocol terminates in expected 47 rounds.

Using MVBA from Section 5. In the second variant, we make use of our MVBA protocol from Section 5. Using our MVBA protocol, gives us a parallel broadcast protocol secure against a (strongly rushing) adaptive adversary. The graded parallel broadcast protocol has a communication complexity of $O(n^2\ell + \kappa n^3)$ and the MVBA protocol has a communication complexity of $O(\kappa n^2)$ when $\ell = O(n)$ (the size of $(\mathsf{GradeList}, \mathcal{AC}(\mathsf{GradeList}))$. Thus, the resulting parallel broadcast protocol will have $O(n^2\ell + \kappa n^3) + E(O(\kappa n^2))$ communication and terminates in expected $O(1)$ rounds. Concretely, this protocol terminates in expected 30 rounds.

6.1 Security Analysis of Parallel Broadcast

Theorem 18. *The protocol in Figure 7 is a parallel broadcast protocol satisfying Definition 2.1.*

Proof. By Theorem 16, all honest parties eventually terminate with a common $\mathsf{GradeList}_h$ where external validity function is presence of $\mathcal{AC}(\mathsf{GradeList}_h)$. Termination follows from termination property of the underlying MVBA protocol.

By the properties of graded parallel broadcast(Theorem 10), all honest parties receive the same value v_j such that $\mathsf{GradeList}_h[j] \in \{3, 4\}$. Since, all honest parties compute final vector $\mathcal{V}$ based on common $\mathsf{GradeList}_h$. Thus, agreement holds.

In addition, the grades corresponding to honest parties in $\mathsf{GradeList}_h$ are in the range {3,4}. Thus, validity holds. □

7 Related Work

7.1 Related Work in Parallel Broadcast Literature

The problem of parallel broadcast (aka, interactive consistency [30]) was originally introduced by Pease et al [30]. In the same work, they show two variants of the protocol (i) a protocol with $t < n/3$ resilience in the *plain* authenticated model or unauthenticated model, and (ii) a protocol with $t < n$ resilience in the authenticated model with authenticators. Both of their protocols had exponential communication complexity and $\Theta(t)$ round complexity.

Ben-or and El-Yaniv [7] showed how to achieve expected $O(1)$ rounds for the interactive consistency problem tolerating $t < n/3$ Byzantine faults in the plain authenticated model. In their solution, they invoked $O(n \log n)$ instances of the BA protocol due to Feldman and Micali [18] in a "black-box" fashion to achieve expected $O(1)$ round parallel broadcast protocol. Their construction has a very high communication as each instance of BA protocol of Feldman and Micali [18] has $O(n^6 \log n)$ communication (without q-SDH setup assumption) even for a single bit.

Very recently, Abraham et al. [1] gave an efficient protocol in the plain authenticated model tolerating $t < n/3$ Byzantine faults and security against an adaptive adversary. Their protocol incurs $O(n^2\ell + n^4 \log n)$ communication (without q-SDH setup assumption) in expectation for input of size ℓ bits and expected $O(1)$ rounds.

In the authenticated model with PKI and digital signatures, the notion of parallel broadcast was recently explored by Tsimos et al. [33]. They show two variants of the protocol each tolerating $t < (1-\varepsilon)n$ Byzantine faults and security against an adaptive adversary. The first protocol works in the authenticated model with

PKI and digital signatures and incurs $\tilde{O}(\kappa^2 n^3)$ communication for single bit input and $O(t \log t)$ rounds. Their second protocol has stronger setup assumptions. In particular, they require a trusted dealer to setup the keys and relies on *bit-specific* committee election [3] to reduce communication. In addition, their protocol requires parties to erase their signatures once a message has been sent. Their protocol incurs $\tilde{O}(\kappa^4 n^2)$ communication for single bit input and $O(\kappa \log t)$ rounds.

Closely related technique. In a recent work [1], Abraham et al. gave a parallel broadcast protocol in the unauthenticated model tolerating $t < n/3$ Byzantine faults with a communication complexity of $O(n^2\ell) + E(O(n^4 \log n))$ and termination in expected $O(1)$ rounds. Their protocol relies on the idea of Fitzi and Garay [19] where multiple BA sub-protocols are run in parallel when only a single leader election is invoked per iteration for all the sub-protocols. In their construction, each party first propagates their ℓ bit input via a gradecast protocol where each gradecast invocation costs $O(n\ell + n^3 \log n)$; the total communication complexity of n parallel gradecast is $O(n^2\ell + n^4 \log n)$. It is followed by parallel invocation of n instances of BA protocol where each BA protocol has a communication complexity of $O(n^3 \log n)$ bits for a single bit input. In addition, their leader election protocol has a communication complexity of $O(n^4 \log n)$ bits. The resulting protocol has a communication complexity of $O(n^2\ell) + E(O(n^4 \log n))$ for input of size ℓ bits.

We note that their technique is relevant but not sufficient to achieve our goal. In the authenticated model with PKI and digital signatures, to the best of our knowledge, the MVBA protocol due to Shrestha et al. [32], when used as a BA protocol, is the most efficient protocol in the setting which has a communication complexity of $O(\kappa n^3)$ in expectation and termination in expected constant rounds. Parallel invocation of $O(n)$ instances of this BA protocol would result in $O(\kappa n^4)$ communication in each round. In contrast, our parallel broadcast in the setting incurs $O(n^2\ell) + E(O(\kappa n^3))$ communication.

With threshold setup assumption, to the best of our knowledge, the BA protocol due to Abraham et al. [4] is the most efficient protocol which has a communication complexity of $O(\kappa n^2)$ in expectation and termination in expected constant rounds. Following the technique of Abraham et al. [1], we can use M-Gradecast$(., 2)$ to propagate ℓ bit input at the total communication complexity of $O(n^2\ell + \kappa n^3)$. Then, parallel invocation of $O(n)$ instances of binary BA protocol due to Abraham et al. [4] along with a single leader election protocol across all BA instances will result in expected $O(\kappa n^3)$ communication and termination in expected constant rounds. The total communication complexity of the protocol following their technique is $O(n^2\ell) + E(O(\kappa n^3))$ and termination in expected constant rounds. In the same setting, our protocol incurs $O(n^2\ell + \kappa n^3) + E(O(\kappa n^2))$ communication and expected constant rounds. In the worst case, when the protocol runs for linear number of rounds, the protocol following their technique would incur $O(n^2\ell + \kappa n^4)$ communication, while our protocol incurs $O(n^2\ell + \kappa n^3)$ communication.

7.2 Related Works in MVBA Literature

Multi-valued validated Byzantine agreement was first introduced by Cachin et al. [13] to allow honest parties to agree on any externally valid values. Their protocol works in asynchronous communication model and has optimal $t < n/3$ resilience with $O(n^2\ell + \kappa n^2 + n^3)$ communication for input of size ℓ. Later, Abraham et al. [5] gave an MVBA protocol with optimal resilience and $O(n^2\ell + \kappa n^2)$ communication in the same asynchronous setting. Lu et al. [26] extended the work of Abraham et al. [5] to handle long messages of size ℓ with a communication complexity of $O(n\ell + \kappa n^2)$. All of these protocols assume threshold setup, are secure against an adaptive adversary and terminate in expected $O(1)$ rounds. We provide technical differences with MVBA protocol of Lu et al. [26].

Comparison with MVBA protocol of Lu et al. [26]. In the MVBA protocol due to Lu et al. [26], they use $(t+1, n)$ RS codes with $t < n/3$ to distribute ℓ bit proposal during proposal dispersal phase. In their protocol, they collect an ack-cert consisting of $2t+1$ ack messages. If there is an ack-cert for a proposal, this implies at least $t+1$ honest parties have received valid code words for the proposal. This is sufficient to decode the proposal since the protocol uses $(t+1, n)$ RS codes. Thus, in their protocol, ack-cert for a proposal implies honest parties have sufficient valid code words to decode the original proposal and honest parties can agree on any proposal with an ack-cert. This is in contrast to our protocol since honest parties may not

be able to decode the proposal even though the proposal is accompanied by an ack-cert. Our protocol relies on synchrony to filter out such "bad" proposals.

Comparison with MVBA protocol of Shrestha et al. [32]. In the synchronous setting, Shrestha et al. [32] gave the first MVBA protocol tolerating $t < n/2$ Byzantine faults secure against static adversary. Their protocol works in the plain PKI model without threshold setup and incurs $O(n^2\ell + \kappa n^3)$ for inputs of size ℓ and expected constant rounds. In this work, we present an MVBA protocol with better communication and security against a strongly rushing adaptive adversary. Our protocol requires threshold setup and incurs $O(n\ell + \kappa n^2)$ for inputs of size ℓ bits and terminates in expected $O(1)$ rounds.

7.3 Related Work in the Gradecast with Multiple Grades Literature

Gradecast with multiple grades was initially introduced by Garay et al. [22]. They gave a protocol in the authenticated model with PKI model tolerating $t < n$ Byzantine failures with a message complexity of $O(g^* n^2)$ and a communication complexity of $O(g^*(n\ell + \kappa n^2))$ for ℓ bit input. A recent work [21] also gave a gradecast with multiple grades under the notion of *Proxcast*. Their protocol tolerates $t < n$ Byzantine faults and a message complexity of $O(g^* n^2)$. This corresponds to a communication complexity of $O(g^*(n\ell + \kappa n^2))$ for ℓ bit input. In this work, we present a slightly relaxed definition for gradecast with multiple grades and provide a construction with a communication complexity of $O(n\ell + \kappa n^2)$ for ℓ bit input tolerating $t < (1-\varepsilon)n$ Byzantine faults where $\varepsilon > 0$ is a constant.

8 Acknowledgments

We thank Adithya Bhat and Aniket Kate for helpful discussions related to this paper.

References

[1] 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, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I*, pages 384–414. Springer, 2023.

[2] Ittai Abraham, Gilad Asharov, and Avishay Yanai. Efficient perfectly secure computation with optimal resilience. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19*, pages 66–96. Springer, 2021.

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

[4] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected O(1) rounds, expected communication, and optimal resilience. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*, pages 320–334. Springer, 2019.

[5] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

[6] Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round mpc with identifiable abort and public verifiability. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II*, pages 562–592. Springer, 2020.

[7] Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[8] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351–371. 2019.

[9] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE, 2015.

[10] Adithya Bhat, Nibesh Shrestha, Zhongtang Luo, Aniket Kate, and Kartik Nayak. Randpiper–reconfiguration-friendly random beacons with quadratic communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3502–3524, 2021.

[11] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of cryptology*, 21(2):149–177, 2008.

[12] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. *Cryptology ePrint Archive*, 2017.

[13] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Annual International Cryptology Conference*, pages 524–541. Springer, 2001.

[14] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 98–116. Springer, 1999.

[15] Ran Cohen, Sandro Coretti, Juan Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III 36*, pages 240–269. Springer, 2016.

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

[17] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.

[18] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 148–161, 1988.

[19] Matthias Fitzi and Juan A Garay. Efficient player-optimal protocols for strong and differential consensus. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 211–220, 2003.

[20] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued byzantine agreement. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 163–168, 2006.

[21] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 355–362, 2021.

[22] Juan A Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 658–668. IEEE, 2007.

[23] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *CRYPTO*, volume 4117, pages 445–462. Springer, 2006.

[24] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 203–212, 2002.

[25] Julian Loss and Tal Moran. Combining asynchronous and synchronous byzantine agreement: The best of both worlds. *Cryptology ePrint Archive*, 2018.

[26] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th symposium on principles of distributed computing*, pages 129–138, 2020.

[27] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[28] 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)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[29] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Cryptographers' track at the RSA conference*, pages 275–292. Springer, 2005.

[30] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[31] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[32] Nibesh Shrestha, Adithya Bhat, Aniket Kate, and Kartik Nayak. Synchronous distributed key generation without broadcasts. *Cryptology ePrint Archive*, 2021.

[33] Georgios Tsimos, Julian Loss, and Charalampos Papamanthou. Gossiping for communication-efficient broadcast. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part III*, pages 439–469. Springer, 2022.