Balanced Byzantine Reliable Broadcast with Near-Optimal Communication and Improved Computation[*]

NICOLAS ALHADDAD, Boston University, USA
SOURAV DAS, University of Illinois at Urbana-Champaign, USA
SISI DUAN, Tsinghua University, China
LING REN, University of Illinois at Urbana-Champaign, USA
MAYANK VARIA, Boston University, USA
ZHUOLUN XIANG, University of Illinois at Urbana-Champaign, USA
HAIBIN ZHANG, Beijing Institute of Technology, China

This paper studies Byzantine reliable broadcast (BRB) under asynchronous networks, and improves the state-of-the-art protocols from the following aspects. *Near-optimal communication cost:* We propose two new BRB protocols for n nodes and input message M that has communication cost $O(n|M|+n^2 \log n)$, which is near-optimal due to the lower bound of $\Omega(n|M|+n^2)$. The first RBC protocol assumes threshold signature but is easy to understand, while the second RBC protocol is error-free but less intuitive. *Improved computation:* We propose a new construction that improves the computation cost of the state-of-the-art BRB by avoiding the expensive online error correction on the input message, while achieving the same communication cost. *Balanced communication:* We propose a technique named balanced multicast that can balance the communication cost for BRB protocols where the broadcaster needs to multicast the message M while other nodes only needs to multicast coded fragments of size $O(|M|/n + \log n)$. The balanced multicast technique can be applied to many existing BRB protocols as well as all our new constructions in this paper, and can make every node incur about the same communication cost. Finally, we present a lower bound to show the near optimality of our protocol in terms of communication cost at each node.

1 INTRODUCTION

Reliable broadcast is a fundamental primitive in distributed computing [14], and has many applications such as fault-tolerant consensus and replication [24, 26, 27, 33, 35, 37, 40], secure multiparty computation [38, 50], verifiable secret sharing [21], and distributed key generation [2, 22, 36]. The goal of reliable broadcast is to have a designated broadcaster send its input message and to have all nodes output the same message.

In this paper, we assume Byzantine faults that may deviate arbitrarily from the protocols. The problem is hence referred to us Byzantine Reliable Broadcast (BRB). We study BRB in asynchronous networks.

Existing works on BRB. The first BRB protocol due to Bracha [14] has a total communication cost of $O(n^2|M|)$, where n is the number of protocol nodes and $|M|$ is the size of the broadcaster's message in bits. The BRB protocol of Bracha [14] is error-free, which means the adversary has unbounded computational power and the protocol always achieves the guarantees in every possible execution. Two decades later, it is improved by Cachin and Tessaro [16] to $O(n|M|+\kappa n^2 \log n)$, assuming a computationally bounded adversary that cannot break a collision resistant hash function of output size $O(\kappa)$. Later, Patra [42] proposed an error-free BRB protocol with improved total communication

[*]This publication is a merge of the following ePrint papers [6, 20, 25]. All authors contributed equally to this work and are listed alphabetically. Corresponding authors: Sisi Duan and Haibin Zhang.

Authors' addresses: Nicolas Alhaddad, Boston University, USA, nhaddad@bu.edu; Sourav Das, University of Illinois at Urbana-Champaign, USA, souravd2@illinois.edu; Sisi Duan, Tsinghua University, China, duansisi@tsinghua.edu.cn; Ling Ren, University of Illinois at Urbana-Champaign, USA, renling@illinois.edu; Mayank Varia, Boston University, USA, varia@bu.edu; Zhuolun Xiang, University of Illinois at Urbana-Champaign, USA, xiangzl@illinois.edu; Haibin Zhang, Beijing Institute of Technology, China, haibin@bit.edu.cn.

cost $O(n|M|+n^4 \log n)$ and has 11 rounds. In all of these BRB protocols, every node, including the broadcaster, incurs the same asymptotic communication cost. Here on, we say such a BRB protocol has a *balanced* communication cost. Very recently, several efforts [5, 21, 41] further improve the communication cost of the BRB protocols. The state-of-the-art cryptographically secure asynchronous BRB protocol is due to Das, Xiang, and Ren [21], which has a total communication cost of $O(n|M|+\kappa n^2)$, incurs 4 rounds and requires collision resistant hash functions. The state-of-the-art error-free asynchronous BRB protocol [41] has a total communication cost of $O(n|M|+n^3 \log n)$ and incurs 7 rounds. We provide a detailed comparison in Table 1 and discuss other related work in detail in §8.

Limitations of the state-of-the-art BRB protocols. However, despite all the improvements made by the above efforts, the state-of-the-art BRB protocols still have the several limitations.

Gap on the communication cost. As shown in [41], a straightforward communication complexity lower bound for BRB is $\Omega(n|M|+n^2)$, since all $O(n)$ honest nodes eventually need to receive M, and even single-bit BRB incurs $\Omega(n^2)$ [23]. However, the BRB protocol of Das, Xiang, and Ren [21] still has cost $O(n|M|+\kappa n^2)$, which has a $O(\kappa)$ gap from the lower bound. The state-of-the-art error-free BRB protocol [41] has cost $O(n|M|+n^3 \log n)$, a gap of $O(n \log n)$ from the optimal. Hence, a natural question is, can we further reduce the communication complexity of the BRB towards its optimality?

Unbalanced communication cost. As mentioned earlier, the BRB protocols like Bracha [14] and Cachin and Tessaro [16] have balanced communication cost, meaning every node, including the broadcaster, incurs the same asymptotic cost. However, the the state-of-the-art BRB protocols [21, 41] have *unbalanced* communication cost. In both protocols, the communication cost of the broadcaster is approximately n times higher than that of other nodes, leading to a bottleneck at the broadcaster. It is because the broadcaster in these protocols needs to send the input message M to all nodes, while other node only needs to exchange the encoded piece of the message which has size $O(|M|/n + \log n)$. In practice, such unbalanced construction may introduce performance bottlenecks in the system, given that BRB serves as a fundamental primitive in many applications. Therefore, an important question is whether we can design a BRB protocol that achieves balanced and near-optimal communication cost for every node?

Inefficiency in the computation. Another limitation of the BRB protocol due to Das, Xiang, and Ren [21] is the computation inefficiency caused by the online error correction (OEC) algorithm [7]. OEC is a decoding algorithm for error correcting codes (ECC) such as Reed-Solomon codes [45], where the original message can be decoded even if some of the coded fragments are corrupted. Roughly speaking, the OEC will decode up to t times where t is the threshold of Byzantine faults, for a set of enough coded fragments each time when the set grows. Das, Xiang, and Ren [21] cannot avoid such a computation since in their BRB protocol, there is no proofs attached to the coded fragments for validity checks. In contrast, the protocol of Cachin and Tessaro [16] does not require OEC as their protocol attaches the Merkle proof for each fragments. As a result, Cachin and Tessaro [16] can use erasure code and only need to perform the decoding once, thus incurs a decoding computation cost at least $O(n)$ times cheaper than Das, Xiang, and Ren [21]. Thus, an interesting question is whether we can improve the computation cost of Das, Xiang, and Ren [21] while preserving the same communication cost?

Our contributions. We present several BRB protocol constructions that resolve the existing limitations and answer the above questions affirmatively. In summary, this paper has the following contributions.

Balanced multicast (§3). We first present a technique that can balance the communication cost of any BRB protocols, where the broadcaster needs to multicast the message M resulting in unbalanced $O(n|M|)$ communication at the broadcaster. Our balanced multicast requires two asynchronous rounds, achieves the same property as the vanilla multicast, but additionally guarantees that all the nodes incur the same cost of $O(|M|+n\log n)$. The main technique to achieve balanced communication cost is to use an additional round of interaction between nodes to help them reconstruct the input message of the broadcaster without having the broadcaster to directly send the input to all nodes. As a result, existing unbalanced communication BRB protocols [21, 41] can be made balanced by replacing the multicast of M with our balanced multicast in a black-box manner. We also apply the same technique to all BRB protocols proposed in this paper to make them balanced.

BRB protocol with improved computation (§4). We propose a computationally efficient BRB protocol that reduces the computation of [21], while keeping the same communication complexity $O(n|M|+\kappa n^2)$. The main idea of the construction is to replace error correcting codes with erasure codes as much as possible. We perform erasure coding on the bulk data M, and then apply error correction only to the vector of hashes of those fragments (of length κn). To agree on the hash vector, all honest nodes will execute Bracha-style BRB on the hash of the hash vector, and perform OEC to reconstruct the vector. Asymptotically, when the size of the message is larger than $O(\kappa n)$, the above technique reduces the computation cost at each node for decoding from $\tilde{O}(n|M|)$ to $\tilde{O}(|M|+\kappa n^2)$, where $\tilde{O}(\cdot)$ hides poly-logarithmic terms. Additionally, erasure coding is concretely more efficient than error correction because optimized implementations use a static field size for all n [43] and they operate using only fast xor and shift operations [30].

BRB protocols with near-optimal communication (§5, §6). We further reduce the communication complexity gap between the lower and upper bound for BRB, by constructing two protocols that have near-optimal communication cost. Our first BRB protocol is called SigBRB, which is inspired by Das, Xiang, and Ren [21], and further reduces the cost to $O(n|M|+\kappa n + n^2\log n)$ with threshold signature. Since usually we can assume $n = poly(\kappa)$, the $O(\kappa n)$ term can be neglected and thus the cost is only $O(\log n)$ from the lower bound. SigBRB breaks the $O(\kappa n^2)$ cost barrier of [21] by replacing its (quadratic) Bracha-style reliable broadcast on hash with a (linear) consistent broadcast on threshold signatures. A novel amplification step is also introduced to ensure the correctness after such modification. Moreover, our SigBRB protocol is intuitive and easy to understand. Our second BRB protocol is error-free, and has total communication cost to $O(n|M|+n^2\log n)$, compared to $O(n|M|+n^3\log n)$ communication of current art [41]. Our error-free BRB protocol builds on top of the recent synchronous error-free Byzantine agreement (BA) protocol of Chen [19]. We make several subtle and important modifications to their BA protocol to accommodate asynchrony (see §6). The error-free protocol has clear advantages in terms of communication cost, cryptographic and setup assumptions, but is less intuitive and require more rounds than SigBRB, and much harder to analyze compared to the first protocol SigBRB.

Lower bound. Another contribution of this paper is a lower bound result (§7). We prove that for any deterministic BRB protocol, each node incurs a communication cost of $\Omega(|M|+n)$. Hence, our SigBRB and EFBRB (also BalSigBRB and BalEFBRB) have near-optimal communication costs – only an factor of $O(\log n)$ gap from the lower bounds.

Paper organization. The rest of the paper is organized as follows. In §2 we provide the necessary background. In §3 we discuss our balanced multicast protocol where the multicaster incurs the same bandwidth cost as other nodes. In §4 we describe our computationally improved BRB protocol. In §5 and §6, we describe our BRB protocols with near-optimal communication complexity, one assuming threshold signature and one is error-free, respectively. In §7 we show lower bounds on the communication cost of BRB. We discuss related work in §8 and conclude in §10.

Table 1. Comparison with existing BRB protocols. The computation cost measures the coding and crytographic operations, and $\tilde{O}(\cdot)$ hides the poly-logarithmic terms (more details in §2.4). The following acronyms are used in the table; q-SDH: q-Strong Diffie-Hellman, DBDH: Decisional Bilinear Diffie-Hellman. *The protocol of [1] is statistically secure with probability $1 - \epsilon$.

Scheme	Communication Cost			Computation Cost	Rounds	Cryptographic	Setup
	Broadcaster	Other node	Total	Per-node		Assumption	
Bracha [14]	$O(n\|M\|)$	$O(n\|M\|)$	$O(n^2\|M\|)$	0	4	None (error-free)	None
Patra [42]	$O(n\|M\|+n^3 \log n)$	$O(\|M\|+n^3 \log n)$	$O(n\|M\|+n^4 \log n)$	$\tilde{O}(\|M\|)$	11	None (error-free)	None
Nayak et al. [41]	$O(n\|M\|+n^2 \log n)$	$O(\|M\|+n^2 \log n)$	$O(n\|M\|+n^3 \log n)$	$\tilde{O}(\|M\|)$	7	None (error-free)	None
Abraham-Asharov [1]*	$O(\|M\|+n \log n)$	$O(\|M\|+n \log(n^3/\epsilon))$	$O(n\|M\|+n^2 \log(n^3/\epsilon))$	$\tilde{O}(n\|M\|)$	7	None (statistical)	None
EFBRB (§6)	$O(n\|M\|+n \log n)$	$O(\|M\|+n \log n)$	$O(n\|M\|+n^2 \log n)$	$\tilde{O}(n\|M\|)$	9	None (error-free)	None
BalEFBRB (§6)	$O(\|M\|+n \log n)$	$O(\|M\|+n \log n)$	$O(n\|M\|+n^2 \log n)$	$\tilde{O}(n\|M\|)$	10	None (error-free)	None
Cachin-Tessaro [16]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	$O(n\|M\|+\kappa n^2 \log n)$	$\tilde{O}(\|M\|+\kappa n)$	4	Hash	None
Das et al. [21]	$O(n\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	$\tilde{O}(n\|M\|)$	4	Hash	None
CCBRB (§4)	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	$\tilde{O}(\|M\|+\kappa n^2)$	4	Hash	None
BalCCBRB (§4)	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	$\tilde{O}(\|M\|+\kappa n^2)$	5	Hash	None
Nayak et al. [41]	$O(n\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	$\tilde{O}(\|M\|)$	7	q-SDH+DBDH	Trusted
SigBRB (§5)	$O(n\|M\|+\kappa n + n \log n)$	$O(\|M\|+\kappa + n \log n)$	$O(n\|M\|+\kappa n + n^2 \log n)$	$\tilde{O}(n\|M\|)$	7	Threshold Sig	Trusted
BalSigBRB (§5)	$O(\|M\|+\kappa n + n \log n)$	$O(\|M\|+\kappa + n \log n)$	$O(n\|M\|+\kappa n + n^2 \log n)$	$\tilde{O}(n\|M\|)$	8	Threshold Sig	Trusted
Lower bound	$\Omega(\|M\|+n)$	$\Omega(\|M\|+n)$	$\Omega(n\|M\|+n^2)$	–	2 [3]	–	–

2 PRELIMINARIES

2.1 System Model

We consider a network of n nodes where every pair of nodes is connected via a pairwise authenticated channel. We consider the presence of a malicious adversary $\mathcal{A}$ that can corrupt up to t nodes in the network. The corrupted (faulty) nodes can behave arbitrarily, and we call a node honest (correct) if it remains non-faulty for the entire protocol execution. We assume the network is asynchronous, i.e., $\mathcal{A}$ can arbitrarily delay any message but must eventually deliver all messages sent between honest nodes. A protocol is error-free if it is secure against any computationally unbounded adversary in all executions.

We use $|S|$ to denote the size of a set S. For any integer a, we use $[a]$ to denote the set $\{1, 2, \ldots, a\}$. We use κ to denote the size of the output of the collision-resistant hash function. Naturally, we assume that $\kappa > \log n$.

Rounds and phases. Under asynchrony, the network delay is unbounded. Therefore, to measure the latency of asynchronous protocols, we use the standard notion of *asynchronous rounds* [17], where a protocol runs in R asynchronous rounds if its running time is at most R times the maximum message delay between honest parties during the execution. We also use the notion of phases for ease of description, where a phase consists of a fixed number of rounds. When describing some of our protocols, we may divide a protocol into several phases, each of which has several rounds.

Identifying protocol instances. We assume each protocol instance is associated with a unique tag *id*. For some of our simpler protocols, when there is no ambiguity, we may simply drop the tag *id*. Moreover, to help readers better understand some of our protocols, we may provide a unique round name (e.g., `BCB-send`, `disperse`).

2.2 Problem Formulations

Definition 1 (Reliable Broadcast [14]). A protocol for a set of nodes $\{1, \ldots, n\}$, where a designated broadcaster holds an input M, is a reliable broadcast protocol, if the following properties hold

- *Agreement:* If an honest node outputs a message M' and another honest node outputs M'', then $M' = M''$.
- *Validity:* If the broadcaster is honest, all honest nodes eventually output the message M.

- *Totality:* If an honest node outputs a message, then every honest node eventually outputs a message.

2.3 Primitives

Erasure Code and Error Correcting Code. State of the art BRB protocols use erasure codes or error correcting codes. Both can encode the data block into fragments. Erasure codes tolerate erasures (unavailable fragments), while error correction codes tolerate errors (incorrect/corrupt fragments). In general, error correcting codes have more restricted syntax and more expensive operations.

An (m, n) erasure coding scheme over an alphabet $\mathcal{M}$ is a pair of algorithms $(\mathsf{ECEnc}, \mathsf{ECDec})$, where $\mathcal{M}$ denotes the alphabet for a single fragment, $\mathsf{ECEnc} : \mathcal{M}^{\mathsf{m}} \rightarrow \mathcal{M}^{\mathsf{n}}$ and $\mathsf{ECDec} : \mathcal{M}^{\mathsf{m}} \rightarrow \mathcal{M}^{\mathsf{m}}$. The ECEnc algorithm takes as input a data block, consisting of m data fragments, and outputs $n > m$ coded fragments. The ECDec algorithm takes as input any m-size subset of coded fragments and outputs the original data block containing m data fragments. Namely, if $[d_1, \ldots, d_n] \leftarrow \mathsf{ECEnc}(\mathsf{M})$, then $\mathsf{ECDec}(\mathsf{d}_{\mathsf{i}_1}, \ldots, \mathsf{d}_{\mathsf{i}_\mathsf{m}}) = \mathsf{M}$ for any distinct $i_1, \ldots, i_m \in [1..n]$.

An (m, n) error correcting coding scheme is a pair of algorithms $(\mathsf{ECCEnc}, \mathsf{ECCDec})$. Let $\mathsf{ECCEnc}(M, m, k)$ denote the encoding scheme. $\mathsf{ECCEnc}(M, m, k)$ takes as input a message M consisting of k symbols, treats it as a polynomial of degree $k - 1$ and outputs m evaluations of the corresponding polynomial. Let $\mathsf{ECCDec}(k, r, T)$ denote the decoding scheme. ECCDec takes as input a set of symbols T (some of which may be incorrect), and outputs a degree $k - 1$ polynomial, i.e., k symbols, by correcting up to r errors (incorrect symbols) in T. It is well-known that ECCDec can correct up to r errors in T and output the original message provided that $|T| \geq k + 2r$ [39].

For concreteness, we will use the standard Reed-Solomon (RS) error correcting codes [45], and zigzag or Cauchy RS erasure codes [30, 44]. Concrete instantiations of RS codes include the Berlekamp-Welch algorithm [49] and the Gao algorithm [29]. Both codes can be implemented over a Galois field $GF(2^{\mathsf{w}})$, where erasure codes require only linear (xor) operations and RS codes also require multiplication tables [43].

Note that for efficiency reasons it is better to work in smaller fields (where a is small). For a message M with $|M|$ denoting the number of bits of M, if $|M| > ma$, then M is broken into $|M|/(ma)$ polynomials, each of which has degree $m - 1$. A fragment i is then the set of all polynomial evaluations at i, i.e., $d_i = f_1(i) \ldots f_{|M|/ma}(i)$.

Online Error Correction. All of our BRB protocols use the Online-Error-Correction (OEC) protocol introduced by Ben-Or, Canetti, and Goldreich [7]. The OEC takes a set T consisting of tuples (j, a_j) where j is an index $j \in [n]$ and a_j is a fragment of a Reed-Solomon codeword. The OEC algorithm then tries to decode a message M such that Reed-Solomon encoding of M matches with at least $2t + 1$ elements in T. More specifically, the OEC algorithm performs up to t trials of reconstruction, and during the r-th trial, it uses $2t + r + 1$ elements in T to decode. If the reconstructed message M' whose encoding matches with at least $2t + 1$ tuples in T, the OEC algorithm successfully outputs the message; otherwise, it waits for one more fragment and tries again. We summarize the OEC algorithm in Algorithm 1. The OEC algorithm is error-free and information-theoretically secure against any adversary that corrupts up to t fragments among a total of $n \geq 3t + 1$ fragments.

Asynchronous data dissemination. Das, Xiang, and Ren [21] propose asynchronous data dissemination (ADD) which allow $t + 1$ correct nodes to disseminate a message to all correct nodes in an asynchronous network, where t is the upper bound on the number of faulty nodes in the system. The ADD construction introduced by Das, Xiang, and Ren has two rounds. In the first round, all nodes holding M send coded fragments to all nodes; in the second round, upon receiving $t + 1$ matching fragments d_i, a node i fixes its fragment as d_i and broadcasts d_i. Then nodes wait to receive fragments

Algorithm 1 Information Theoretic Online Error-correcting (IT-OEC) protocol

```
1: Input: $T$        // $T$ consisting of tuples $(j, a_j)$ where $j \in [n]$ and $a_j$ is a fragment
2: for $0 \leq r \leq t$ do        // online Error Correction
3:     Wait till $|T| \geq 2t + r + 1$
4:     Let $M := \mathsf{ECCDec}(t + 1, r, T)$
5:     Let $T' := \mathsf{ECCEnc}(M, m, t + 1)$
6:     if $2t + 1$ fragments in $T'$ match with $T$ then
7:         return $M$
```

and use OEC algorithm to decode the original block M. The above ADD construction is information-theoretic and does not use any cryptographic tools.

We next analyze the communication cost of their ADD protocol. Let $|M|$ be the size of the message M. Also, Reed-Solomon code require a field size of at least n, so each fragment has at least $\log n$ bits. This implies, each data fragment is $O(\max\{|M|/n, \log n\})$ bits, so an all-to-all exchange of the fragments results in $O(n^2 \cdot \max\{|M|/n, \log n\}) = O(n|M| + n^2 \log n)$ communication cost. Therefore, the total communication cost is $O(n|M| + n^2 \log n)$.[*]

Collision-resistant Hash Function. We use a cryptographic collision-resistant hash function hash, which guarantees that a computationally bounded adversary cannot come up with two inputs that hash to the same value, except for a negligible probability.

Signatures, threshold signatures, and multi-signatures. We use a conventional signature scheme consisting of three algorithms (*siggen*, *sigsign*, *sigverify*). *siggen* outputs a pair of public/secret keys (pk, sk). A signature signing algorithm *sigsign* takes as input a message M and a private key sk and outputs a signature σ. A signature verification algorithm *sigverify* takes as input pk, a message M, and a signature σ, and outputs a bit. We require the conventional unforgeability property for signatures.

A (ℓ, n) threshold signature scheme [8, 47] consists of the following algorithms (*tgen*, *tsign*, *shareverify*, *tcombine*, *tverfiy*). *tgen* outputs a system public key known to anyone and a vector of n private keys. A partial signature signing algorithm *tsign* takes as input a message M and a private key sk_i and outputs a partial signature σ_i. A combining algorithm *tcombine* takes as input pk, a message M, and a set of ℓ valid partial signatures, and outputs a signature σ. A signature verification algorithm *tverify* takes as input pk, a message M, and a signature σ, and outputs a bit. We require the conventional robustness and unforgeability properties for threshold signatures.

We simply omit the public keys, private keys, and key generation algorithms when no ambiguity arises. We may leave the verification of partial signatures and threshold signatures implicit when describing algorithms.

Multi-signature scheme [8, 11, 13] allows everyone to aggregate n signatures on the same message into one signature for the message. Given n signatures $\delta_i = sigsign(sk_i, M)$ on the same message M with public keys pk_i for $i \in [1..n]$, a multi-signature scheme can *multi-aggregate* the n signatures into a single signature δ, where $|\delta| = |\delta_i|$. The aggregated signature δ can be publicly verified using a verification function *multi-verify*$(pk_1, .., pk_n, \delta, M, L)$, where L is the list of n signers with public keys pk_i.

Note, above, we use σ and δ for threshold signatures and multi-signatures, respectively.

[*]The original ADD paper by Das, Xiang, and Ren [21] overlooked the Reed-Solomon fragment size and incorrectly claimed the communication cost as $O(n|M| + n^2)$.

Consistent broadcast. We review the definition of Byzantine consistent broadcast (BCB). Put it simply, consistent broadcast is reliable broadcast without the totality requirement. A protocol for a set of nodes $\{1, \ldots, n\}$, where a designated broadcaster holds an input M, is a consistent broadcast protocol, if the following properties hold

- *Agreement:* If an honest node outputs a message M' and another honest node outputs M'', then $M' = M''$.
- *Validity:* If the broadcaster is honest, all honest nodes eventually output the message M.

2.4 Metrics

Measurement of communication. We will measure the standard communication cost, defined as follows.

Definition 2 (Communication Cost)**.** The (total) communication cost of a protocol measures the total number of bits sent by all honest nodes during the execution of the protocol.

In addition to the standard communication cost above which measures the total cost of a protocol, we also measure the cost for each honest protocol node, as the per-node communication cost defined below.

Definition 3 (Per-node Communication Cost)**.** The communication cost of any honest protocol node p running a protocol measures the number of bits sent by p, and the number of bits p received from any other honest node, during the execution of the protocol. We say the protocol has per-node communication cost of C, if every honest node has communication cost at most C.

For instance, in the BRB protocol of Das, Xiang and Ren [21], the broadcaster incurs cost $O(n|M|)$ and any other node incurs cost $O(|M|+\kappa n)$, therefore the per-node communication cost of the protocol is $O(n|M|)$. Note that the total communication cost of a protocol equals the sum of communication costs of all honest nodes.

Definition 4 (Balanced Communication)**.** We say a protocol has balanced communication, if the per-node communication cost is $O(C/n)$ where C is the total communication cost of the protocol; otherwise, the protocol is unbalanced.

For instance, in the BRB protocol of Das, Xiang and Ren [21] has *unbalanced* communication cost, since the per-node communication cost of the protocol is $O(n|M|)$ and the total communication cost of the protocol is $O(n|M|+\kappa n^2)$. In contrast, using our balancing technique presented in §3, the per-node communication cost of [21] can be made balanced as $O(|M|+\kappa n)$.

Measurement of computation. In this paper, we will focus on the cost of coding operations and cryptographic operations when measuring the computation cost of our protocols. For the schemes defined in the previous section, we will list their costs as follows. We will use $\tilde{O}(\cdot)$ to hide the poly-logarithmic terms in the complexity.

- For both erasure codes and error correcting codes, encoding or decoding of a message M costs $\tilde{O}(|M|)$ each time. As mentioned, erasure codes will be concretely more efficient than error correcting codes but we do not distinguish them to keep things simple.
- For crytographic hash function, computing the hash for message M costs $\tilde{O}(|M|)$.
- For signatures, each signing or verification operation for message M costs $\tilde{O}(|M|)$.

3 BALANCING COMMUNICATION

In this section, we discuss a technique named *balanced multicast* that can compile an unbalanced BRB protocol where the broadcaster incurs higher communication cost than rest of the nodes, into a balanced one where every node incurs

Algorithm 2 Balanced Multicast (BalMC)

```
1:  // the broadcaster node invokes BalMC(M)
2:  input $M$
3:  Let $[m_1, m_2, \ldots, m_n]$ := ECCEnc($M, n, t + 1$)
4:  send ⟨PROPOSE, $m_j$⟩ to node $j$ for each $j \in [n]$
    // each node i
5:  Let $M := \perp$, $T := \{\}$
6:  upon receiving the first ⟨PROPOSE, $m_i$⟩ from the broadcaster do
7:      send ⟨SHARE, $m_i$⟩ to all nodes
8:  upon receiving the first ⟨SHARE, $m_j^*$⟩ from any node $j$ do
9:      $T := T \cup \{(j, m_j^*)\}$
10: Run IT-OEC on the set $T$
11: Let $M'$ be the output of IT-OEC($T$)
12: output $M'$ and return
```

the same asymptotic cost. For instance, in the BRB protocols of [21, 41, 42], the broadcaster incurs cost of at least $O(n|M|)$ due to sending the entire input message to all the nodes, and such cost can be made balanced through our technique. We first define balanced multicast as follows. Intuitively, it implements the standard multicast with balanced cost among all honest nodes.

Definition 5 (Multicast)**.** A protocol for a set of nodes $\{1, \ldots, n\}$, where a designated broadcaster holds an input M, is a multicast protocol, if the following property holds

- *Validity:* If the broadcaster is honest, all honest nodes eventually output the message M.

The *balanced* multicast further guarantees *balanced* communication cost: all honest nodes incur the same worst-case communication complexity asymptotically.

Challenges and our approaches. The state-of-the-art BRB protocol of Das, Xiang, and Ren [21] crucially uses the fact that the broadcaster sends its input message to all nodes at the start of the protocol. In their protocol, roughly speaking, nodes run Bracha's BRB on the hash digest of their message received from the broadcaster, and only exchange coded fragments of the message to reduce the communication cost to $O(n|M|+\kappa n^2)$. To ensure correctness, a node should BRB the digest and exchange the coded fragments of the same message. It is straightforward in the protocol of Das, Xiang, and Ren [21] since nodes directly receive the message from the broadcaster, which however incurs a cost of $O(n|M|)$ at the broadcaster. To reduce the cost, a natural idea would be let the broadcaster only send coded fragments of its message. Then, some kinds of proofs would be necessary to convince the nodes that the coded fragments are consistent with each other, otherwise the nodes can no longer run the erasure decoding protocol to recover the message. In fact, Cachin and Tessaro [16] obviates the need for the broadcaster to send its input to all via this approach. Specifically, the broadcaster encodes its input using an Erasure Code, computes a Merkle tree on the encoded fragments, and to each node, sends one encoded fragment and the corresponding Merkle path. A consequence of using a Merkle tree is that the resulting protocol has a communication cost of $O(n|M|+\kappa n^2 \log n)$, since the size of the Merkle path is $O(\kappa \log n)$ and there are all-to-all message exchanges with Merkle path attached. Similarly, Alhaddad et al. [5] uses a trusted setup-based constant size polynomial commitment scheme where the proof has size $O(\kappa)$ instead of the Merkle tree, to design a balanced BRB with a total communication cost of $O(n|M|+\kappa n^2)$. Omitting the Merkle tree or the polynomial commitment in a naive manner introduces the challenge that we can no longer run the erasure decoding protocol used by [5, 16] as there does not exist a way to distinguish an incorrect fragment from a correct one.

Our observation is that there is a simple way to balance the cost without attaching proofs with the coded fragments. The idea is to add one more communication round for the nodes to exchange their fragments received from the broadcaster and try Information-Theoretic Online Error Correction (IT-OEC) to reconstruct the broadcaster's message. If the broadcaster is honest, then IT-OEC can always recover the broadcaster's message. In the case of a malicious broadcaster, an honest node may not recover the message from IT-OEC. However, this is okay as the rest of our protocol guarantees that if any honest node output for the BRB, then there are enough honest nodes holding correct fragments that will send the fragments to all nodes for reconstruction. This simple idea turns out to be very useful, as we can abstract it as the *balanced multicast* primitive and apply it to many existing unbalanced BRB protocols [41, 42] and all protocols in this paper for balancing their cost.

Protocol description. In order to reduce the cost of the broadcaster node, our protocol BalMC first lets the broadcaster encode its message M into n fragments using a $(n, t+1)$ Reed-Solomon code (line 3) and only send the i-th fragment to node i. In particular, let $[m_1, m_2, \ldots, m_n] = \mathsf{ECCEnc}(M, n, t)$ be the RS encoding of M. Then, to node i, the broadcaster sends the message $\langle \mathsf{PROPOSE}, m_i \rangle$ (line 4). Note that due to properties RS code, each fragment has size $|M|/(t+1)$, and therefore the cost of the broadcaster is reduced to $O(n \cdot (|M|/(t+1))) = O(|M|)$.

Next, each node i upon receiving the $\langle \mathsf{PROPOSE}, m_i \rangle$ message from the broadcaster sends the $\langle \mathsf{SHARE}, m_i \rangle$ to all nodes (line 6-7). When a node receives a SHARE message from other nodes, it adds the corresponding fragment to the set T. Once enough fragments are collected, nodes use the Online Error Correcting (OEC) algorithm (line 10) to decode the message. As described in 2, intuitively, the OEC algorithm performs up to t trials of reconstruction, and during the r-th trial, a node uses $2t + r + 1$ fragments to decode. If the reconstructed message M' has the matches with at least $2t + 1$ tuples in T, a node successfully reconstructs the message; otherwise, it waits for one more fragment and tries again.

Applications. We can directly apply balanced multicast to existing unbalanced BRB protocols, such as [21, 41, 42]. As a concrete example, we will explain BalBRB, which is the BRB protocol of [21] after applying BalMC. In the first step of [21], the broadcaster sends M to all nodes, which is replaced by all nodes invoke BalMC(M) with the broadcaster inputting M. Then, when a node outputs M from BalMC, it invokes the steps as receiving M from the broadcaster in [21]. In this way, we balance the communication of [21] while keeping all its properties. It is also possible to apply balanced multicast to other fault-tolerant distributed protocols, for balancing the communication at the cost of adding one extra round of communication.

3.1 Analysis

Theorem 1 (Validity). *If the broadcaster node is honest, then all honest nodes eventually output the message M.*

Proof. When the broadcaster is honest and has input M, it sends the correct fragments to all nodes. Then, all honest nodes send the SHARE messages with the correct fragments. Thus, after receiving all SHARE message from honest nodes, any honest node can reconstruct M due to OEC. □

Theorem 2 (Performance). *Algorithm 2 solves balanced multicast with per-node communication cost of $O(|M| + n \log n)$, and per-node computation cost of $\tilde{O}(n|M|)$.*

Proof. In algorithm 2 the broadcaster sends a single PROPOSE to all other nodes. Moreover, each honest node sends a single SHARE message. Each message is $O(\max(|M|/n, \log n))$ bits long, since $|m_i| = \max(|M|/(t+1), \log n)$. Hence, each node incurs a per-node communication cost of $O(|M|+n \log n)$. Since each node invokes online error correction which will decode up to t times in the worst case, each node incurs a computation cost of $\tilde{O}(n|M|)$. □

4 CROSS-CHECKSUM BRB: REDUCING THE COST OF ERROR CORRECTION

In this section, we introduce a new reliable broadcast construction that improves computation for large messages of size $|M| \geq O(\kappa n)$, while maintaining its near-optimal communication complexity. This new construction, CCBRB, is compatible with our balancing technique from §3 at the cost of one extra round of communication. Also, here on we will refer to the BRB construction from Das, Xiang and Ren [21] as DXR BRB.

4.1 Improving computation vs. DXR BRB

An open problem left by Das, Xiang, and Ren in DXR BRB is related to the computational inefficiency problem of Reed-Solomon (RS) error-correcting codes (ECC) and the online error correction (OEC) algorithm. Specifically, in DXR BRB, each node needs to repeatedly run the OEC algorithm on the entire message. As we illustrate below, this results in large computation overhead when the messages are large.

Although not specified in [21], a natural way to implement ECC for a large message M, i.e., $|M| > a(t+1)$ bits, while ensuring that the field size a is independent of $|M|$, is to split M into $\frac{|M|}{a(t+1)}$ polynomials. Then, each fragment i would then be the concatenation of the evaluation of those polynomials at the same point i. However, such an approach is expensive for ECC. For a single polynomial of degree n, the standard error-correcting decoding algorithm has a run time complexity of $\tilde{O}(an)$ [29] for each one of the $\frac{|M|}{a(t+1)}$ polynomials. The run time becomes even more expensive when running OEC because then the ECC needs to repeated up to t times. This would bring the total run time complexity to $\tilde{O}(\frac{an^2|M|}{a(t+1)}) = \tilde{O}(n|M|)$. Note that OEC for a vector of small-sized messages outputs a result only if OEC is successful for each small-sized message.

In contrast, applying OEC only to the hash of the vector of messages brings down the OEC run time complexity from $\tilde{O}(n|M|)$ to $\tilde{O}(\kappa n^2)$. Also, using erasure coding on M incurs a cost of O(M) [30, 43]; it is faster than ECC as described in §1.

4.2 Cross-Checksum BRB

The idea of Cross-Checksum BRB is to limit the online error correcting to the cross checksum and not the message itself. The message itself would be encoded using erasure code rather than error correcting code. The BRB construction requires only 3 steps. It shares the structure of Bracha's broadcast and "combines" both approaches of DXR BRB [21] and Cachin-Tessaro BRB [16]. At a high level, the protocol uses the cross checksum of Cachin-Tessaro BRB [16] to send fragments of a message m (SEND phase and ECHO phase) but use the DXR BRB [21] approach to send fragments of the cross checksum itself (in the ECHO and READY phases). This approach will:

- avoid running ECC or OEC on the whole message M (like what DXR BRB [21] does), and
- avoid sending the whole cross checksum in the ECHO phase (like what Cachin-Tessaro BRB [16] does), otherwise we would obtain higher than κn^2 communication.

We use erasure coding to deal with bulk data, and use inefficient OEC only for n hashes. The minimized use of OEC and ECC makes DXR BRB practical for large-size messages.

We describe the pseudocode of CCBRB in Algorithm 4.3. Our BRB protocol, CCBRB, uses both standard erasure coding (ECEnc, ECDec) and Reed-Solomon ECC (ECCEnc, ECCDec). We also define a hash function: $H: \{0,1\}^* \rightarrow \{0,1\}^\kappa$.

Algorithm 3 CCBRB using hash functions with identifier id and sender s. Code shown for node i.

```
Initialization
    $fragments_{data} \leftarrow \perp$    //dictionary $(id, c) \mapsto$ list of fragments $d_j$
    $fragments_{hashes} \leftarrow \perp$    //dictionary $(id, c) \mapsto$ list of fragments $\pi_j$
    $e \leftarrow 0$    //number of errors to be corrected by the online error code
input $M$    //SEND round
    $\boldsymbol{d} \leftarrow \mathsf{ECEnc}(M)$, $D \leftarrow [H(d_1), \ldots, H(d_n)]$
    for $1 \leq j \leq n$, send $\langle id, \mathsf{SEND}, d_j, D\rangle$ to $j$
upon receiving $\langle id, \mathsf{SEND}, D, d_i\rangle$ from $s$ for first time do    //ECHO round
    if $H(d_i) = D_i$ then
        $c \leftarrow H(D)$, $\boldsymbol{\pi} \leftarrow \mathsf{ECCEnc}(D)$
        send $\langle id, \mathsf{ECHO}, (d_i, \pi_j, c)\rangle$ to node $j$
upon receiving $\langle id, \mathsf{ECHO}, (d_j, \pi_i, c)\rangle$ from node $j$ for first time do    // READY round
    $fragments_{data}[(id, c)] \leftarrow fragments[(id, c)] \cup [d_j]$
    if (not yet sent a READY message and received $2t + 1$ $\langle \mathsf{ECHO}\rangle$ messages with the same id, $c$ and same $\pi_i$)
        send $\langle id, \mathsf{READY}, c, \pi_i\rangle$ to all nodes
upon receiving $\langle id, \mathsf{READY}, c, \pi_j\rangle$ from node $j$ for the first time do    //verification
    $fragments_{hashes}[(id, c)] \leftarrow fragments_{hashes}[(id, c)] \cup [\pi_j]$
    if (not yet sent $\langle id, \mathsf{READY}, c\rangle$ and received $t + 1$ $\langle \mathsf{READY}\rangle$ messages with the same $c$) then
        wait for $t + 1$ $\langle \mathsf{ECHO}\rangle$ messages with the same $c$ and $\pi_i$
            send $\langle id, \mathsf{READY}, c, \pi_i\rangle$ to all nodes
    if $fragments_{hashes}[(id, c)] \geq 2t + 1$ then    //online error correcting code to reconstruct $D$
        $D' \leftarrow \mathsf{ECCDec}(t + 1, e, fragment_{hashes}[id, c])$
        if $H(D') = c$ then
            wait for t+1 $\langle \mathsf{ECHO}\rangle$ message where $H(d_j) \in D'$ and filter $fragments_{data}[(id, c)]$ accordingly
              $M \leftarrow \mathsf{ECDec}(\mathrm{fragments_{data}}[(\mathrm{id, c})])$    //reconstruct $M$ from the fragments that are contained in $D'$
              $\boldsymbol{d'} \leftarrow \mathsf{ECEnc}(M)$
              if $D' = H(d'_1), \ldots, H(d'_n)$ then
                  output $M$ and return
              else output $\perp$
        else $e \rightarrow e + 1$    //increase the number of errors to align with the online error correcting code procedure
```

We initialize CCBRB by creating two empty dictionaries called $fragments_{data}$ and $fragments_{hashes}$ for each node i. Here, $fragments_{data}$ maps each id tag and c to possible data fragments for some message M. Additionally, $fragments_{hashes}$ maps each id tag message of a message M and c to possible fragments of the list of hashes of all n fragment of that message M. Then, CCBRB proceeds as follows.

- SEND *phase*: The sender s encodes the messages M into n fragments using an erasure code $(t + 1, n)$. Each fragment is of size $\frac{|M|}{t+1}$. The dealer then hashes each fragment and create D, a list of n hashes each of size k. The dealer then sends each node j a SEND message containing the fragment d_j and the list of hashes D.

- ECHO *phase*: Upon receiving a SEND message, each node i verifies the fragment d_i by checking that $H(d_i)$ is equal to the i^{th} hash in the cross checksum D. If the check succeeds then: node i uses ECC to encode D into n fragments (each of size k) and stores them in the list $\boldsymbol{\pi}$. Then i sends an ECHO message containing the data fragment d_i, fragment π_j (the j^{th} fragment of $\boldsymbol{\pi}$), and $c = H(D)$ to every node j.

- READY *phase*: Each node stores the fragments it received in the ECHO messages. A node i broadcasts a READY message containing c and π_i in two cases:

(1) Node i receives $2t + 1$ ECHO messages with the same c and π_i.

(2) Node i receives $t + 1$ READY messages with the same c and has not sent a READY message. In this case, i waits for $t + 1$ ECHO with the same c and π_i and then sends a READY message.

Upon receiving $n - t$ READY messages with the same c, each node starts decoding. In particular, i first decodes the coded fragments of hashes using the ECCDecfunction and outputs D'. It then compares $H(D')$ with c, the value it receives from $2t + 1$ READY messages. If $H(D') = c$, i waits for at least $t + 1$ ECHO messages such that for each coded fragment d_j included in an ECHO message, $H(d_j) \in D'$. Then i decodes the fragments and outputs M. Finally, i further encodes M and calculates the list of hashes for the coded fragments. If the list of hashes are consistent with the hashes of the coded fragments, i outputs M. Otherwise, i outputs $\perp$.

Theorem 3. *CCBRB (Algorithm 4.3) is a secure BRB protocol.*

Proof. **Validity.** If a correct sender s runs *broadcast* for id *id* and a message M then all correct nodes will pass the check $H(d_j) = D_j$ and have a valid data fragment of M (d_j = ECEnc(M)[j]) because of the correctness property of the hash function and the encoding algorithm. Hence, every correct node i will echo to node j a valid data fragment d_i of M, its own cross checksum fragment π_j of D, $c = H(D)$, and *id*. Therefore, all correct nodes will receive at least $2t + 1$ ECHO messages with a data fragment and the same c, *id* and consistent cross checksum fragments. Thus, every correct node i will receive at least $t + 1$ valid data fragments of M whose hashes are contained in D. After that, every correct node i will send a READY message with c, *id* and π_i. Accordingly, every correct node will eventually receive collectively at least $2t + 1$ READY messages with the same c, *id* but $2t + 1$ distinct cross checksum fragments of D, and decodes the fragments to D. Even if some READY messages with invalid cross checksum fragments of D were received, the node can detect this by virtue of the correctness of the online error correcting code algorithm *OEC* (since the maximum number of faulty READY messages is t). Hence, every correct node will be able to reconstruct D and by consequence M. M can be decoded because every correct node has at least $t + 1$ data fragments whose hashes are contained in D and because of the collision resistance of the hash function.

Agreement and Totality. Agreement and totality follow immediately from Lemmas 1-3 below. □

Lemma 1. *If a correct node i outputs M with id associated with a cross checksum D and a hash c such that $c = H(D)$, then every correct node i will eventually receive at least $t + 1$ ECHO messages with the same id, c and π_i. Additionally, each of these $t + 1$ ECHO messages will contain a distinct data fragment d_j whose hash is in the cross checksum ($H(d_j) \in D$). Finally, π_i is a valid fragment of D; that is, π_i = ECCEnc(D)[i].*

Proof. If a correct node *outputs* M with *id*, then it must have received $2t + 1$ READY messages with the same *id* and c. Therefore, $t + 1$ of those READY messages must have been sent by correct nodes. Hence, there is at least one correct node i that received $2t + 1$ ECHO messages with the same *id*, c and π_i. Since t is the total number of faulty nodes then at least $t + 1$ nodes received a SEND message from the sender with *id* containing a fragment such that the hash is contained in the cross checksum. Therefore, every correct node j will eventually receive at least $t + 1$ ECHO messages with the same *id*, c and π_j each containing a data fragment whose hash is in the cross checksum. Finally, correct nodes who generate the hash c for their ECHO messages must have received D in their SEND message (unless the sender s has broken collision resistance of the hash function), and therefore they will generate π_i consistent with D. □

Lemma 2. *If a correct node i outputs M with id associated with a cross checksum D and a hash c such that $c = H(D)$, then every correct node will eventually be able to reconstruct D.*

Proof. As stated above, if node i *outputs* M then with id, then it must have received at least $t + 1$ READY messages with the same c and id from correct nodes (possibly including node i itself). Therefore, all other t correct nodes will receive at least $t + 1$ READY messages with the same c and id. By Lemma 1, every correct node j will eventually receive $t + 1$ ECHO messages with the same c and id and valid π_j. Thus, they will be able to send their own READY messages with valid π_j. As a result, at least $2t + 1$ correct nodes will send READY messages with the same c and id as well as coded fragments that are consistent with the encoding of D. By the correctness of the *OEC* algorithm, it follows that every correct node will eventually reconstruct D. □

Lemma 3. *If a correct node i outputs M with id associated with a cross checksum D and a hash c such that $c = H(D)$, then every correct node eventually outputs M associated with the same id and D.*

Proof. By Lemma 2, every correct node will receive the same cross checksum D. Hence, every correct node can determine the validity of D deterministically and reconstruct M accordingly. Then, either node i will detect that the cross checksum D is a *valid* cross checksum consistent with some message $M \neq \perp$, or will detect that D is an invalid cross checksum and will *r-deliver* $M = \perp$. Either way, all other correct nodes will also detect D to be valid or invalid in the same way, due to the collision resistance of the hash function and the correctness of the encode and decode algorithm and Lemma 1. □

Communication complexity. In the following analysis we will assume optimal resilience with $n = 3t + 1$. The protocol consists of three steps:

(1) SEND: p_s sends one SEND message to all n replicas. Each SEND message consists of the cross-checksum D and the fragment d_i each of size $n\kappa$ and $\frac{|M|}{t+1}$ respectively. Thus, the total communication complexity for the SEND is $3|M|+\kappa n^2$.
(2) ECHO: Every correct replica p_i sends one ECHO message to all n replicas. Each ECHO message consists of the cross checksum fragment π_j, the data fragment d_j and a hash c where the size of the terms costs $\frac{n\kappa}{t+1}$, $\frac{|M|}{t+1}$ and κ respectively. Thus the total communication complexity for the ECHO is $3n|M|+4\kappa n^2$.
(3) READY: Every correct replica p_i sends one READY message to all n replicas. Each READY message consists of the cross-checksum fragment π_i and the hash c respectively. Thus the total communication complexity is $4kn^2$.

Hence the total communication complexity is: $3n|M|+9n\kappa^2 + 3|M|$ or $O(n|M|+n\kappa^2)$

4.3 Balanced Cross-Checksum BRB

The Cross-Checksum BRB discussed in algorithm is not balanced. The dealer has to send the whole cross-checksum D to all nodes. However, we can easily make the protocol balanced at the expense of adding one extra step of communication. To do this, just like the BalMC algorithm introduced in algorithm 2, the dealer disperses the cross-checksum D instead of sending it in full as part of the $\langle id,\ \text{SEND}, d_j, D\rangle$ message. This would effectively translate to the dealer sending to node j the ECC fragment D_j instead of D along side each erasure fragment d_j. Following on with the BalMC algorithm, an extra step has to be added here, where every node j would send it's own ECC fragment D_j to every other node. Each node would then run the IT-OEC algorithm to reconstruct the vector D. Once the vector D is reconstructed, the protocol continues in the same way as depicted in Algorithm 4.3.

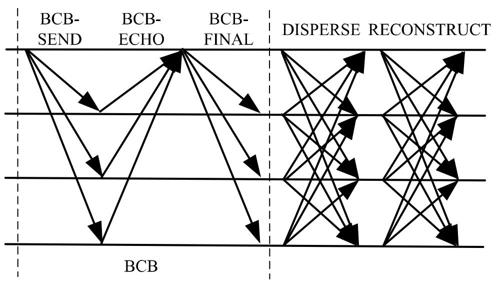

Fig. 1. A half-baked idea to reduce communication.

5 REDUCING COMMUNICATION USING THRESHOLD SIGNATURES

This section first describes SigBRB, a BRB protocol that uses threshold signatures to reduce communication. Then we present BalSigBRB that uses the technique in §3 and achieves balanced communication. Both protocols work in the trusted setup model, as threshold signatures require trusted setup. In applications where trusted setup is permitted (for instance, all known asynchronous BFT protocols implemented using BRB), one can directly use SigBRB. In applications where trusted setup is not allowed, one may run a distributed key generation (DKG) algorithm to generate the needed public parameters [2, 22, 36].

5.1 Overview of SigBRB

A half-baked idea: breaking the symmetry for BRB design. Our first idea is to break the symmetry in designing BRB protocols. Indeed, when designing efficient BRB protocols, one typically follows a symmetric design approach: in the first round, the broadcaster sends some data (either the whole input message M or a coded fragment) to every node; in the following rounds, nodes broadcast fragments and/or short cryptographic proofs to each other in order to achieve agreement.

In our new design, we break BRB constructions into a linear communication phase and a broadcast phase. We use cryptography (e.g., hashes) in the first linear communication phase, while the broadcast phase explicitly rules out using any cryptographic tools (e.g., hashes, signatures).

Our starting protocol works as follows. In the first linear communication phase, the goal is to disperse the input to ensure that $t+1$ correct nodes to have consistent data, a goal that consistent broadcast (BCB) [15, 46] or its information dispersal version may achieve; in the second broadcast phase, the idea is to "amplify" consistent data from $t+1$ correct nodes to all nodes, a goal that asynchronous data dissemination (ADD) [21] may achieve. Such a construction is depicted in Figure 1. We use BCB and ADD in a black-box manner.

For the above construction, the first phase has $O(n|M|+\kappa n)$ communication, while the second phase has $O(n|M|+n^2 \log n)$ communication. Adding them together, we have a construction with $O(n|M|+\kappa n + n^2 \log n)$. Unfortunately, the construction only achieves validity but not agreement. Indeed, it is easy to show that some correct nodes output message M, while some other nodes do not output any message, violating agreement. Note that in this case not all correct nodes start ADD.

As an example, a faulty sender may make only one correct node deliver the message in the BCB phase and enter the second phase. All t faulty nodes collude and disseminate correct fragments to $t+1$ correct nodes. Together with the fragment from the correct node, each of the $t+1$ correct nodes receive $t+1$ matching fragments, share their fragments,

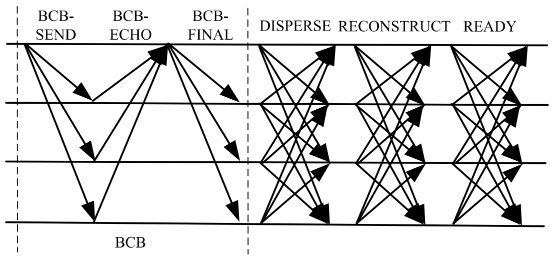

Fig. 2. SigBRB workflow.

complete ADD, and deliver the corresponding message. The rest t correct nodes, however, cannot deliver the message, since they fail to start ADD.

Below, we outline how we solve the agreement problem, by providing an approach to handling the issue *retroactively*.

As shown in Figure 2), our main idea is to let the agreement issue occur and then fix it retroactively. We add one more READY round after the ADD phase and ask nodes to output a message M only if it receives enough $\langle$READY$\rangle$ messages. The most interesting part is that an amplification round is now introduced, *going back to the very first round of the broadcast phase*, instead of the beginning of the same round. To our knowledge, our novel amplification technique is in contrast to all other known amplification rounds ever used in BRB and even fault-tolerant distributed computing. Strikingly, the READY round and the "unconventional" amplification round are all we need for a secure BRB construction.

5.2 The SigBRB Protocol

We show the workflow of SigBRB in Figure 2 and pseudocode in Algorithm 4. SigBRB consists of two phases: a linear BCB phase and a broadcast phase.

BCB phase (Algorithm 4: line 5-17). This phase runs a standard BCB. In particular, the broadcaster node s broadcasts a message $\langle id, \text{BCB-SEND}, M\rangle$. Upon receiving $\langle id, \text{BCB-SEND}, M\rangle$, each node generates a partial signature σ_i and sends a message $\langle id, \text{BCB-REP}, M, \sigma_i\rangle$ to node s. If s receives $n-t$ partial signatures, it combines them into a threshold signature σ and broadcasts an $\langle id, \text{BCB-FINAL}, M, \sigma\rangle$ message to all nodes. Upon receiving an $\langle id, \text{BCB-FINAL}, M, \sigma\rangle$ message, each node sets msg as M and $proof_1$ as σ and completes BCB.

Broadcast phase (Algorithm 4: line 18-39). The broadcast phase consists of three rounds: DISPERSE, RECONSTRUCT, and READY. Upon the completion of BCB, each node i encodes its msg into coded fragments $\boldsymbol{d}$. For each node j, node i sends it an $\langle id, \text{DISPERSE}, d_j\rangle$ message. Upon receiving $t+1$ matching $\langle id, \text{DISPERSE}, d_i^*\rangle$ messages, node i fixes d_i^* and then broadcasts a $\langle id, \text{RECONSTRUCT}, d_i^*\rangle$ message. Upon receiving at least $n-t$ $\langle$RECONSTRUCT$\rangle$ messages, each node starts to decode the message using OEC. This process may continue until OEC outputs a message M. A local parameter val is then set as M.

But this is not the last round of the SigBRB. When OEC outputs M, each node broadcasts an $\langle id, \text{READY}\rangle$ message. Furthermore, if node i previously has not sent a $\langle$DISPERSE$\rangle$ message, it disperses the coded fragments, i.e, i encodes M and sends node j (for any $j \in \{1, \cdots n\}$) an $\langle id, \text{DISPERSE}, d_j\rangle$ message. Each node waits for $n-t$ $\langle id, \text{READY}\rangle$ messages and then delivers message val (message output by OEC).

Discussion and communication complexity. The crucial round for SigBRB to achieve agreement is the amplification round after message M is output by OEC. In particular, if the OEC outputs M and a node has not previously sent a $\langle$DISPERSE$\rangle$ message, the node encodes M and sends the coded fragments via a $\langle$DISPERSE$\rangle$ message to the nodes. If a

Algorithm 4 SigBRB with identifier *id* and sender *s*. Code for node *i*, $i \in [n]$

```
1:  Initialization
2:    $(pk, sk) \leftarrow tgen(1^k)$ //threshold signature key generation; $pk$ is the public key and $sk$ is a vector of $n$ private keys
3:    $proof_1 \leftarrow \perp, msg \leftarrow \perp$                                  //initialize the parameters
4:    $val \leftarrow \perp, pset_1 \leftarrow \emptyset, T \leftarrow \emptyset$
5:   input $M$                                                           //BCB-SEND round
6:    $msg \leftarrow M$
7:    broadcast $\langle id, \texttt{BCB-SEND}, M\rangle$
8:  upon receiving $\langle id, \texttt{BCB-SEND}, M\rangle$ from $s$ do
9:    $msg \leftarrow M, \sigma_i \leftarrow tsign(id, M)$                              //BCB-REP round
10:   send $\langle id, \texttt{BCB-REP}, M, \sigma_i\rangle$ to $s$
11: upon receiving $\langle id, \texttt{BCB-REP}, M, \sigma_j\rangle$ from $j$ and $i = s$ do
12:   if $shareverify((id, M), \sigma_j)$ and $M = msg$ then
13:       add $\sigma_j$ to $pset_1$
14:   if $|pset_1| \geq n - t$ then                                        //BCB-FINAL round
15:       $\sigma \leftarrow tcombine((id, M), pset_1)$
16:       broadcast $\langle id, \texttt{BCB-FINAL}, M, \sigma\rangle$
17:  upon receiving $\langle id, \texttt{BCB-FINAL}, M, \sigma\rangle$ from $s$ do
18:    if $tverify((id, M), \sigma)$ and $M = msg$ then
19:       $proof_1 \leftarrow \sigma$                                                //DISPERSE round
20:       $\boldsymbol{d} \leftarrow \mathsf{ECCEnc}(t + 1, n, msg)$
21:       for $j \in \{1, \cdots n\}$
22:          send $\langle id, \texttt{DISPERSE}, d_j\rangle$ to $j$
23: upon receiving $t + 1$ matching $\langle id, \texttt{DISPERSE}, d_i^*\rangle$ do
24:   fix $d_i^*$
25:   broadcast $\langle id, \texttt{RECONSTRUCT}, d_i^*\rangle$                        //RECONSTRUCT round
26: upon receiving $\langle id, \texttt{RECONSTRUCT}, d_j\rangle$ from $j$ do
27:   add $d_j$ to $T$
28:   for $0 \leq r \leq t$ do
29:       wait until $|T| \geq 2t + r + 1$
30:       if $\mathsf{ECCDec}(t + 1, T, r) = M$ then
31:          $val \leftarrow M$
32:          broadcast $\langle id, \texttt{READY}\rangle$                                //send ⟨READY⟩
33:          if $\langle\texttt{DISPERSE}\rangle$ has not been sent
34:             $\boldsymbol{d} \leftarrow \mathsf{ECCEnc}(t + 1, n, M)$ then
35:             for $j \in \{1, \cdots n\}$                                       //amplification
36:                send $\langle id, \texttt{DISPERSE}, d_j\rangle$ to $j$
37: upon receiving $n - t$ $\langle id, \texttt{READY}\rangle$ do                            //READY round
38:   if $val \neq \perp$ then
39:        output $val$ and return
```

node outputs M, it has received $n - t$ $\langle id, \texttt{READY}\rangle$ messages and at least $t + 1$ correct nodes have completed the OEC. These correct nodes will send their coded fragments via the $\langle\texttt{DISPERSE}\rangle$ message. Accordingly, it is guaranteed that all correct nodes eventually decode M, broadcast the $\langle\texttt{READY}\rangle$ messages, and output M.

The consistency property of BCB guarantees that no correct nodes will broadcast inconsistent coded fragments, as each node only disperses the coded fragments upon the completion of BCB. An adversary cannot force any correct nodes to receive $t + 1$ matching but incorrect fragments in the DISPERSE round. Therefore, OEC can correct the errors and ensure that all correct nodes output the same message M.

Let us analyze the communication complexity of SigBRB. First, the first linear BCB phase has $O(n|M|+\kappa n)$ communication. The DISPERSE and RECONSTRUCT rounds both have $O(|M|n + n^2 \log n)$ communication. The READY phase does not carry bulk data and has $O(n^2)$ communication only. Therefore, the communication complexity for SigBRB is $O(n|M|+\kappa n + n^2 \log n)$.

It is also easy to replace threshold signatures using aggregate signatures, so the resulting protocol maintains the same complexity while working in the PKI model.

5.3 Analysis

Theorem 4. *Assuming a secure threshold signature and authenticated channels, SigBRB satisfies validity, agreement, and integrity.*

Proof. We first provide the following three lemmas.

Lemma 4. *If a correct node sends a $\langle$DISPERSE$\rangle$ message with fragments encoded from message M, at least one correct node has completed BCB and received M from the sender.*

Proof. Each correct node sends a $\langle$DISPERSE$\rangle$ message with fragments encoded from M for two cases: 1) It completes BCB and receives M from the sender; 2) It completes the RECONSTRUCT round and obtains M. We show that in both cases, at least one correct node has completed BCB and receives M from the sender. For the first case, trivial. For the second case, if a correct node completes the RECONSTRUCT round, it must have received at least $n - t$ RECONSTRUCT messages, among which at least $t + 1$ are sent by correct nodes. For any of the correct nodes, it must have also received $t + 1$ matching $\langle$DISPERSE$\rangle$ messages. As there are at most t faulty nodes, there must exist at least one correct node that sends the $\langle$DISPERSE$\rangle$ message after it completes BCB and receives M. □

Lemma 5. *If a correct node i sends an $\langle id, \text{RECONSTRUCT}, d_i^* \rangle$ message and a correct node j sends an $\langle id, \text{RECONSTRUCT}, d_j^* \rangle$ message, d_i^* and d_j^* are both encoded from the same message M.*

Proof. Let d_i^* be a coded fragment encoded from message M. If a correct node i sends an $\langle id, \text{RECONSTRUCT}, d_i^* \rangle$ message, it must have received $t + 1$ matching $\langle id, \text{DISPERSE}, d_i^* \rangle$ messages, among which at least one is sent by a correct node. Among them, at least one correct node has sent a $\langle$DISPERSE$\rangle$ message. According to Lemma 4, at least one correct node completes BCB and receives message M from the sender.

Let d_j^* be a coded fragment encoded from message $\widetilde{M}$. If j sends an $\langle id, \text{RECONSTRUCT}, d_j^* \rangle$ message, according to Lemma 4, at least one correct node completes BCB and receives message $\widetilde{M}$. This violates the consistency property of BCB. Thus, it holds $M = \widetilde{M}$. □

Lemma 6. *If a correct node outputs M, the sender s has previously broadcast M and at least one correct node has received a valid threshold signature $proof_1 = \sigma$.*

Proof. If a correct node i outputs M, it receives $n - t$ $\langle id, \text{READY} \rangle$ messages. Furthermore, in the RECONSTRUCT round, the OEC outputs a decoded message M. Accordingly, i must have received at least $2t + 1$ $\langle id, \text{RECONSTRUCT}, d_j \rangle$ messages, among which at least $t + 1$ are sent by correct nodes.

According to Lemma 5, the $t + 1$ correct nodes only send coded fragments encoded from the same message M'. Therefore, the reconstructed t-degree polynomial for M must agree with $t + 1$ fragments from correct nodes. It must hold that $M = M'$.

If i outputs M, it has received at least $2t + 1$ $\langle \texttt{READY} \rangle$ messages and also $2t + 1$ $\langle \texttt{RECONSTRUCT} \rangle$ messages, among which at least $t + 1$ are sent by correct nodes. According to Lemma 4, at least one correct node has completed BCB and received a valid threshold signature $proof_1$ for M, and meanwhile the sender s has sent M. □

In the following, we prove that SigBRB satisfies validity, agreement, and totality.

Validity. If a correct node s inputs M, all correct nodes complete BCB, according to the validity property of BCB. Therefore, all correct nodes will send the $\langle \texttt{DISPERSE} \rangle$ messages for the same M, receive $t + 1$ matching $\langle \texttt{DISPERSE} \rangle$ messages, and broadcast the $\langle \texttt{RECONSTRUCT} \rangle$ messages. No correct node can receive $t + 1$ matching $\langle id, \texttt{DISPERSE}, d_i \rangle$ for $M' \neq M$. Each correct node will then send a $\langle id, \texttt{RECONSTRUCT}, d_i \rangle$ messages such that d_i is encoded from M. Thus, all correct nodes will receive $2t + 1$ $\langle \texttt{RECONSTRUCT} \rangle$ messages from all correct nodes, output some value and then send the $\langle id, \texttt{READY} \rangle$ messages. All correct nodes, including the sender, will then receive $n - t$ $\langle id, \texttt{READY} \rangle$ messages and output some message. Furthermore, as shown in Lemma 6, if any correct node outputs $M' \neq M$, s has previously sent M', contradicting the fact that s is a correct node and inputs M. Therefore, all correct nodes will output the original message M.

Agreement. If a correct node i outputs M, according to Lemma 6, at least one correct node has completed BCB and possessed a valid threshold signature for M. Furthermore, If j outputs M', at least one correct node has completed BCB and possessed a valid threshold signature for M'. This violates the consistency property of BCB. Thus, $M = M'$.

Totality. If a correct node i outputs a message, it has received $n - t$ $\langle \texttt{READY} \rangle$ messages, among which at least $t + 1$ are sent by correct nodes. The $t + 1$ correct nodes must all output M by OEC, as proved in the first part. Furthermore, for each of the $t + 1$ correct nodes, if it has not previously sent a $\langle \texttt{DISPERSE} \rangle$ message, it will encode message M and broadcast the $\langle \texttt{DISPERSE} \rangle$ messages according to our protocol. Therefore, each correct node eventually receives at least $t + 1$ matching $\langle \texttt{DISPERSE} \rangle$ messages. Eventually, all correct nodes will receive at least $n - t$ $\langle \texttt{RECONSTRUCT} \rangle$ messages. According to Lemma 5, each correct node sends a fragment $\langle id, \texttt{RECONSTRUCT}, d_i \rangle$ such that d_i is encoded from the same message M. Therefore, each correct node eventually outputs some message. Finally, each correct node will eventually send an $\langle id, \texttt{READY} \rangle$ message, receive $n - t$ $\langle id, \texttt{READY} \rangle$ messages, and output the value. □

5.4 BalSigBRB

It is easy to transform the SigBRB protocol to BalSigBRB achieving balanced communication by using the technique used in §3. Doing so incurs an additional round of communication.

6 BALANCED ERROR-FREE BRB

In this section, we will introduce our BalEFBRB protocol, which is error-free, balanced and achieves near-optimal communication cost. Our protocol is heavily inspired by the recent error-free synchronous Byzantine agreement protocol named COOL [19]. We extend the COOL protocol [19] to obtain an error-free asynchronous BRB protocol with per-node communication cost $O(|M| + n \log n)$. We will first intuitively explain the modifications on top of COOL, and then describe our BalEFBRB protocol in more detail. We make the following three major changes to obtain our BalEFBRB protocol.

(1) *Triggering the next message upon receiving sufficient messages asynchronously, instead of receiving all messages from the previous synchronous round.* The COOL protocol is a synchronous protocol that proceeds in lock-step rounds, so it contains several steps where nodes wait for all messages from the previous round before taking their next step. For instance, a node will send an indicator message for 1 if it receives $n - t$ 1-indicators in the previous round; otherwise it sends an indicator for 0. However, under asynchrony, the node cannot expect to receive all messages, since a slow honest node is indistinguishable from a Byzantine node. Therefore, we need to change the triggering event to receiving enough messages asynchronously. For instance, the above example is changed to the following: a node sends 1-indicator upon receiving $n - t$ 1-indicators and sends 0-indicator upon receiving $t + 1$ 0-indicators. Since there are n nodes in total and each node can send one indicator, the above two conditions will not hold simultaneously. Moreover, if the original synchronous protocol relies on the fact that a node receives enough indicators, then the new asynchronous protocol preserves the same property since the node only triggers the message event after receiving enough indicators.

(2) *Replacing the 1-bit asynchronous BA with 1-bit Bracha's BRB.* The COOL protocol uses a synchronous binary BA protocol for all the nodes to agree on whether there are enough honest nodes holding the correct coded message fragments in order to recover the message. Our BalEFBRB also requires a similar step under asynchrony. However, we cannot use an asynchronous binary BA to construct an error-free BRB, because any asynchronous BA has to be randomized due to the FLP impossibility result [28]. Instead, we use the 1-bit Bracha's BRB [14], which is error-free, as follows. When a node inputs 1 (or 0) to the synchronous BA in the COOL protocol, we let the node send an `ECHO` message for 1 (or 0) in the 1-bit Bracha's BRB. As a result, the 1-bit Bracha's BRB guarantees agreement among the nodes on whether they should reconstruct the message or simply output a default message $\perp$. Moreover, as we will explain in Lemma 7, if one honest node outputs 1 in the 1-bit Bracha's BRB, then every honest node will be able to reconstruct and output the same message.

(3) *Balancing the broadcaster's cost by the technique of §3.* The BRB protocol obtained after the above two changes is still not balanced, since the straightforward transformation from agreement to broadcast asks the broadcaster to send the entire message in the first step of the protocol, leading to a cost of at least $\Omega(n|M|)$ at the broadcaster. Therefore, we apply the technique of §3, which ensures that broadcaster incurs a communication cost of $O(|M|+n \log n)$.

For brevity, we will only present the balanced version of our error-free BRB protocol BalEFBRB in the next section. The unbalanced protocol EFBRB can be easily obtained by broadcaster using multicast instead of the balanced multicast for sending its input message. The unbalanced protocol EFBRB has one less round, but the broadcaster incurs communication cost $O(n|M|+n \log n)$ instead of $O(|M|+n \log n)$, compared with the balanced one.

6.1 Design of BalEFBRB

Our error-free BRB has five phases: phase 0 to 4. We summarize our protocol in Algorithm 5 and describe each phase in detail.

Phase 0: The purpose of phase 0 is to let the broadcaster to send its proposal, M, to all nodes. As discussed in §3, if the broadcaster sends its proposal directly to each node, the broadcaster would incur a communication cost of $O(n|M|)$. We adopt the approach we design in §3. More specifically, during phase 0, the broadcaster encodes M using a $(n, t + 1)$ Reed-Solomon code. Let $[m_1, m_2, \dots, m_n]$ be the encoded fragments. The broadcaster then sends the i-th fragment m_i to node i as $\langle \texttt{PROPOSE}, m_i \rangle$ message. Each node i, upon receiving $\langle \texttt{PROPOSE}, m_i \rangle$ message from the broadcaster, sends

Algorithm 5 BalEFBRB protocol, code for node i, $i \in [n]$

```
PHASE 0:
1:  // only broadcaster node
2:  input $M$
3:  Let $[m_1, m_2, \ldots, m_n] := \mathsf{ECCEnc}(M, n, t+1)$
4:  send ⟨PROPOSE, $m_j$⟩ to node $j$ for each $j \in [n]$

    // each node i
5:  Let $M := \perp$
6:  Initialize $\mathcal{S}_0^1, \mathcal{S}_1^1, \mathcal{S}_0^2, \mathcal{S}_1^2, \mathcal{S}_0^3, \mathcal{S}_1^3, \mathcal{S}_0^4, \mathcal{S}_1^4$ to be $\emptyset$
7:  upon receiving the first ⟨PROPOSE, $m_i$⟩ from the broadcaster do
8:      send ⟨SHARE, $m_i$⟩ to all nodes
9:  For the first ⟨SHARE, $m_j^*$⟩ received from node $j$, add $(j, m_j^*)$ to $T$ // $T$ initialized as $\emptyset$
10: Perform IT-OEC for set $T$ (Algorithm 1)
11: Let $M_i$ be the returned value of IT-OEC

PHASE 1:
12: Let $[y_1^{(i)}, y_2^{(i)}, \ldots, y_n^{(i)}] := \mathsf{ECCEnc}(M_i, n, k)$
13: send ⟨SYMBOLS, $(y_j^{(i)}, y_i^{(i)})$⟩ to node $j$, $\forall j \in [n]$.   // Exchange fragments
14: upon receiving ⟨SYMBOLS, $(y_i^{(j)}, y_j^{(j)})$⟩ from node $j$ for the first time do
15:     if $(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})$ then
16:         Let $\mathcal{S}_1^1 := \mathcal{S}_1^1 \cup \{j\}$.
17:     else
18:         Let $\mathcal{S}_0^1 := \mathcal{S}_0^1 \cup \{j\}$.
19: upon $|\mathcal{S}_1^1| >= n - t$ do
20:     set $s_i^1 = 1$, send ⟨$P1, s_i^1$⟩ to all.
21: upon $|\mathcal{S}_0^1| >= t + 1$ do
22:     set $s_i^1 = 0$, send ⟨$P1, s_i^1$⟩ to all.
23: upon receiving ⟨$P1, s_j^1$⟩ from node $j$ for the first time do
24:     if $s_j^1 = 1$ then
25:         Wait till $j \in \mathcal{S}_0^1 \cup \mathcal{S}_1^1$
26:         if $j \in \mathcal{S}_1^1$ then
27:             Let $\mathcal{S}_1^2 := \mathcal{S}_1^2 \cup \{j\}$.
28:     else
29:         Let $\mathcal{S}_0^2 = \mathcal{S}_0^2 \cup \{j\}$.

PHASE 2:
30: if $s_i^1 = 1$ then
31:     upon $|\mathcal{S}_1^2| >= n - t$ do
32:         set $s_i^2 = 1$, send ⟨$P2, s_i^2$⟩ to all.
33:     upon $|\mathcal{S}_0^2| >= t + 1$ do
34:         set $s_i^2 = 0$, send ⟨$P2, s_i^2$⟩ to all.
35: else
36:     set $s_i^2 = 0$, send ⟨$P2, s_i^2$⟩ to all
37: upon receiving ⟨$P2, s_j^2$⟩ from node $j$ for the first time do
38:     if $s_j^2 = 1$ then
39:         Wait till $j \in \mathcal{S}_0^1 \cup \mathcal{S}_1^1$
40:         if $j \in \mathcal{S}_1^1$ then
41:             Let $\mathcal{S}_1^3 := \mathcal{S}_1^3 \cup \{j\}$.
42:     else
43:         Let $\mathcal{S}_0^3 := \mathcal{S}_0^3 \cup \{j\}$.

PHASE 3:
44: if $s_i^2 = 1$ then
45:     upon $|\mathcal{S}_1^3| >= n - t$ do
46:         set $s_i^3 = 1$, send ⟨$P3, s_i^3$⟩ to all.
47:     upon $|\mathcal{S}_0^3| >= t + 1$ do
48:         set $s_i^3 = 0$, send ⟨$P3, s_i^3$⟩ to all.
49: else
50:     set $s_i^3 = 0$, send ⟨$P3, s_i^3$⟩ to all.
51: upon receiving ⟨$P3, s_j^3$⟩ from node $j$ for the first time do
52:     if $s_j^3 = 1$ then
53:         Let $\mathcal{S}_1^4 := \mathcal{S}_1^4 \cup \{j\}$.
54:     else
55:         Let $\mathcal{S}_0^4 = \mathcal{S}_0^4 \cup \{j\}$.
56: upon $|\mathcal{S}_1^4| >= n - t$ do
57:     send ⟨ECHO, $s_i^4 = 1$⟩ to all.
58: upon $|\mathcal{S}_0^4| >= t + 1$ do
59:     send ⟨ECHO, $s_i^4 = 0$⟩ to all.
60: upon receiving $2t+1$ ⟨ECHO, $s$⟩ for matching $s$ and not having sent a READY message do
61:     send ⟨READY, $s$⟩ to all
62: upon receiving $t+1$ ⟨READY, $s$⟩ for matching $s$ and not having sent a READY message do
63:     send ⟨READY, $s$⟩ to all
64: upon receiving $2t+1$ ⟨READY, $s$⟩ for matching $s$ do
65:     if $s = 0$ then
66:         output $M = \perp$ and return
67:     start PHASE 4

PHASE 4:
68: // only after executing line 67
69: Wait till receiving $t+1$ ⟨SYMBOLS, $(y_i^{(j)}, *)$⟩, $\forall j \in \mathcal{S}_1^4$ // SYMBOLS messages from PHASE 1, and set $\mathcal{S}_1^4$ from PHASE 3
70: Let $m_i := y_i^{(j)}$
71: send ⟨RECONSTRUCT, $m_i$⟩ to all
72: For the first ⟨RECONSTRUCT, $m_j^*$⟩ received from node $j$, add $(j, m_j^*)$ to $T$ // $T$ initialized as $\emptyset$
73: Perform IT-OEC for set $T$ (Algorithm 1)
74: Let $M_i$ be the returned value of IT-OEC
75: output $M_i$ and return.
```

$\langle \texttt{SHARE}, m_i \rangle$ to all other nodes. Every node then performs information theoretic online error correction using the SHARE messages to recover the potential proposal. Let M_i be the proposal node i recovers at the end of phase 0.

<u>*Phase 1*</u>: During phase 1, each node first encode the proposal it recovered during phase 0 using a (n, k) Reed-Solomon code for $k = \lfloor \frac{t}{5} \rfloor + 1$. Let $[y_1^{(i)}, y_2^{(i)}, \ldots, y_n^{(i)}] := \texttt{ECCEnc}(M_i, n, k)$ be the output of the encoding procedure at node i (line 11). Node i then sends the $\langle \texttt{SYMBOLS}, y_i^{(i)}, y_j^{(i)} \rangle$ to node j for every $j \in [n]$ (line 12). Also, node i upon receiving $\langle \texttt{SYMBOLS}, y_i^{(j)}, y_j^{(j)} \rangle$ from node j adds node j to the set $\mathcal{S}_1^1$ if $(y_i^{(i)}, y_j^{(i)}) = (y_i^{(j)}, y_j^{(j)})$ (line 15-16). Otherwise, node i adds j to the set $\mathcal{S}_0^1$ (line 18). Node i then waits until either $|\mathcal{S}_1^1|$ is greater than or equal to $n - t$, or $|\mathcal{S}_0^1|$ is greater than or equal to $t + 1$. The event $|\mathcal{S}_1^1| \geq n - t$ implies that node i received matching fragments from at least $n - t$ nodes. Alternatively, the event $|\mathcal{S}_0^1| \geq t + 1$ implies that node i received non-matching fragment from at least $t + 1$ nodes.

Upon $|\mathcal{S}_1^1| \geq n - t$, node i sends the message $\langle P1, s_i^1 = 1 \rangle$ to all nodes (line 19-20). Alternatively, if $|\mathcal{S}_0^1| \geq t + 1$, node i sends the message $\langle P1, s_i^1 = 0 \rangle$ to all nodes (line 21-22). Finally, for every $\langle P1, s_j^1 = 1 \rangle$ message received from any node j, node i adds node j to the set $\mathcal{S}_1^2$ once it received j's fragments in phase 1 and the fragments match (line 24-27). Otherwise, node i adds the senders of $\langle P1, 0 \rangle$ message to the set $\mathcal{S}_0^2$ (line 29). The reason for waiting to receive node j's fragments in line 25 is that we do not want to add nodes with mismatched fragments in $\mathcal{S}_1^2$, which is crucial for Lemma 7 to hold. Note that it is still possible that an honest node j whose fragments do not match node i, i.e., $j \in \mathcal{S}_0^1$ at node i, sends $s_j^2 = 1$. In our protocol node i ignores all such messages (line 24-27).

<u>*Phase 2*</u>: During phase 2, if s_i^1 as calculated in phase 1 (line 19-22) is equal to 0, node i sends the $\langle P2, s_i^2 = 0 \rangle$ to every node. Otherwise, if s_i^1 is 1, then depending upon the size of $\mathcal{S}_1^2$ or $\mathcal{S}_0^2$, node i sends the following message. Upon $|\mathcal{S}_1^2| \geq n - t$, node i sends $\langle P2, s_i^2 = 1 \rangle$ to all nodes (line 31-32). Otherwise, upon $|\mathcal{S}_0^2| \geq t + 1$, node i sends $\langle P2, s_i^2 = 0 \rangle$ to every node (line 33-34). Also, similar to phase 1, for every $\langle P2, s_j^2 = 1 \rangle$ message received from any node j, node i adds node j to the set $\mathcal{S}_1^3$ once it received j's fragments in phase 1 and the fragments match (Line 38-41). Otherwise, node i adds the senders of $\langle P2, 0 \rangle$ message to the set $\mathcal{S}_0^3$ (line 43).

<u>*Phase 3*</u>: The first part of phase 3 (line 44-55) is similar to phase 2 (line 30-43), except that any node j that sends $\langle P3, s_i^3 = 1 \rangle$ is included in set $\mathcal{S}_1^4$ without the additional checks as in phase 1 and 2.

The remaining steps of phase 3 is analogous to running the 1-bit BRB protocol due to Bracha [14]. Specifically, upon $|\mathcal{S}_1^4| \geq n - t$, node i sends $\langle \texttt{ECHO}, s_i^4 = 1 \rangle$ to all nodes (line 56-57). Otherwise, if $|\mathcal{S}_0^4| \geq t + 1$, node i sends $\langle \texttt{ECHO}, s_i^4 = 0 \rangle$ to every node (line 58-59). Intuitively, the content of the ECHO message (1 or 0) from node i denotes the opinion of node i on whether every node should output $M' \neq \perp$ or $M' = \perp$. Each node upon receiving $2t + 1$ $\langle \texttt{ECHO}, s \rangle$ messages for a matching s, sends the $\langle \texttt{READY}, s \rangle$ message, if it have not sent it already (line 60-61). A node also sends the $\langle \texttt{READY}, s \rangle$ message upon receiving $t + 1$ matching $\langle \texttt{READY}, s \rangle$ messages, if have not sent it already (line 62-63). Finally, upon receiving $2t + 1$ matching $\langle \texttt{READY}, s \rangle$, if $s = 0$, node i outputs $\perp$ and returns (line 65-66). Otherwise, node i proceeds to phase 4 (line 67).

<u>*Phase 4*</u>: A node starts phase 4 only after receiving $2t + 1$ $\langle \texttt{READY}, 1 \rangle$ messages. During phase 4, each node waits for $t + 1$ matching $\langle \texttt{SYMBOLS}, y_i^{(j)}, * \rangle$ from nodes in $\mathcal{S}_1^4$ (line 69). Recall that $\langle \texttt{SYMBOLS}, y_i^{(j)}, * \rangle$ are sent during phase 1. Let $m_i := y_i^(j)$ be the fragment received in $t + 1$ SYMBOLS messages with matching $y_i^(j)$ (line 70). Then, each node sends the message $\langle \texttt{RECONSTRUCT}, m_i \rangle$ to all nodes (line 71). Finally, each node uses the received RECONSTRUCT messages to perform OEC and outputs the output of the OEC algorithm (line 72-75).

6.2 Analysis

In this section we will analyze algorithm 5 and show that it implements an error-free BRB protocol for large messages with communication cost of $O(n|M|+n^2 \log n)$ and tolerates up to 1/3-rd Byzantine nodes. Our proof directly uses several Lemmas from [18] and we only provide the lemma statement for those lemmas. Some of the lemmas and their proofs are deferred to Appendix A for brevity.

Lemma 7 (Key Lemma). *When any honest node i sends* $\langle \mathsf{ECHO}, 1\rangle$ *at phase 3, all honest nodes in node i's set* $\mathcal{S}_1^4$ *recovers the same message at the end of phase 0.*

Proof. The proof of this lemma follows directly from the proof of [18, Lemma 3] where we use our proof of Lemma 10 and 11. □

Theorem 5 (Totality and Agreement). *Algorithm 5 guarantees the Totality and Agreement property.*

Proof. Suppose an honest node outputs M', then it has received $2t + 1$ $\langle \mathsf{READY}, s\rangle$ messages for matching s. Then since at least $t + 1$ messages above are from honest nodes, every honest node will eventually receive $t + 1$ $\langle \mathsf{READY}, s\rangle$ messages. Note that no honest node can send READY for any $s' \neq s$, due to the quorum intersection of ECHO messages. Thus all honest nodes will send $\langle \mathsf{READY}, s\rangle$ and thus receive $2t + 1$ $\langle \mathsf{READY}, s\rangle$. If $s = 0$, this implies that every honest will output the default message $M' = \perp$ and return. Otherwise, if $s = 1$, each node will start phase 4. What remains to show is that, during phase 4, each honest node will send a reconstruct message with a correct fragment of the encoding of a unique message M' (line 71) after receiving $t + 1$ matching SYMBOLS (line 69), and decode and output M' (line 72-75).

An honest node receiving $2t + 1$ $\langle \mathsf{READY}, 1\rangle$ implies that at least one honest node sent a $\langle \mathsf{ECHO}, 1\rangle$ message. Without loss of generality, let i be the first node that sent an $\langle \mathsf{ECHO}, 1\rangle$ message. Observe that node i sends $\langle \mathsf{ECHO}, 1\rangle$ only when $|\mathcal{S}_1^4| \geq n - t$ at node i. This means at least $n - 2t \geq t + 1$ nodes in $|\mathcal{S}_1^4|$ are honest and each such node j sent $s_j^3 = 1$ to all nodes. Also, due to Lemma 7, for every pair of honest nodes, they have the same initial message at the end of phase 0, i.e., $j, \ell \in \mathcal{S}_1^4$, $M_j = M_\ell$. Hence, every honest node i will eventually will receive at least $t+1$ matching $\langle \mathsf{SYMBOLS}, (y_i^{(j)}, *)\rangle$ messages from honest nodes in $\mathcal{S}_1^4$. This implies, every honest node will send RECONSTRUCT message with correct fragment, and due to properties of OEC, each honest will output the same message M'. □

Theorem 6 (Validity). *Algorithm 5 guarantees the Validity property.*

Proof. When the broadcaster is honest and has input M, due to guarantees of OEC, during phase 0, every honest node will eventually receive M, i.e., $M_i = M_j = M$ for all honest nodes i and j. Since, ECCEnc is a deterministic function, for every pair of honest nodes i and j, the tuple of fragments will match, i.e., $(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})$. Since there are at least $n - t$ honest nodes, eventually $|\mathcal{S}_1^1|$ will be greater than or equal to $n - t$ at all honest nodes and every honest node i will send $\langle P1, s_i^2 = 1\rangle$ to others. No honest node will send $\langle P1, s_i^2 = 0\rangle$ since there are at most t Byzantine nodes who may send inconsistent fragments. A similar argument also implies that during phase 2 and 3, each honest node i will send $\langle P2, s_i^3 = 1\rangle$ and $\langle P3, s_i^3 = 1\rangle$, respectively, to all other nodes. Hence, every honest node will eventually send $\langle \mathsf{ECHO}, 1\rangle$ and $\langle \mathsf{READY}, 1\rangle$ to others and all honest node will start to phase 4. Finally, during phase 4, every honest node sends a valid coded fragment of M in RECONSTRUCT message. Thus, again due to guarantees of OEC, every honest node will reconstruct the message M. □

Theorem 7 (Performance). *For any message M of size* $|M|$, *the total communication cost of Algorithm 5 is* $O(n|M|+n^2 \log n)$ *bits. Furthermore each node incurs a communication cost of* $O(|M|+n \log n)$.

Proof. During phase 0, the broadcaster sends a fragment to each node, and each node gossips the fragment to every other node. Note that in Reed-Solomon code, each fragment is of size $\max\{|M|/n, \log n\}$ bits. Hence, the communication cost of each node during phase 0 is at most $O(|M|+n\log n)$. Hence, the total communication during phase 0 is at most $O(n|M|+n^2\log n)$. During phase 1, each node sends two fragments (line 12) and a single bit (line 19 and 22) to every other node. Hence, by the same argument as above, the total communication cost of phase 1 is at most $O(n|M|+n^2\log n)$. A node only sends a single bit to other nodes during phase 2. Similarly, during phase 3, each node only sends 1-bit to others and runs a 1-bit Bracha's BRB protocol. Hence, the communication cost of phase 2 and phase 3 is $O(n^2)$. Finally, during phase 4, each node sends a fragment to all other nodes, hence the per node and the total communication cost of phase 4 is $O(|M|+n\log n)$ and $O(n|M|+n^2\log n)$, respectively.

Combining the above, the per node and the total communication cost of Algorithm 5 is $O(|M|+n\log n)$ and $O(n|M|+n^2\log n)$, respectively. □

7 LOWER BOUNDS

In this section, we prove a communication complexity lower bound for deterministic protocols that solve BRB, which have been mentioned in Table 1. To strengthen the result, the lower bound is proven under synchrony. The lower bound proof is inspired by [23].

For any deterministic BRB protocol with input M that tolerates up to $\Theta(n)$ Byzantine nodes, it is straightforward to show a lower bound of $\Omega(n|M|+n^2)$ [41] on the communication cost even under synchrony. The $\Omega(n|M|)$ part is because $O(n)$ honest nodes need to receive the message when the protocol terminates, and the $\Omega(n^2)$ part is due to the classic Dolev-Reischuk lower bound [23]. Therefore, all our protocols SigBRB, BalSigBRB, EFBRB and BalEFBRB have near-optimal communication cost.

Next, for any deterministic protocol that solves BRB under synchrony, we will show that $\Omega(|M|+n)$ is a lower bound on the communication cost of any protocol node including the broadcaster, which implies our BalSigBRB and BalEFBRB has near-optimal per-node cost as well.

Theorem 8. *In any deterministic protocol that solves BRB, for any honest node p, there exists an execution in which p incurs a communication cost of $\Omega(|M|+n)$.*

Proof. We will prove that for any deterministic BRB protocol, all honest nodes incur at least $\Omega(|M|+n)$ communication cost in at least one execution.

The argument for broadcaster is straightforward. First, the broadcaster needs to send at least $\Omega(|M|)$ bits for its input message M. Moreover, the broadcaster has to send messages to at least $t+1$ nodes, otherwise it is possible that no honest node receives any information from the broadcaster, and the Validity property of BRB can be violated. Since $t = \Theta(n)$, we conclude that the broadcaster has to send $\Omega(|M|+n)$ bits.

Consider any non-broadcaster honest node during any failure-free execution where the broadcaster has input M. This honest node needs to output M due to the Validity requirement, so at least $\Omega(|M|)$ bits need to be received.

Let $C_{p,E}$ denote the number of messages an honest node p sends to any node and receives from any honest node during an execution E. We show that $C_{p,E} \geq t/2+1$ for any honest node p in at least one execution E. Otherwise, suppose there exists a BRB protocol where an honest node q has $C_{q,E} \leq t/2$ for any execution E. If q receives no message during the entire execution but other honest nodes output for BRB, due to Totality q eventually outputs as well. Without loss of generality, suppose q outputs 0 in this case. Consider a failure-free execution $E1$ where the honest broadcaster has input 1. By assumption, $C_{q,E1} \leq t/2$. Let S denote the set of nodes that q receives messages from in $E1$. We have

$|S| \leq t/2$. Consider execution $E2$ where the honest broadcaster has input 1, and q is Byzantine and remains silent. Since the broadcaster is honest and has input 1, by Validity, all honest nodes output 1 in $E2$. Then, we construct another execution $E3$ same as $E2$ except that the nodes in S are Byzantine and q is now honest. The nodes in S behave identically as in $E2$, except that they send no message to q. By assumption, $C_{q,E3} \leq t/2$. The adversary also corrupts the set of nodes R that q sends messages to in $E3$. This is within the adversary's corruption budget since $|S \cup R| \leq |S| + |R| \leq t$. The Byzantine nodes in R behave identically as in $E2$. Since q receives no message in $E3$, q will output 0 in $E3$ by assumption. Other honest nodes will output 1 in $E3$ since they cannot distinguish $E2, E3$. However, the Agreement property of BRB is then violated. Therefore, we prove that $C_{p,E} \geq t/2 + 1$ for any honest node p in at least one execution E, which implies the communication cost at any honest node for any BRB protocol is $\Omega(n)$.

Therefore, in any deterministic protocol that solves BRB, for any honest node p, there exists an execution in which p incurs a communication cost of $\Omega(\max\{|M|, n\}) = \Omega(|M| + n)$. □

8 RELATED WORK

The problem of reliable broadcast (BRB) was introduced by Bracha [14]. In the same paper, Bracha provided an error-free BRB protocol for a single bit with a communication cost of $O(n^2)$, thus $O(n^2|M|)$ for $|M|$ bits using a naïve approach. Almost two decades later, Cachin and Tessaro [16] improved the cost to $O(n|M| + \kappa n^2 \log n)$ assuming a collision-resistant hash function with κ being the output size of the hash. Hendricks et al. in [34] propose an alternate BRB protocol with a communication cost of $O(n|M| + \kappa n^3)$ using a erasure coding scheme where each element of a codeword can be verified for correctness. Assuming a trusted setup phase, hardness of q-SDH [9, 10] and Decisional Bilinear Diffie-Hellman (DBDH) [12], Nayak et al. [41] reduced the communication cost to $O(n|M| + \kappa n^2)$.

Recently, Das et al. [21] presents a BRB protocol that has a communication cost to $O(n|M| + \kappa n^2)$ assuming only collision-resistant hash function. However, their BRB protocol has two limitations: 1) For sufficiently large messages, the protocol suffers from computational inefficiencies. 2) The broadcaster incurs a higher communication cost than the non-broadcaster nodes. We address both of these concerns while still maintaining the same total communication cost by introducing our two protocols CCBRB and BalCCBRB in §4. CCBRB is not balanced but offers better computational cost for large messages, while BalCCBRB is balanced, but at the cost of one extra step of communication.

The original BRB protocol due to Bracha [14] is error-free, i.e., it does not require any cryptographic assumptions and is secure against any computationally unbounded adversary in all executions. The error-free BRB protocol due to Patra [42] achieves a total communication cost of $O(n|M| + n^4 \log n)$, and it is later improved to $O(n|M| + n^3 \log n)$ by Nayak et al. [41]. The two protocols above are not balanced; the broadcaster has a cost roughly $O(n)$ higher than other nodes.

Our error-free BRB builds upon the recent result on synchronous error-free Byzantine agreement due to Chen [19]. In particular, our observation is that, with appropriate changes, we can use Chen's protocol to establish the initial condition of the Asynchronous Data Dissemination (ADD) problem introduced by [21]. ADD is a protocol that efficiently disseminates the message from a subset of honest node to all honest nodes in an asynchronous network. Note that, Chen's approach do not rely on any cryptographic assumption and incurs a communication cost of $O(n|M| + n^2 \log n)$. Thus, by combining the modified Chen's protocol along with our balancing technique from §3 and information theoretic Asynchronous Data Dissemination protocol of [21], we get an information-theoretically secure BRB protocol with near-optimal communication cost of $O(n|M| + n^2 \log n)$.

For SigBRB, we use Byzantine consistent broadcast (CBC) which may be viewed as BRB without the totality property. The notion has been implicitly discussed in [14, 48] and more formally by Reiter [46] and Cachin, Kursawe, Petzold, and Shoup (CKPS) [15]. The CBC construction is due to Cachin, Kursawe, Petzold, and Shoup [15].

BRB has also been explored in some extended settings, e.g., probabilistic BRB [32], BRB with dynamic membership [31].

Concurrent work. A concurrent and independent work of Abraham and Asharov [1] obtains asynchronous Byzantine reliable broadcast with near-optimal communication cost of $O(n|M|+n^2\log(n^3/\epsilon))$. Their protocol tolerates optimal resilience $t < n/3$ and is statistically secure with probability $1-\epsilon$. The computational cost of their protocol is $\tilde{O}(n|M|)$ since they need online error correction on the input message. Their protocol has 7 round if made balanced using our technique of §3, or 6 round without the balancing. Abraham and Asharov [1] also obtain similar results for synchronous gradecast.

9 BRB VS. AVID

Asynchronous verifiable information dispersal (AVID), introduced by Cachin and Tessaro [16], is a primitive closely related to BRB. The difference between these two primitives is that BRB requires that replicas store a full copy of message, but AVID may only ask replicas to store erasure-coded fragments in a consistent manner. It is straightforward to obtain a BRB protocol from an AVID, by all nodes invoking retrieval after the dispersal phase of AVID terminates.

The new techniques introduced in this paper apply to the problem of AVID as well. More specifically, the balanced multicast from §3 and cross-checksum from §4 inspired us to design a new AVID protocol with near-optimal communication cost. More details can be found in our PODC 2022 brief announcement [4].

10 CONCLUSION AND OPEN PROBLEMS

This paper investigates asynchronous Byzantine reliable broadcast, and make improvements over existing results in terms of computation cost, and communication cost in total and per-node. An intriguing open problem is that, can we simultaneously achieve improved computation cost and near-optimal communication cost? All our solutions with near-optimal communication have expensive online error correction on the input message, and our computation efficient protocol still has an $O(\kappa)$ gap from the communication complexity lower bound. Another interesting open problem is that, can we achieve optimal communication cost for Byzantine reliable broadcast? Our near-optimal solutions still have an $O(\log n)$ gap from the lower bound.

ACKNOWLEDGMENTS

The authors would like to thank our shepherd Elad Schiller and PODC reviewers for their helpful comments on the paper.

Nicolas Alhaddad and Mayank Varia are supported by NSF Grants No. 1718135, 1801564, 1915763, and 1931714, by the DARPA SIEVE program under Agreement No. HR00112020021, and by DARPA and the Naval Information Warfare Center (NIWC) under Contract No. N66001-15-C-4071. Sisi Duan was supported in part by Tsinghua Independent Research Program, National Financial Cryptography Research Center, and National Key Research and Development Program of China under grant No. 2018YFA0704701. Ling Ren is supported by NSF Grant No. 2143058. Haibin Zhang was supported in part by Shandong Provincial Key Research and Development Program (2021CXGC010106) and Teli Youth Scholarship.

REFERENCES

[1] Ittai Abraham and Gilad Asharov. Gradecast in synchrony and reliable broadcast in asynchrony with optimal resilience, efficiency, and unconditional security. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, 2022.

[2] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, page 363–373, 2021.

[3] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC)*, page 331–341, 2021.

[4] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Brief announcement: Asynchronous verifiable information dispersal with near-optimal communication. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, 2022.

[5] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Succinct erasure coding proof systems. *Cryptology ePrint Archive*, 2021.

[6] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Practical and improved byzantine reliable broadcast and asynchronous verifiable information dispersal from hash functions. Cryptology ePrint Archive, Paper 2022/171, 2022.

[7] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 52–61, 1993.

[8] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography — PKC 2003*, 2002.

[9] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *International conference on the theory and applications of cryptographic techniques*, pages 56–73. Springer, 2004.

[10] 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.

[11] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International conference on the theory and applications of cryptographic techniques*, pages 416–432. Springer, 2003.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of cryptology*, 17(4):297–319, 2004.

[14] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[15] 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.

[16] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201. IEEE, 2005.

[17] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 42–51, 1993.

[18] Jinyuan Chen. Fundamental limits of byzantine agreement. *arXiv preprint arXiv:2009.10965*, 2020.

[19] Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[20] Sourav Das, Ling Ren, and Zhuolun Xiang. Near-optimal balanced reliable broadcast and asynchronous verifiable information dispersal. Cryptology ePrint Archive, Paper 2022/052, 2022.

[21] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721, 2021.

[22] Sourav Das, Tom Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1549–1549, 2022.

[23] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

[24] Sisi Duan, Michael K Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2028–2041, 2018.

[25] Sisi Duan and Haibin Zhang. Byzantine reliable broadcast with $o(nl + kn + n^2 logn)$ communication. Cryptology ePrint Archive, Paper 2022/554, 2022.

[26] Sisi Duan and Haibin Zhang. Pace: Fully parallelizable bft from reproposable byzantine agreement. *ACM CCS*, 2022.

[27] Sisi Duan, Haibin Zhang, and Boxin Zhao. Waterbear: Information-theoretic asynchronous bft made practical. *Cryptology ePrint Archive*, 2022.

[28] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[29] Shuhong Gao. A new algorithm for decoding reed-solomon codes. In *Communications, information and network security*, pages 55–68. Springer, 2003.

[30] Xueqing Gong and Chi Wan Sung. Zigzag decodable codes: Linear-time erasure codes with applications to data storage. *Journal of Computer and System Sciences*, 89:190–208, 2017.

[31] Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Dynamic byzantine reliable broadcast. In *OPODIS 2020*, 2021.

[32] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. Scalable byzantine reliable broadcast. In *DISC 2019*, 2019.

[33] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.

[34] James Hendricks, Gregory R Ganger, and Michael K Reiter. Verifying distributed erasure-coded data. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 139–146, 2007.

[35] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is DAG. In *PODC*, pages 165–175. ACM, 2021.

[36] Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.

[37] Chao Liu, Sisi Duan, and Haibin Zhang. Mib: Asynchronous bft with more replicas. *arXiv preprint arXiv:2108.04488*, 2021.

[38] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. Honeybadgermpc and asynchromix: Practical asynchronous mpc and its application to anonymous communication. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 887–903, 2019.

[39] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[40] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[41] 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.

[42] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *International Conference On Principles Of Distributed Systems*, pages 34–49. Springer, 2011.

[43] James S. Plank, Jianqiang Luo, Catherine D. Schuman, Lihao Xu, and Zooko Wilcox-O'Hearn. A performance evaluation and examination of open-source erasure coding libraries for storage. In *FAST*, pages 253–265. USENIX, 2009.

[44] James S Plank and Lihao Xu. Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications. In *NCA*, pages 173–180. IEEE, 2006.

[45] 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.

[46] Michael K Reiter. Secure agreement protocols: Reliable and atomic group multicast in rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, 1994.

[47] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology — EUROCRYPT 2000*, 2000.

[48] Sam Toueg. Randomized byzantine agreements. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 163–178, 1984.

[49] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[50] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbacss: How to robustly share many secrets. In *(To appear) Proceedings of the 29th Annual Network and Distributed System Security Symposium*, 2022.

A RE-PROVING LEMMAS

In this section we will re-prove some lemmas from [18] for our Key Lemma (Lemma 7). Our proofs basically follow the original proof of [18], adapted to our asynchronous BRB protocol.

We will first restate the definitions introduced by Chen [18] in our protocol language.

Notation for groups of nodes. We divide the n-node network into group of nodes. The group definition is based on the values of the messages recovered by nodes at the end of phase 0 and values of the success indicators $\{s_i^j\}_{i=1}^n$ for $j = 1, 2, 3, 4$. Let $\mathcal{F}$ be the group consisting of the indices of all of the dishonest nodes. Note that $|\mathcal{F}| \leq t$. We define the following groups of honest nodes.

$$\mathcal{A}_l \triangleq \{i : M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta] \tag{1}$$

$$\mathcal{A}_l^{[1]} \triangleq \{i : s_i^1 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[1]}] \tag{2}$$

$$\mathcal{A}_l^{[2]} \triangleq \{i : s_i^2 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[2]}] \tag{3}$$

$$\mathcal{A}_l^{[3]} \triangleq \{i : s_i^3 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[3]}] \tag{4}$$

for some different non-empty values $\bar{M}_1, \bar{M}_2, \cdots, \bar{M}_\eta$ and some non-negative integers $\eta, \eta^{[1]}, \eta^{[2]}, \eta^{[3]}$ such that $\eta^{[3]} \leq \eta^{[2]} \leq \eta^{[1]} \leq \eta$. The above definition implies that Group $\mathcal{A}_l$ is a subset of honest nodes who recovered the same message at the end of phase 0. $\mathcal{A}_l^{[1]}$ is a subset of $\mathcal{A}_l$ who have the same non-empty value of updated messages at the end of phase 1. Note that at the end of phase 1, if the updated message of honest node i is non-empty, then it implies that its updated message remains the same as its message after phase 0. Moreover, $s_i^1 = 1$. Similarly, $\mathcal{A}_l^{[2]}$ is a subset of $\mathcal{A}_l^{[1]}$ who have the same non-empty value of updated messages at the end of Phase 2 for $l \in [\eta^{[2]}]$, while $\mathcal{A}_l^{[3]}$ is a subset of $\mathcal{A}_l^{[2]}$ who have the same non-empty value of updated messages at the end of Phase 3 for $l \in [\eta^{[3]}]$. In our setting, when $1 \leq \eta^{[3]} \leq \eta^{[2]} \leq \eta^{[1]} \leq \eta$, the sets $\mathcal{A}_l, \mathcal{A}_{l_1}^{[1]}, \mathcal{A}_{l_2}^{[2]}, \mathcal{A}_{l_3}^{[3]}$ are all non-empty for any $l \in [\eta], l_1 \in [\eta^{[1]}], l_2 \in [\eta^{[2]}], l_3 \in [\eta^{[3]}]$. Note that $\sum_{l=1}^{\eta} |\mathcal{A}_l| + |\mathcal{F}| = n$.

Let $\mathcal{B}^{[p]}$ defined as

$$\mathcal{B}^{[p]} \triangleq \{i : s_i^p = 0,\ i \notin \mathcal{F},\ i \in [n]\}, \quad p \in \{1, 2, 3\}. \tag{5}$$

Based on our definitions, it holds true that

$$\sum_{l=1}^{\eta^{[p]}} |\mathcal{A}_l^{[p]}| + |\mathcal{B}^{[p]}| + |\mathcal{F}| = n, \quad p \in \{1, 2, 3\}. \tag{6}$$

Throughout our analysis, for any message M, we use $M(\cdot) = \mathsf{ECCEnc}(M, n, k)$ to denote the Reed-Solomon encoding of the message M. Moreover, we use $M(i)$ to denote the i-th fragment of $M(\cdot)$. For some $i \in \mathcal{A}_l$, the equality of $\bar{M}_l(i) = \bar{M}_j(i)$ might be satisfied for some j and l. Thus, we further sub-divide the group $\mathcal{A}_l$ the following (possibly overlapping) sub-groups

$$\mathcal{A}_{l,j} \triangleq \{i :\ i \in \mathcal{A}_l, \bar{M}_l(i) = \bar{M}_j(i)\}, \quad j \neq l, j, l \in [\eta] \tag{7}$$

$$\mathcal{A}_{l,l} \triangleq \mathcal{A}_l \setminus \{\cup_{j=1, j \neq l}^{\eta} \mathcal{A}_{l,j}\}, \quad l \in [\eta]. \tag{8}$$

Similarly, Group $\mathcal{A}_l^{[p]}$ can be further divided into some sub-groups defined as

$$\mathcal{A}_{l,j}^{[p]} \triangleq \{i : i \in \mathcal{A}_l^{[p]}, \bar{M}_l(i) = \bar{M}_j(i)\}, \quad j \neq l, j, l \in [\eta^{[p]}] \tag{9}$$

$$\mathcal{A}_{l,l}^{[p]} \triangleq \mathcal{A}_l^{[p]} \setminus \{\cup_{j=1, j\neq l}^{\eta^{[p]}} \mathcal{A}_{l,j}^{[p]}\}, \quad l \in [\eta^{[p]}]. \tag{10}$$

for $p \in \{1, 2, 3\}$.

Notation for graphs. Chen [18] defines a graph $G = (\mathcal{P}, \mathcal{E})$, where $\mathcal{P}$ consists of $n - t$ vertices, i.e., $\mathcal{P} = [n - t]$, and $\mathcal{E}$ is the set of edges. Let $i^* \in \mathcal{P}$, and let $C \subseteq \mathcal{P} \setminus \{i^*\}$ be a of vertices with $|C| \geq n - 2t - 1$, such that each vertex in C is connected with at least $n - 2t$ edges and one of the edges is connected to vertex i^*. We count an edge connecting to itself as an edge as well. For any pair of vertices $i, j \in \mathcal{P}$, we use $E_{i,j} = 1$ (resp. $E_{i,j} = 0$) to indicate that there is an edge (resp. no edge) between vertex i and vertex j. In summary, in G, for a given $i^* \in \mathcal{P} = [n - t]$, the following properties regarding the set C holds.

$$E_{i,i^*} = 1 \quad \forall i \in C \tag{11}$$

$$\sum_{j \in \mathcal{P}} E_{i,j} \geq n - 2t \quad \forall i \in C \tag{12}$$

$$|C| \geq n - 2t - 1 \tag{13}$$

For the graph G, let $\mathcal{D} \subseteq \mathcal{P}$ denote the set of vertices such that each vertex in $\mathcal{D}$ is connected with at least k vertices in C, that is,

$$\mathcal{D} \triangleq \left\{ i : \sum_{j \in C} E_{i,j} \geq k, \ i \in \mathcal{P} \setminus \{i^*\} \right\} \tag{14}$$

where k is the Reed-Solomon encoding parameter. Then, the following lemma provides a result on bounding the size of $\mathcal{D}$.

Lemmas from [18]. Next we will provide Lemmas from [18] that we will directly use later for re-proving Lemmas for our Algorithm 5.

Lemma 8. *For $\mathcal{A}_{l,j}$ and $\mathcal{A}_{l,j}^{[1]}$ defined in (7) and (9), and for $\eta \geq \eta^{[1]} \geq 2$, the following inequalities hold true*

$$|\mathcal{A}_{l,j}| + |\mathcal{A}_{j,l}| < k, \quad \forall j \neq l, \ j, l \in [\eta] \tag{15}$$

$$|\mathcal{A}_{l,j}^{[1]}| + |\mathcal{A}_{j,l}^{[1]}| < k, \quad \forall j \neq l, \ j, l \in [\eta^{[1]}] \tag{16}$$

where k is the Reed-Solomon encoding parameter.

Proof. Refer to [18, Lemma 7] for proof. □

Lemma 9. *For any graph $G = (\mathcal{P}, \mathcal{E})$ specified by (11)-(13) and for the set $\mathcal{D} \subseteq \mathcal{P}$ defined by (14), and given $n \geq 3t + 1$, it holds true that*

$$|\mathcal{D}| \geq n - 9t/4 - 1. \tag{17}$$

Proof. We refer the reader to [18, Lemma 8] for proof. □

Re-proving Lemmas from [18]. We next argue that at the end of phase 2, if there exists 1 or more group of honest nodes with different messages, then it must hold true that the initial size of each group must be at least $n - 9t/4$. More formally,

Lemma 10. *When $\eta^{[2]} \geq 1$, it holds true that $|\mathcal{A}_l| \geq n - 9t/4$, for any $l \in [\eta^{[2]}]$.*

Proof. The proof consists of the following steps, which mostly follows the proof of [18, Lemma 9]:

- Step (a): Transform the network into a graph that is within the family of graphs satisfying (11)-(13) for a fixed i^* in $\mathcal{A}^{[2]}_{l^*}$ and $l^* \in [\eta^{[2]}]$.
- Step (b): Bound the size of a group of honest nodes, denoted by $\mathcal{D}'$ (with the same form as in (14)), using the result of Lemma 9, i.e., $|\mathcal{D}'| \geq n - 9t/4 - 1$.
- Step (c): Argue that every node in $\mathcal{D}'$ has the same initial message as node i^*.
- Step (d): Conclude from Step (c) that $\mathcal{D}'$ is a subset of $\mathcal{A}_{l^*}$, i.e., $\mathcal{D}' \cup \{i^*\} \subseteq \mathcal{A}_{l^*}$ and conclude that the size of $\mathcal{A}_{l^*}$ is bounded by the number determined in Step (b), i.e., $|\mathcal{A}_{l^*}| \geq |\mathcal{D}'| + 1 \geq n - 9t/4 - 1 + 1$, for $l^* \in [\eta^{[2]}]$.

Step (a): The first step of the proof is to transform the network into a graph that is within the family of graphs defined above. We will consider the case of $\eta^{[2]} \geq 1$. Recall that, when $\eta^{[2]} \geq 1$, we have $|\mathcal{A}^{[2]}_l| \geq 1$ for any $l \in [\eta^{[2]}]$. Let us consider a fixed i^* for $i^* \in \mathcal{A}^{[2]}_{l^*}$ and $l^* \in [\eta^{[2]}]$. Note that,

$$s^2_{i^*} = 1 \Rightarrow \text{At node } i^*, |\mathcal{S}^2_1| \geq n - t \tag{18}$$

$$\Rightarrow \text{At node } i^*, |\mathcal{S}^2_1 \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}| \geq n - 2t \tag{19}$$

$$\Rightarrow \text{At node } i^*, |\mathcal{S}^1_1 \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}| \geq n - 2t \tag{20}$$

the last implication follows from the fact that $j \in \mathcal{S}^2_1 \Rightarrow j \in \mathcal{S}^1_1, \forall j \in \cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p$, which holds due to the check in line 25-26 in Algorithm 5.

Let C' be defined as follows;

$$C' \triangleq \{\mathcal{S}^1_1 \text{ at node } i^* \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}\} \setminus \{i^*\} \Rightarrow |C'| \geq n - 2t - 1 \tag{21}$$

Moreover, since C' is a subset of $\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p$, it implies

$$s^1_j = 1, \forall j \in C' \Rightarrow \text{At every node} j \in C', |\mathcal{S}^1_1| \geq n - t \tag{22}$$

$$\Rightarrow \text{At every node} j \in C', |\mathcal{S}^1_1 \cap \{\cup_{l=1}^{\eta} \mathcal{A}_l\}| \geq n - 2t \tag{23}$$

$$\tag{24}$$

In other words, for any $j \in C'$, node j receives at least $n - 2t$ number of matched observations from honest nodes during phase 1. Let us define a subset of $\{\cup_{l=1}^{\eta} \mathcal{A}_l\} \setminus \{i^*\}$ of honest nodes as

$$\mathcal{D}' \triangleq \{p : p \text{ received matching } \texttt{SYMBOLS} \text{ message from} \tag{25}$$

$$\text{at least } k \text{ nodes in } C' \text{ and } p \in \{\cup_{l=1}^{\eta} \mathcal{A}_l\} \setminus \{i^*\}\} \tag{26}$$

where k is the Reed-Solomon encoding parameter.

Now we map the network into a graph by considering the honest nodes as the vertices and considering the link indicators as edges. Let $\mathcal{P} \triangleq \cup_{l=1}^{\eta} \mathcal{A}_l$. Let $\mathcal{E}$ consists of $E_{i,j}$ for all $i, j \in \mathcal{P}$ such that $E_{i,j} = 1$ if $i \in \mathcal{S}^1_1$ at node j. Note that $i \in \mathcal{S}^1_1$ at node j implies eventually $j \in \mathcal{S}^1_1$ at node i. Let $G = (\mathcal{P}, \mathcal{E})$ be a graph, then $C' \subseteq \mathcal{P}' \setminus \{i^*\}$ be as defined equation (21). It is easy to see that the graph G falls into a family of graphs satisfying (11)-(13).

The step (b), (c), and (d) of our proof is identical to the proof of steps (b), (c), and (d) of [18, Lemma 9]. Thus we omit them for brevity. □

Next we provide a lemma that will be used later for the analysis of the proposed protocol.

Lemma 11. *In algorithm 5 with $n \geq 3t+1$, if $\eta^{[1]} = 2$ then it holds true that $\eta^{[3]} \leq 1$.*

Proof. The proof mostly follows the proof of [18, Lemma 10]. Given $\eta^{[1]} = 2$, the definitions in (1)-(10) imply that

$$\mathcal{A}_1^{[1]} = \{i : s_i^1 = 1, M_i = \bar{M}_1,\ i \notin \mathcal{F},\ i \in [n]\} \tag{27}$$

$$\mathcal{A}_2^{[1]} = \{i : s_i^1 = 1, M_i = \bar{M}_2,\ i \notin \mathcal{F},\ i \in [n]\} \tag{28}$$

$$\mathcal{A}_{1,2}^{[1]} = \{i :\ i \in \mathcal{A}_1^{[1]},\ \bar{M}_1(i) = \bar{M}_2(i)\} \tag{29}$$

$$\mathcal{A}_{1,1}^{[1]} = \mathcal{A}_1^{[1]} \setminus \mathcal{A}_{1,2}^{[1]} = \{i :\ i \in \mathcal{A}_1^{[1]}, \bar{M}_1(i) \neq \bar{M}_2(i)\} \tag{30}$$

$$\mathcal{A}_{2,1}^{[1]} = \{i :\ i \in \mathcal{A}_2^{[1]}, \bar{M}_2(i) = \bar{M}_1(i)\} \tag{31}$$

$$\mathcal{A}_{2,2}^{[1]} = \mathcal{A}_2^{[1]} \setminus \mathcal{A}_{2,1}^{[1]} = \{i :\ i \in \mathcal{A}_2^{[1]},\ \bar{M}_2(i) \neq \bar{M}_1(i)\} \tag{32}$$

$$\begin{aligned}\mathcal{B}^{[1]} &= \{i : s_i^1 = 0,\ i \notin \mathcal{F},\ i \in [n]\} \\ &= \{i : i \in [n],\ i \notin \mathcal{F} \cup \mathcal{A}_1^{[1]} \cup \mathcal{A}_2^{[1]}\}.\end{aligned} \tag{33}$$

In the following we will complete the proof by focusing on the following three cases

$$\text{Case 1:}\quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| \geq t+1 \tag{34}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| < t+1 \tag{35}$$

$$\text{Case 2:}\quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| < t+1 \tag{36}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| \geq t+1 \tag{37}$$

$$\text{Case 3:}\quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| \geq t+1 \tag{38}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| \geq t+1. \tag{39}$$

Note that the following case

$$\text{Case 4:}\quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| < t+1 \tag{40}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| < t+1 \tag{41}$$

does not exist. See [18] for its proof.

<u>*Case 1*</u>: Recall that in the first step of Phase 2, due to the check of line 25-26 in Algorithm 5, each node $i \in \mathcal{A}_{2,2}$ with $s_i^1 = 1$ eventually adds a node j to $\mathcal{S}_0^2$ for every $j \in \mathcal{S}_0^1$. Moreover, by definition eventually every node in $\mathcal{B}^{[1]} \cup \mathcal{A}_1$ will be added to $\mathcal{S}_0^1$ at each node $i \in \mathcal{A}_{2,2}$ because $s_i^1 = 0, \quad \forall i \in \mathcal{B}^{[1]}$ and nodes $\mathcal{A}_1$ and $\mathcal{A}_{2,2}$ have mismatched fragments. Since we assume in (34), that $|\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| \geq t+1$, it implies that eventually during Phase 2, every node $i \in \mathcal{A}_{2,2}$ sets

$$s_i^2 = 0, \quad \forall i \in \mathcal{A}_{2,2}^{[1]} \tag{42}$$

With the outcome in (42) and after exchanging the success indicators, eventually, the set of $\mathcal{A}_{2,2}^{[1]}$, as well as $\mathcal{B}^{[1]}$, will be in the list of $\mathcal{S}_0$ at each honest node. Note that a subset of $\mathcal{A}_{2,1}^{[1]}$, i.e., $\mathcal{A}_{2,1}^{[1]} \cap \{p : s_p^2 = 1\}$) may still be in list of $\mathcal{S}_1^2$. Below we will argue that the complete set of $\mathcal{A}_{2,1}^{[1]}$ will be in the list of $\mathcal{S}_0^3$.

During Phase 3, a node $i \in \mathcal{A}_{2,1}$ and with $s_i^3 = 1$, eventually adds node j to $\mathcal{S}_0^3$ for each $j \in \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}$. This holds due to the check in line 39-40 of Algorithm 5. It is also true that

$$\bar{M}_i(j) \neq \bar{M}_j(j), \quad \forall j \in \mathcal{A}_{1,1}^{[1]},\ i \in \mathcal{A}_{2,1}^{[1]} \tag{43}$$

Note that, the size of $\mathcal{A}_{1,1}^{[1]} \cup \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}$ can be bounded by

$$\begin{aligned} |\mathcal{A}_{1,1}^{[1]} \cup \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}| &= |\mathcal{A}_{1,1}^{[1]}| + |\mathcal{A}_{2,2}^{[1]}| + |\mathcal{B}^{[1]}| && (44)\\ &= n - |\mathcal{F}| - |\mathcal{A}_{1,2}^{[1]}| - |\mathcal{A}_{2,1}^{[1]}| && (45)\\ &\geq n - |\mathcal{F}| - (k-1) && (46)\\ &\geq 2t + 1 - (k-1) && (47)\\ &\geq t + 1 && (48) \end{aligned}$$

where (44) uses the disjoint property between $\mathcal{A}_{1,1}^{[1]}$, $\mathcal{A}_{2,2}^{[1]}$ and $\mathcal{B}^{[1]}$; (45) is from (6) and the disjoint property between $\mathcal{A}_{1,1}^{[1]}$, $\mathcal{A}_{1,2}^{[1]}$, $\mathcal{A}_{2,1}^{[1]}$, $\mathcal{A}_{2,2}^{[1]}$ and $\mathcal{B}^{[1]}$; (46) follows from Lemma 8, which implies that $|\mathcal{A}_{1,2}^{[1]}| + |\mathcal{A}_{2,1}^{[1]}| < k$ (or equivalently $|\mathcal{A}_{1,2}^{[1]}| + |\mathcal{A}_{2,1}^{[1]}| \leq k - 1$); (47) uses the condition that $n \geq 3t + 1$ and $|\mathcal{F}| = t$; (48) results from the fact that $t \geq k - 1$ based on our design of k. Hence, for every node $i \in \mathcal{A}_{2,1}$, during phase 3, eventually $|\mathcal{S}_0^3| \geq t + 1$. Therefore, during phase 3, the node i updates its success indicator $s_i^3 = 0$. Since, $\mathcal{A}_{2,1}^{[1]} \cap \{p : s_p^2 = 0\} \subseteq \mathcal{S}_0$, this implies that eventually $\mathcal{A}_2^{[1]} \subseteq \mathcal{S}_0^3$. Stating differently, at the end of Phase 3 of there exists *at most* 1 group of honest nodes, where the honest nodes within this group have the same non-empty updated message (with success indicators as ones), and the honest nodes outside this group have the same empty updated message (with success indicators as zeros), that is, $\eta^{[3]} \leq 1$, for Case 1.

<u>Case 2</u>: Due to the symmetry between Case 1 and Case 2, one can follow from the proof steps for Case 1 and interchange the roles of Groups $\mathcal{A}_1$ and $\mathcal{A}_2$ (as well as the roles of Groups $\mathcal{A}_1^{[p]}$ and $\mathcal{A}_2^{[p]}$ accordingly for $p \in \{1, 2, 3\}$), to show for Case 2 that at the end of Phase 3 it is that $\mathcal{A}_1^{[1]} \subseteq \mathcal{S}_0$.

<u>Case 3</u>: Follows from the analysis of case 1 and the argument presented in [18]. □