Optimal Extension Protocols for Byzantine Broadcast and Agreement

Chaya Ganesh[1] and Arpita Patra[2]

[1]Aarhus University, ganesh@cs.au.dk
[2]Indian Institute of Science, arpita@iisc.ac.in

Abstract

The problems of Byzantine Broadcast (BB) and Byzantine Agreement (BA) are of interest to both distributed computing and cryptography community. Extension protocols for these primitives have been introduced to handle long messages efficiently at the cost of small number of single-bit broadcasts, referred to as *seed* broadcasts. While the communication optimality has remained the most sought-after property of an extension protocol in the literature, we prioritize both communication and round optimality in this work.

In a setting with n parties and an adversary controlling at most t parties in Byzantine fashion, we present BB and BA extension protocols with $t < n$, $t < n/2$ and $t < n/3$ that are simultaneously optimal in terms of communication and round complexity. The best communication that an extension protocol can achieve in any setting is $\mathcal{O}(\ell n)$ bits for a message of length ℓ bits. The best achievable round complexity is $\mathcal{O}(n)$ for the setting $t < n$ and $\mathcal{O}(1)$ in the other two settings $t < n/2$ and $t < n/3$. The existing constructions are either optimal *only* in terms of communication complexity, or require more rounds than our protocols, or achieve optimal round complexity at the cost of sub-optimal communication. Specifically, we construct *communication-optimal* protocols in the three corruption scenarios with the following round complexities:

- $t < n/3$**:** 3 rounds, improving over $\mathcal{O}(\sqrt{\ell} + n^2)$
- $t < n/2$**:** 5 rounds, improving over 6
- $t < n$**:** $\mathcal{O}(n)$ rounds, improving over $\mathcal{O}(n^2)$

A *concrete* protocol from an extension protocol is obtained by replacing the seed broadcasts with a BB protocol for a single bit. Our extension protocols minimize the *seed-round complexity* and *seed-communication complexity*. The former refers to the number of rounds in an extension protocol in which seed broadcasts are invoked and impacts the round complexity of a concrete protocol due to a number of sequential calls to bit broadcast. The latter refers to the number of bits communicated through the seed broadcasts and impacts the round and communication complexity due to parallel instances of single-bit broadcast. In the settings of $t < n/3$, $t < n/2$ and $t < n$, our protocols improve the seed-round complexity from $\mathcal{O}(\sqrt{\ell} + n^2)$ to 1, from 3 to 2 and from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ respectively. Our protocols keep the seed-communication complexity independent of the message length ℓ and, either improve or keep the complexity almost in the same order compared to the existing protocols.

1 Introduction

In the Byzantine Broadcast (BB) problem, a designated party (called the sender) holds an input message m, and the goal is for all parties to learn m and agree on it. In the related Byzantine Agreement (BA) problem, every party P_i holds a message m_i, and the goal is for all parties to agree on a common message. BB and BA are important primitives used widely in Multi-Party Computation (MPC) and distributed computing protocols in order to reach agreement on some messages. Two important parameters of BB and BA protocols are: communication complexity and round complexity. The communication complexity of a protocol [Yao79] is defined to be the total number of bits sent/received by the *honest* parties during the protocol execution. Note that, only the bits that should be received according to the protocol specification are counted. The round complexity refers to the number of rounds taken by the protocol to terminate.

BB and BA have been studied in two settings: with or without a trusted set-up assumption. In the model where no set-up is assumed, error-free (deterministic) and information-theoretic BA as well as BB is achievable if and only if the number of corrupt parties t is at most $t < n/3$ where n is the total number of parties [PSL80, CW92, BGP92]. The bound cannot be improved with the help of cryptography or randomization [KY86]. In the model where there is a set-up among the parties, BA is achievable for $t < n/2$ and BB is achievable for $t < n$ both cryptographically [DS83] and information theoretically [PW96]. The most popular set-up assumption is a public-key infrastructure (PKI) set-up among n parties. By a PKI set-up among n parties, we mean the following: All parties hold a set of n public keys for a signature scheme where the ith key corresponds to the ith party. Every honest party holds the honestly-generated secret key associated with its own public key. The corrupted parties may generate their keys arbitrarily. The PKI set-up has been considered in two variants- (i) information-theoretic: a PKI set-up for an information-theoretically secure pseudo-signature scheme [PW96], and (ii) cryptographic or computational: a PKI set-up for a cryptographically secure unforgeable digital signature scheme [DS83].

In many distributed computing applications, like reaching agreement on a large file in fault-tolerant distributed storage system, distributed voting where ballots containing gigabytes of data are to be handled [CGS97], MPC [GMW87] where many broadcasts and agreements are invoked, there is an inherent need for dealing with long messages for BB and BA protocols. One could, of course, broadcast (or reach agreement on) a long message by just broadcasting the message bit-by-bit using a single-bit broadcast protocol. This approach leads to huge communication overhead due to the following known bounds. Irrespective of corruption threshold, the seminal result of [DR85] shows that any *deterministic* BB or BA protocol must communicate $\Omega(n^2)$ bits (all-to-all communication). Since the message is at least a single bit, the lower bound on the communication complexity for single bit is $\Omega(n^2)$ bits. The same work shows an upper bound that works for $t < n$ and has a correctness error that is negligible in the security parameter of the underlying digital signature scheme. A line of work [KSSV06, KS09, KS11, KLST11, BSGH13] requiring $t < n/3$ or even stricter bound on t demonstrate how to break the $\Omega(n^2)$ barrier leveraging randomization to induce a partial communication graph (potentially a dynamic one), where the parties choose to initiate communication with one another based on their private randomness. This circumvention is possible necessarily at the cost of achieving correctness with error probability in the *number of parties*. As a result, these works yield scalable solution for *large* size networks where the number of involved parties is *huge*. For protocols that allow no error or negligible error in the security parameter (and not in the number of parties), the lower bound of $\Omega(n^2)$ holds and therefore the trivial approach of reaching consensus bit-by-bit would lead to solutions resulting in $\Omega(\ell n^2)$ bits of

communication.

BB and BA extension protocols[1] are introduced specially to handle long messages and use amortization to beat the communication complexity of the trivial approach. Extension protocols achieve agreement for long messages relying on *point-to-point* communication and a small number of oracle access to a *single-bit broadcast*. Thus in any specific round of an extension protocol, both point-to-point communication as well as seed broadcasts may be invoked. This is conceptually similar to Oblivious Transfer (OT) extension [IKNP03] where a large number of OTs are obtained at the cost of a small number of *seed* OTs and cheap symmetric key operations. We recall that OT [EGL85] is a fundamental building block of MPC – it is a protocol between a sender and a receiver where the sender with two inputs (x_0, x_1) can transmit x_σ to the receiver without knowing σ, where σ is the choice bit of the receiver. Following OT extension literature, we denote the single-bit broadcasts used in the BB and BA extension protocols as *seed* broadcasts. Below, we summarize the results on extension protocols.

1.1 Extension Protocols

Extension protocols for BB and BA are constructions for long messages, built from single-bit seed broadcasts and point-to-point communications. Historically, gaining communication efficiency motivated the study of such protocols with communication $\mathcal{O}(\ell n)$ for message of length ℓ. In any BB extension protocol, since each honest party must learn the message, a correct protocol will incur a communication complexity of at least $\mathcal{O}(\ell n)$ where ℓ is the message length. The same lower bound on the communication complexity holds for BA extension [FH06]. The communication complexity of extension protocols may have a term that depends on ℓ (for large ℓ this term dominates the overall communication complexity) and terms that are independent of ℓ and is only polynomial in parameters such as n and/or security parameter. These terms dictates the lower bound on ℓ for which the optimal communication of $\mathcal{O}(\ell n)$ bits is attained.

It is well-known how to achieve communication optimality in the setting of $t < n/3$ [LV11], $t < n/2$ [FH06] and $t < n$ [HR14]. In fact, except for the first extension protocol of [TC84], the remaining protocols in the literature [FH06, LV11, PR11, PR, HR14, CO18] achieve communication optimality.

With the historical goal of communication efficiency and optimality, the extension protocols often do not prioritise round complexity. In certain scenarios the latency associated with the communication rounds can be a huge bottleneck. While maintaining communication optimality, the proven lower bound on the round complexity of an extension protocol is $\Omega(n)$ for the setting $t < n$ [HR14] and constant when $t < n/2$ [FH06]. The communication-optimal extension protocol of [HR14] for $t < n$ has a round complexity of $\mathcal{O}(n^2)$ which is non-optimal. The round complexity of the communication optimal extension protocol of [LV11] for $t < n/3$ is far from optimal. Namely, the round complexity of [LV11] is $\mathcal{O}(\sqrt{\ell} + n^2)$. In this work, we study and propose protocols that are optimal in both complexity measures of round and communication.

A *concrete* protocol from an extension protocol is obtained by replacing the seed broadcasts with a BB protocol for a single bit. The number of sequential and parallel instances of single-bit BB protocol inherently dictate the complexities of such a concrete protocol. To control the inflation in round and communication complexity due to composition issues in the concrete protocols, we consider two more complexity measures for extension protocols, namely *seed-round* complexity and

[1] In the literature, they are known as multi-valued protocols.

seed-communication complexity. They refer to the number of rounds in which a seed broadcast is invoked, and to the number of bits sent via seed broadcasts. We minimize both these measures and keep the latter independent of message length ℓ. Below, we discuss the known relevant composition results for BB and BA.

1.2 On the composition of BB and BA protocols

The seed-round complexity of an extension protocol directly impacts the round efficiency of its derived concrete protocols via sequential composition. Prior works [KKK08, GGOR13, KK07] on primitives such as variable secret sharing (VSS), multi-party computation (MPC) that use oracle calls to broadcasts, acknowledge this issue, treat broadcast rounds as more expensive than rounds where only point-to-point communications are invoked and seek to minimize what they term broadcast complexity. Indeed, the impact is huge given the state-of-the-art of the round complexity of BB and BA protocols and the impact of sequential composition on round complexity. Any deterministic BB (and BA) protocol necessarily requires $(t + 1)$ rounds [LF82, DR85] and invoking such a protocol sequentially explodes the round complexity by a linear blowup factor. When randomization is introduced, BB and BA protocols achieve a constant round complexity in expected terms [FM97, KK06], yet suffer from non-simultaneous termination leading to substantial increase in expected round complexity due to subtle sequential compositional issues [LLR02, BOEY03, KK06, LLR06, CCGZ16, CCGZ17]. For instance, the expected round complexity of the BB protocols of [KK06] in the $t < n/3$ setting is 23. If there are multiple sequential calls to the BB protocol, then, except the first call which takes expected 23 rounds, each additional call will cost expected 49 rounds. The corresponding figures in the honest majority setting of $t < n/2$ are 56 and 89. Recent work by Micali [Mic17] improves the round complexity for the $t < n/3$ case to an expected 9 rounds, albeit in the computational setting and under the assumption of PKI. The use of randomization does not help much in taming the round complexity in the dishonest majority setting which is considered to be the most practical setting. The proven lower bound on the round complexity for a BB protocol for bit is $\Omega(\frac{n}{n-t})$ [GKKO07]. The best known upper bound presented in [GKKO07] achieves expected $\mathcal{O}(k^2)$ round complexity when $t \leq n/2+k$. In summary, minimizing seed-round complexity irrespective of the corruption threshold remains an important design goal of any protocol that invokes single-bit broadcasts [KKK08, GGOR13, KK07, CCGZ16, CCGZ17].

The existing extension protocols have more than one seed round. The protocol of [HR14] for $t < n$ has a seed round complexity of $\mathcal{O}(n^2)$. The protocol of [FH06] for the $t < n/2$ case has a seed-round complexity of 3. The protocol of [LV11] for the $t < n/3$ case has a seed-round complexity of $\mathcal{O}(\sqrt{\ell}+n^2)$. Our extension protocols improve seed-round complexity in all the settings, in addition to improving the round complexity.

The multiple calls to single-bit broadcasts invoked in a seed round result in parallel instances leading to parallel composition issues in concrete extension protocols. The number of parallel instances is impacted by the seed-communication complexity of the extension protocol. The seed-communication complexity impacts the round as well as communication complexity, yet fortunately restricted to logarithmic in the number of instances in either case [BOEY03, LLR06]. While the result of [BOEY03] begins with a rather pessimistic note that the round complexity of a naïve parallel composition of a expected constant-round BA protocol is (expected) logarithmic in the number of instances, they show a mechanism that preserves constant round complexity in expectation. The above results of [BOEY03] are in the setting of $t < n/3$. In the work of [LLR06], the authors propose a way to preserve (expected) round complexity under parallel composition with

the use of unique identifiers, resulting in a logarithmic blowup in the communication complexity for the setting of $t < n/2$. More recent works [CCGZ16, CCGZ17] deal with composition issues in the $t < n/3$ and $t < n/2$ settings with formal treatment on security definition and composition for randomized BA and BB protocols with non-simultaneous termination. To minimize the effect of parallel composition, our protocols keep seed-communication complexity (which upper bounds the number of parallel calls in any seed round) independent of message length ℓ and further improve, or preserve the same asymptotic order in n and security parameter as the previous protocols.

With the note that the effect of sequential and/or parallel composition issues discussed above are are not unique to this work, but come up in all extension protocols, we proceed to state our results.

1.3 Our Results

We study BA and BB extension protocols with $t < n$, $t < n/2$ and $t < n/3$, and present protocols that are simultaneously communication and round optimal. The existing constructions are either optimal *only* in terms of communication complexity, or require more rounds as well as seed rounds than our protocols, or achieve optimal round complexity by giving up on optimal communication. All our constructions achieve $\mathcal{O}(\ell n)$ bits of communication for sufficiently large message size, and are thus communication optimal (for different bounds on ℓ). Along the way, we also minimize their seed-round and seed-communication complexity.

Table 1: BB and BA Extension Protocols.

Threshold	Security	Communication Complexity	Round Complexity	Seed-Round Complexity	Seed-Communication Complexity	Reference
$t < n/3$	i.t (error-free)	$\mathcal{O}\left(\ell n + (n^2\sqrt{\ell} + n^4)\mathcal{B}(1)\right)$	$\mathcal{O}(\sqrt{\ell} + n^2)$	$\mathcal{O}(\sqrt{\ell} + n^2)$	$n^2\sqrt{\ell} + n^4$	[LV11]
	i.t (error-free)	$\mathcal{O}\left(\ell n + n^2\mathcal{B}(1)\right)$	3	1	n^2	**This paper**
$t < n/2$	i.t	$\mathcal{O}\left(\ell n + n^3\kappa + (n^2 + n\kappa)\mathcal{B}(1)\right)$	6	3	$n^2 + n\kappa$	[FH06]
	crypto	$\mathcal{O}\left(\ell n + n^3\kappa + n\kappa\mathcal{B}(1)\right)$	5	2	$n\kappa$	**This paper**
$t < n$	i.t	$\mathcal{O}\left(\ell n + (n^4 + n^3\kappa)\mathcal{B}(1)\right)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$n^4 + n^3\kappa$	[HR14]
	i.t	$\mathcal{O}\left(\ell n + (n^3\kappa + n^4 \log n)\mathcal{B}(1)\right)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n^3\kappa + n^4 \log n$	[CO18]
	crypto	$\mathcal{O}\left(\ell n + (n^2 + n\kappa)\mathcal{B}(1)\right)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$n^2 + n\kappa$	[HR14]
	crypto	$\mathcal{O}\left(\ell n + (n\kappa + n^3 \log n)\mathcal{B}(1)\right)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$n\kappa + n^3 \log n$	**This paper**

– $t < n/3$**:** Our protocol provides a round complexity of 3 and seed-round complexity of 1. Both the round and seed-round complexity of the best known communication-optimal extension protocol in this setting [LV11] is $\mathcal{O}(\sqrt{\ell}+n^2)$. Since $\ell = \Omega(n^6)$ in their protocol, the complexity translates to $\Omega(n^3)$. The seed-communication complexity of our protocol is n^2 in contrast to $n^2\sqrt{\ell} + n^4$ of [LV11].

– $t < n/2$**:** Our protocol provides a round complexity of 5 and seed-round complexity of 2. The best known extension protocol that is communication-optimal [FH06], has a round complexity of 6 and a seed-round complexity of 3. We improve the seed-communication complexity from $n^2 + n\kappa$ to $n\kappa$ compared to the same construction.

– $t < n$**:** Our protocol has a round complexity and a seed-round complexity of $\mathcal{O}(n)$. Our protocol beats the communication-optimal extension protocol of [HR14] by a factor of $\Omega(n)$ in terms of both round as well as seed-round complexities. Our seed-communication complexity is $n\kappa + n^3 \log n$ compared to $n^2 + n\kappa$ of the cryptographic protocol of [HR14]. Given that the effect of parallel composition is logarithmic in the worst case in the number of parallel

instances on the round or communication complexity [BOEY03,LLR06], our protocol does not suffer from much of a loss in terms of either parameter compared the existing constructions when expanded to a concrete protocol.

The construction with $t < n/3$ is deterministic, *error-free* and information-theoretically secure. The latter implies that the protocol guarantees hold even in the face of a computationally unbounded adversary. The protocols in the other two settings are cryptographic and their correctness reduces to the security of a collision-resistant hash function. The protocol guarantees, therefore hold against a polynomially bounded adversary. Our contributions put in context of other results are summarized in Table 1. We use κ to denote the cryptographic (statistical, respectively) security parameter for the cryptographic (information-theoretic, respectively) primitives. 'i.t' denotes information-theoretic and 'crypto' denotes cryptographic security. Let $\mathcal{B}(l)$ denote the communication complexity of broadcasting an l-bit message.

Our protocol in $t < n/3$ setting can lead to the *first* instantiation without set-up assumption that provides optimal communication complexity and constant round complexity after replacing the seed broadcasts with the single-bit broadcast protocol of [KK06] via the parallel composition result of [BOEY03] that preserves constant expected round complexity. In the $t < n$ setting, our protocol leads to the *first* instantiation in dishonest majority setting with optimal communication complexity and round complexity of $\mathcal{O}(n)$ after replacing the seed broadcasts with the protocol of [GKKO07] for some values of k (e.g. when $t \leq n/2 + k$ and k is a constant).

Our protocol for $t < n$ could only attain a seed-round complexity of $\mathcal{O}(n)$. Designing protocols for $t < n$ that achieve a seed-round complexity of one while preserving communication and round optimality is left as an interesting open question.

1.4 Organization

In Section 2, we discuss the model and definitions of BB and BA protocols. Our results for $t < n/3$, $t < n/2$ and $t < n$ appear in Section 3, Section 4 and Section 5 respectively. We summarize and conclude with questions for further work in Section 6.

2 Models and Definitions

We work in the standard point-to-point network where the set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ are connected by pairwise authenticated channels and communicate in synchronous rounds. The faultiness of the parties is modeled in terms of a monolithic adversary A corrupting t out of the n parties in Byzantine fasion. A can make the corrupted parties deviate from the protocol in any desired manner. The parties who are not under the control of A are referred to as *honest*. We distinguish between cryptographic security and information-theoretic security. Information-theoretic security guarantees that the security properties of the protocol hold even in the presence of a computationally unbounded adversary. When the adversary is bounded, we write PPT to denote a probabilistic polynomial-time algorithm. We use κ to denote the security parameter. A function is negligible if for all large enough values of the input, it is smaller than the inverse of any polynomial. We use negl to denote a negligible function. We now recall some definitions.

Definition 2.1 (Byzantine Broadcast)**.** *A protocol for a set of parties $\mathcal{P} = \{P_1, \cdots, P_n\}$, where a distinguished party called the sender, $P_s \in \mathcal{P}$ holds an initial input m, $|m| = \ell$, is a broadcast protocol tolerating A, if the following properties hold except with negligible probability in κ:*

– *Agreement. All the honest parties output the same value.*
– *Validity. If the sender is honest, all honest parties output the value m.*

Definition 2.2 (Byzantine Agreement)**.** *A protocol for a set of parties $\mathcal{P} = \{P_1, \cdots, P_n\}$, where each party $P_i \in \mathcal{P}$ holds an initial input m_i ($|m_i| = \ell$) is a Byzantine Agreement protocol tolerating* A*, if the following properties hold except with negligible probability in κ:*
– *Agreement. All the honest parties output the same value.*
– *Validity. If every honest party P_i hold the same message $m_i = m$, then all honest parties output the value m.*

In $t < n$ setting, only BB is possible and so we design a BB extension protocol. In the honest majority settings (that includes $t < n/2$ and $t < n/3$), a BB protocol can be obtained by making one call to a BA protocol plus $\mathcal{O}(\ell n)$ bits of communication over point-to-point channels, where ℓ is the length of the message. Therefore, we design a BA extension protocol with various optimal complexity measures. This implies a BB protocol with the optimal complexities in the honest majority settings.

Our protocols in settings $t < n$ and $t < n/2$ (where set-up assumption is required) are cryptographically secure tolerating any polynomially bounded adversary A. Cryptographic or computational security guarantees that the protocol is secure based on some computational assumptions. Our protocols rely on a cryptographic collision-resistant hash function Hash. The collision-resistance property guarantees that it is hard for a polynomially bounded adversary to come up with two preimages of Hash that hash to the same value. A formal definition of collision-resistant hash functions is provided below.

Definition 2.3 (Collision Resistant Hash Functions)**.** *A family of functions $\{\mathsf{Hash}_s\}_{s \in I}$ is a collision resistant hash function family if the following conditions hold:*

1. *Efficient Sampling. There exists a PPT algorithm* Gen *that outputs an index s from the index set I given a security parameter κ, $s \leftarrow \mathsf{Gen}(1^\kappa)$.*

2. *Compression. The function* Hash_s *maps inputs of length n to outputs of length m such that $m < n$.*

3. *Easy to compute. There exists a PPT algorithm that takes an index s, an input $x \in \{0,1\}^n$ and computes $y = \mathsf{Hash}_s(x)$.*

4. *Collision resistance. For every PPT algorithm B,*
$\Pr[\mathsf{Hash}_s(x) = \mathsf{Hash}_s(x'), x \neq x' | x, x' \leftarrow B(s), s \leftarrow Gen(1^\kappa)] = \mathsf{negl}(\kappa)$.

A set-up assumption is needed in the setting $t \geq n/3$ for instantiating the seed broadcasts. Hence, our concrete extension protocols (after instantiation of the oracle) are secure assuming PKI and a collision-resistant hash function. In settings $t < n$ and $t < n/2$, we relax the standard requirements in the definitions of BB and BA above, in that we allow a protocol to fail with probability that is negligible in the security parameter κ of the underlying cryptographic tool. On the other hand, our protocol in the $t < n/3$ setting is *error-free* and is *information-theoretically secure*. The security analysis of our protocols is for a *static* adversary that corrupts parties at the beginning of the protocol.

3 Extension Protocols for $t < n/3$

In this section, we present an error-free and information-theoretic BA and BB extension protocol for the $t < n/3$ setting with: (i) communication complexity: $\mathcal{O}(\ell n)$ bits, (ii) round complexity: 3, (iii) seed-round complexity: 1 and (iv) seed-communication complexity: n^2. We describe an extension protocol for BA. A BB extension protocol with the same complexity as that of the BA protocol can be achieved by letting the sender send the message to all the parties and then running a BA to reach agreement. This is the standard reduction in synchronous settings from BA to BB [Lyn96].

Our protocol relies on techniques from coding theory and graph theory. Specifically, as technical tools, the protocol uses linear error correcting codes (e.g. Reed-Solomon Code) and a graph theoretic algorithm for finding some special structure ((n,t)-star) in an undirected graph. Our approach differs from all existing constructions in this setting which are constructed in player-elimination [HMP00] or dispute-control [BH06] framework. We start with a brief presentation of the tools that we use: (a) An algorithm for finding a graphical structure called (n,t)-star in an undirected graph; (b) Linear Error Correcting Code.

3.1 Building Blocks

Finding (n,t)-star in an Undirected Graph. We now describe an existing solution for a graph theoretic problem, called finding (n,t)-star in an undirected graph $G = (V, E)$. Let G be an undirected graph with the n parties in $\mathcal{P}$ as its vertex set. A pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$ is an (n,t)-star [Can96, BOCG93] in G, if: (i) $|\mathcal{C}| \geq n - 2t$; (ii) $|\mathcal{D}| \geq n - t$; (iii) for every $P_j \in \mathcal{C}$ and every $P_k \in \mathcal{D}$ the edge (P_j, P_k) exists in G.

Following the idea of [GJ79], in [BOCG93], the authors presented an elegant and efficient algorithm for finding an (n,t)-star in a graph of n nodes, *provided that the graph contains a clique of size $n - t$*. The algorithm, called STAR takes the complementary graph $\overline{G}$ of G as input and tries to find (n,t)-$\overline{\text{star}}$ in $\overline{G}$, where (n,t)-$\overline{\text{star}}$ is a pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$, satisfying the following conditions: (a) $|\mathcal{C}| \geq n - 2t$; (b) $|\mathcal{D}| \geq n - t$; (c) There are no edges between the nodes in $\mathcal{C}$ and nodes in $\mathcal{C} \cup \mathcal{D}$ in $\overline{G}$. Clearly, a pair $(\mathcal{C}, \mathcal{D})$ representing an (n,t)-$\overline{\text{star}}$ in $\overline{G}$, is an (n,t)-star in G. STAR outputs either an (n,t)-$\overline{\text{star}}$, or a message `noSTAR`. Whenever *the input graph $\overline{G}$ contains an independent set of size $n - t$*, STAR always outputs an (n,t)-$\overline{\text{star}}$. For simplicity of notation, we denote $\overline{G}$ by H. The algorithm uses matching which is defined below.

Let $G = (V, E)$ be an undirected graph consisting of a set V of vertices and a set E of edges. A *matching* $M \subseteq E$ is a collection of edges such that every vertex of V is incident to at most one edge in M. A maximal matching is a matching M with the property that if any edge not in M is added to M, it is no longer a matching. A maximum matching is a matching that contains the largest possible number of edges; M is a maximum matching, if for any other matching M', $|M| \geq |M'|$.

The algorithm STAR is now presented in Figure 1.

We instantiate the algorithm for finding maximum matching in a general graph with a deterministic algorithm (like [Blu90]) and obtain a deterministic algorithm for finding star.

Linear Error Correcting Code. We use Reed-Solomon (RS) codes [RS60] in our protocols. We consider an $(n, t+1)$ RS code in Galois Field $\mathbb{F} = GF(2^c)$, where $n \leq 2^c$. Each element of $\mathbb{F}$ is represented by c bits. An $(n, t+1)$ RS code encodes $t+1$ elements of $\mathbb{F}$ into a codeword consisting of n elements from $\mathbb{F}$. We denote the encoding function as ENC() and the corresponding decoding function as DEC(). Let $m_0, m_1, \ldots, m_t$ be the input to ENC. Then ENC computes a

Algorithm STAR

– **Input:** An undirected graph $H = (\mathcal{P}, E)$.

– **Algorithm Required:** An algorithm for the maximum matching problem on general graphs.

1. Find a maximum matching M in H. Let N be the set of matched nodes (namely, the endpoints of the edges in M), and let $\overline{N} = \mathcal{P} \setminus N$.
2. Compute output as follows:
   (a) Let $T = \{P_i \in \overline{N} \mid \exists P_j, P_k \text{ s.t } (P_j, P_k) \in M \text{ and } (P_i, P_j), (P_i, P_k) \in E\}$. T is called the set of triangle-heads. Let $\mathcal{C} = \overline{N} \setminus T$.
   (b) Let B be the set of matched nodes that have neighbors in $\mathcal{C}$. So $B = \{P_j \in N \mid \exists P_i \in \mathcal{C} \text{ s. t. } (P_i, P_j) \in E\}$. Let $\mathcal{D} = \mathcal{P} \setminus B$.
   (c) If $|\mathcal{C}| \geq n - 2t$ and $|\mathcal{D}| \geq n - t$, output $(\mathcal{C}, \mathcal{D})$. Otherwise, output `noSTAR`.

Figure 1: Algorithm for Finding (n, t)-star.

codeword of length n, $(s_1, \ldots, s_n)$ as follows. It first constructs a polynomial of degree-t, $f(x) = m_0 + m_1 x + \ldots + m_t x^t$ and then computes $s_i = f(i)$. We use the following syntax for ENC: $(s_1, s_2, \ldots, s_n) = \mathsf{ENC}(m_0, m_1, \ldots, m_t)$. Each element of the codeword is computed as a linear combination of the $t + 1$ input message elements, such that every subset of $(t + 1)$ elements from the codeword uniquely determine the input message elements. Similarly, knowledge of any $t + 1$ elements from the codeword suffices to determine the remaining elements of the codeword.

The decoding function DEC can be applied as long as $t + 1$ elements from a codeword are available. A RS code is capable of error correction and detection. The task of error correction is to find the error locations and error values in a received vector. On the other hand, error detection means an indication that errors have occurred, without attempting to correct them. We will be concerned with Byzantine errors which are errors that are adversarial in nature. That is, c Byzantine errors means c elements of the codeword are arbitrarily changed. We recall the following well known result from coding theory [MS78]. DEC can correct up to u Byzantine errors and simultaneously detect up to additional v Byzantine errors in a vector of length N (where $N \leq n$) if and only if $N - t - 1 \geq 2u + v$. In our protocols, we invoke DEC on a vector of length $N \leq n$ with specific values of u and v. If u, v and N satisfy the above relation, then DEC returns the correct data elements corresponding to the vector. Otherwise, DEC returns 'failure'.

3.2 The BA Protocol

With the above tools, we are ready to present our BA extension protocol. Each party P_i with message m_i containing ℓ bits distributes the codeword of its message among the parties. Each party verifies the part of the codeword received from other parties against its codeword and announces

the outcome in public. The public responses are turned into a consistency graph. Then a special structure in the graph that implies existence of an honest majority set holding the same message is looked for. Namely, the special structure is a quadruple $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$ such that $(\mathcal{C}, \mathcal{D})$ is an (n, t)-star, $|\mathcal{F}| \geq 2t+1$ and every party in $\mathcal{F}$ has at least $t+1$ neighbors in $\mathcal{C}$, $|\mathcal{E}| \geq 2t+1$ and every party in $\mathcal{E}$ has at least $2t+1$ neighbors in $\mathcal{F}$. The novelty then lies in proving that the honest parties in $\mathcal{E}$ hold the same message. The parties then rely on the error correction and detection of the RS code to compute and agree on the common message of the parties in $\mathcal{E}$. If such a set $\mathcal{E}$ does not exist, all honest parties agree on some pre-determined message. The extension protocol is presented in Figure 2. The sm in our notation $\mathcal{P}_{\mathsf{sm}}$ stands for 'same message'.

Protocol $(\frac{\mathsf{n}}{3})$-BA

– **Input of every** P_i**:** An ℓ-bit message m_i.

– **Oracle:** Broadcast oracle for bits.

Every party P_i does the following:

1. Divide the ℓ-bit message m_i into $t+1$ blocks, $m_{i0}, \ldots, m_{it}$, each containing $\frac{\ell}{t+1}$ bits. Compute $(s_{i1}, \ldots, s_{in}) = \mathsf{ENC}(m_{i0}, \ldots, m_{it})$. Send s_{ii} to every party. Send s_{ij} to P_j for $j = [1, n]$.
2. Construct a binary vector $\mathbf{v}_i$ of length n. Assign $\mathbf{v}_i[j] = 1$, if $s_{ij} = s_{jj}$ and $s_{ii} = s_{ji}$ where s_{jj} and s_{ji} are received from P_j. Otherwise assign $\mathbf{v}_i[j] = 0$. Broadcast $\mathbf{v}_i$.
3. Construct graph G using parties in $\mathcal{P}$ as the vertices. Add edge (P_j, P_k) if $\mathbf{v}_j[k] = 1$ and $\mathbf{v}_k[j] = 1$. Invoke STAR $(\overline{G})$ and continue as follows.
   (a) If $(\mathcal{C}, \mathcal{D})$ is returned by STAR, then find $\mathcal{F}$ as the set of parties who have at least $t+1$ neighbours in $\mathcal{C}$ in graph G. Find $\mathcal{E}$ as the set of parties who have at least $2t+1$ neighbours in $\mathcal{F}$ in graph G. If $|\mathcal{E}| \geq 2t+1$, then set $\mathcal{P}_{\mathsf{sm}} = \mathcal{E}$. Otherwise, agree on some predefined message $m^\star$ of length ℓ and abort.
   (b) If `noSTAR` is returned, then agree on some predefined message $m^\star$ of length ℓ and abort.
4. Assign s_i to be the value s_{ji} received from the majority of the parties in $\mathcal{P}_{\mathsf{sm}}$. Send s_i to every party.
5. Let $(s_1, \ldots, s_n)$ be the vector where s_j is received from P_j. Apply DEC on $(s_1, \ldots, s_n)$ with $c = t$ and $d = 0$. Let $m_0, m_1, \ldots, m_t$ be the data returned by DEC. Output $m = m_0 | \ldots | m_t$.

Figure 2: Error-free BA extension protocol in $t < n/3$ setting.

Lemma 3.1. *The honest parties in $\mathcal{P}_{\mathsf{sm}}$ hold the same message of length ℓ.*

Proof. The set $\mathcal{P}_{\mathsf{sm}}$ is the $\mathcal{E}$ component of a quadruple $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$. We start with proving that the honest parties in $\mathcal{C}$ hold the same message of length ℓ. We recall that $\mathcal{D}$ contains at least $t+1$ honest parties and every $P_i \in \mathcal{C}$ is neighbor of every party in $\mathcal{D}$. Let $\{P_{i_1}, \ldots, P_{i_\alpha}\}$ be the set of α honest parties in $\mathcal{D}$, where $\alpha \geq t+1$. Then for every P_i in $\mathcal{C}$, s_{ii_k} is same as $s_{i_k i_k}$ of all $k \in \{1, \ldots, \alpha\}$. Therefore the codewords corresponding to the messages of the honest parties in $\mathcal{C}$ are same at least at $t+1$ locations corresponding to the identities of the honest parties in $\mathcal{D}$. Since the codewords belong to $(n, t+1)$ RS code, the messages of the honest parties in $\mathcal{C}$ are same. Let the common message be m, $|m| = \ell$. Let $(s_1, \ldots, s_n) = \mathsf{ENC}(m_0, m_1, \ldots, m_t)$, where $m = m_0|m_1| \ldots, |m_t$. Now we show that every honest party $P_i \in \mathcal{F}$ holds s_i. Recall that P_i has at least $t+1$ neighbors in $\mathcal{C}$ in which at least one is honest, say P_j. This implies that s_{ii} of P_i is same as s_{ji} of P_j. However, $s_{ji} = s_i$, since P_j holds m. Hence $s_{ii} = s_i$. Therefore every honest P_i in $\mathcal{F}$ holds s_i which is same as s_{ii}. Finally, we show that every honest $P_i \in \mathcal{E}$ holds m. Recall that P_i has at least $2t+1$ neighbors in $\mathcal{F}$ in which at least $t+1$ are honest. Let $\{P_{i_1}, \ldots, P_{i_\alpha}\}$ be the set of α honest parties in $\mathcal{F}$, where $\alpha \geq t+1$. Then s_{ii_k} of P_i is same as $s_{i_k i_k}$ of every honest P_{i_k} for $k \in \{1, \ldots, \alpha\}$. Now $s_{i_k i_k}$ of P_{i_k} is same as s_{i_k}. Therefore the codeword corresponding to the message of $P_i \in \mathcal{E}$ matches with $(s_1, \ldots, s_n)$ at least at $t+1$ locations corresponding to the identities of the honest parties in $\mathcal{F}$. This implies the codeword of P_i is identical to $(s_1, \ldots, s_n)$, since they belong to $(n, t+1)$ RS code. Hence $P_i \in \mathcal{E}$ holds m. □

Lemma 3.2. *If all honest parties start with same input m, then all the parties will agree on $\mathcal{P}_{\mathsf{sm}}$ where $|\mathcal{P}_{\mathsf{sm}}| \geq 2t+1$.*

Proof. All honest parties start with the same input m. Therefore, all honest parties generate the same codeword, $(s_1, \ldots, s_n) = \mathsf{ENC}(m_0, \ldots, m_t)$, such that $m = m_0|m_1| \ldots |m_t$. This means that there will be an edge between every pair of honest parties. In other words, the edges in the complementary graph will be either (a) between an honest and a corrupted party OR (b) between two corrupted parties.

This implies that there will be a clique of size at least $2t+1$. This guarantees the existence of (n,t)-star in G for an honest P_i, and the $\mathcal{C}$ component of an (n,t)-star will contain at least $t+1$ *honest* parties. Subsequently, the $\mathcal{F}$ and $\mathcal{E}$ components will be of size at least $2t+1$. In this case it is guaranteed that all honest parties find the same quadruple $(\mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{E})$ (here we rely on the determinism of STAR algorithm), and will never agree on predefined $m^\star$. From the same quadruple, all honest parties will reach agreement on $\mathcal{P}_{\mathsf{sm}}$.

□

Lemma 3.3. *If $\mathcal{P}_{\mathsf{sm}}$ is agreed on, all honest parties output the common message of the parties in $\mathcal{P}_{\mathsf{sm}}$.*

Proof. By Lemma 3.1, all honest parties in $\mathcal{P}_{\mathsf{sm}}$ hold the same message, say m. This means they hold the same codeword $(s_1, \ldots, s_n) = \mathsf{ENC}(m_0, m_1, \ldots, m_t)$, where $m = m_0|m_1| \ldots, |m_t$. Then every honest P_i in $\mathcal{P}_{\mathsf{sm}}$ already holds s_i, the ith element in the codeword. Now every party P_i will receive s_i correctly as majority of the parties in $\mathcal{P}_{\mathsf{sm}}$ are honest and they will send s_i to P_i. Once every honest P_i holds correct s_i, he sends that to everybody. Therefore a party will receive n values from n parties in which at most t can be wrong (sent by Byzantine corrupted party). However, DEC of $(n, t+1)$ RS code with $n = 3t+1$ allows to correct t errors. Therefore DEC will return $m_0, \ldots, m_t$ such that $m = m_0| \ldots |m_t$.

□

Theorem 3.1. *The protocol* $(\frac{\mathsf{n}}{3})$-BA *satisfies:*

Agreement: Every honest party will output the same message.

Validity: If every honest party P_i *holds the same message* $m_i = m$*, then all honest parties output* m*.*

Complexity: The protocol has a round complexity of 3*, seed-round complexity of* 1*, communication complexity of* $\mathcal{O}\big(\ell n + n^2\mathcal{B}(1)\big)$ *bits and a seed-communication complexity of* $\mathcal{O}(n^2)$ *bits.*

Proof.

Agreement: If $\mathcal{P}_{\mathsf{sm}}$ is agreed, then all honest parties output the common message of the parties in $\mathcal{P}_{\mathsf{sm}}$ (by Lemma 3.3). If $\mathcal{P}_{\mathsf{sm}}$ is not agreed, then all honest parties agree on predefined $m^\star$. Hence, agreement is achieved.

Validity: If all the honest parties start with same m, then by Lemma 3.2, they agree on $\mathcal{P}_{\mathsf{sm}}$ and output m (by Lemma 3.3).

Complexity: Every P_i sends two values s_{ii} and s_{ij} to every other party P_j. The values are $\frac{\ell}{t+1}$ bits long each. Therefore in total there are $\frac{\ell}{t+1}\mathcal{O}(n^2) = \mathcal{O}(n\ell)$ bits of communication. Every party P_i broadcasts n-length binary vector $\mathbf{v}_i$. This leads to total $\mathcal{O}(n^2)$ instances of broadcast for single bit. So the communication complexity is $\mathcal{O}(\ell n + n^2\mathcal{B}(1))$ and the seed-communication complexity is $\mathcal{O}(n^2)$.

The communication takes place in steps 1, 2 and 4 of $(\frac{\mathsf{n}}{3})$-BA with step 2 invoking bit broadcast protocol. So the round and the seed-round complexity of $(\frac{\mathsf{n}}{3})$-BA are 3 and 1 respectively.

□

4 Byzantine Agreement Extension for $t < n/2$

We present a BA extension protocol for ℓ bit message in the honest majority setting with: (i) communication complexity: $\mathcal{O}(\ell n)$ bits, (ii) round complexity: 5, (iii) seed-round complexity: 2 and (iv) seed-communication complexity: $n\kappa$. Given a protocol for BA, a BB protocol can be constructed using the same folklore transformation mentioned in the previous section [Lyn96].

At a high level, our BA protocol closely follows the protocol of [FH06] (will be referred as FH protocol from now onwards) with the main difference being the seed-round complexity. While [FH06] requires three seed rounds, our protocol requires just one round of seed broadcast. A closer look will reveal that it is non-trivial to reduce the number of seed rounds in [FH06]. The FH protocol proceeds in three phases, where each phase brings the parties "closer" to agreement. The protocol may be aborted in the first two phases when some inconsistency is detected. In that case, the parties will output $\perp$. However, if the parties reach the third phase, agreement will be reached without any abort. In each of the first two phases, the parties must agree on whether to abort or to continue to the next phase. Otherwise the BA protocol will have no agreement property. This calls for at least two seed rounds (one in each of the first two phases). The FH protocol requires three seed rounds, one in the first phase and the other two in the second phase. Thus far, there is no known information-theoretic BA extension with optimal communication complexity and one seed round in the honest majority setting. In this paper, we propose a cryptographically secure protocol and leave open the design of an information-theoretic protocol with the same complexity.

Our protocol proceeds in two phases. The first phase denoted as the checking phase is similar to the first phase of the FH protocol. The parties check if there are at least $n - t$ parties who hold the same message, denoted as $\mathcal{P}_{\mathsf{sm}}$ (sm stands for 'same message'). If such a set does not

exist, the parties output $\perp$ and terminate the protocol. This phase consists of a single round and uses broadcast to reach agreement on $\mathcal{P}_{\mathsf{sm}}$ if it exists or on $\perp$ when no such set exists. The communication involves broadcasting the hashes of the individual party's messages and so the communication complexity remains independent of the message size. The second phase denoted as the agreement phase is initiated when the parties have agreed on $\mathcal{P}_{\mathsf{sm}}$. Here, the parties who are not in $\mathcal{P}_{\mathsf{sm}}$ will obtain the common message held by the honest parties in $\mathcal{P}_{\mathsf{sm}}$. The idea is to come up with a set $\mathcal{P}_{\mathsf{hmsm}}$ where the messages held by the honest parties in $\mathcal{P}_{\mathsf{hmsm}}$ and $\mathcal{P}_{\mathsf{sm}}$ are the same. Furthermore, $\mathcal{P}_{\mathsf{hmsm}}$ is guaranteed to have honest majority (hmsm stands for 'honest majority same message'). Now the honest parties in $\mathcal{P}_{\mathsf{hmsm}}$ can together transfer their common message to a party with just $\mathcal{O}(\ell)$ communication complexity using a simple yet clever technique suggested in [FH06]. Lastly, since this phase does not use any broadcast, the honest parties outside $\mathcal{P}_{\mathsf{sm}}$ may have different $\mathcal{P}_{\mathsf{hmsm}}$ sets. But for each such set, the honest majority will be guaranteed and thus the technique will work without any problem. The complete details of the protocol are presented in Figure 3.

We now proceed to the proofs.

Lemma 4.1. *The checking phase satisfies the following properties:*

(a) If all the honest parties P_i start with the same message $m_i = m$, then the honest parties do not abort and output the same set $\mathcal{P}_{\mathsf{sm}}$. Moreover, every honest P_i will belong to $\mathcal{P}_{\mathsf{sm}}$.

(b) All honest parties in $\mathcal{P}_{\mathsf{sm}}$ hold the same input message $m_i = m$ with very high probability.

Proof.

(a) If all the honest parties hold the same message m, then all honest parties broadcast $h = \mathsf{Hash}(m)$. Since there are $n - t$ honest parties and all of them will broadcast a common hash value h, there will be a set of size at least $n - t$ parties whose broadcasted hash values will be the same. So the set $\mathcal{P}_{\mathsf{sm}}$ will exist and the honest parties will not abort the checking phase. Since the hash values are broadcasted, all the honest parties will output the same and unique $\mathcal{P}_{\mathsf{sm}}$. The uniqueness of the agreed set $\mathcal{P}_{\mathsf{sm}}$ is argued as follows. There cannot be two sets of size $n - t$ parties such that one set broadcasts h and the other set broadcast h' with $h \neq h'$. If two such sets exist, it implies that one party has broadcasted both h and h'. But since every party broadcasts one hash value, the set $\mathcal{P}_{\mathsf{sm}}$ is unique and includes all the parties who broadcast h. Clearly the set $\mathcal{P}_{\mathsf{sm}}$ will include all the honest parties.

(b) If two honest parties P_i and P_j in $\mathcal{P}_{\mathsf{sm}}$ hold two different messages $m_i \neq m_j$, then by the collision resistance of the hash function, $\mathsf{Hash}(m_i) \neq \mathsf{Hash}(m_j)$ with high probability and therefore both P_i and P_j cannot belong to $\mathcal{P}_{\mathsf{sm}}$. Hence all honest parties in $\mathcal{P}_{\mathsf{sm}}$ hold the same message.

□

Lemma 4.2. *The agreement phase satisfies the following properties:*

(a) The majority of the parties in $\mathcal{P}_{\mathsf{hmsm}}$ are honest.

(b) The output messages of the honest parties in $\mathcal{P}_{\mathsf{sm}}$ and $\mathcal{P}_{\mathsf{hmsm}}$ are the same.

(c) Every honest party holds the same output message m.

Proof.

(a) Now we show that $\mathcal{P}_{\mathsf{hmsm}}$ has honest majority. Consider the set $\mathcal{P}_{\mathsf{conflict}} = \mathcal{P} \setminus \mathcal{P}_{\mathsf{hmsm}}$. This set consists of pairs of parties $(P_j, \phi(P_j))$. It is not possible that both the parties $(P_j, \phi(P_j))$ are honest. If $\phi(P_j)$ is honest, by Lemma 4.1, she holds a message m that matches with h

and he will send m to P_j. If P_j was honest too, its check $\mathsf{Hash}(m) = h$ will verify and it will send $\mathtt{happy}_j = 1$. Since the pair $(P_j, \phi(P_j))$ is included in $\mathcal{P}^i_{\mathsf{conflict}}$, P_j had sent $\mathtt{happy}_j = 0$ to P_i. This implies either P_j is corrupt and sent $\mathtt{happy}_j = 0$, or $\phi(P_j)$ is corrupt and sent a message not matching h to P_j. Therefore, at least half of $\mathcal{P}_{\mathsf{conflict}}$ are corrupted parties. Since we have honest majority in $\mathcal{P}$, we have that the majority of the parties in $\mathcal{P}_{\mathsf{hmsm}}$ are honest.

(b) By Lemma 4.1, all the honest parties in $\mathcal{P}_{\mathsf{sm}}$ hold the same input message, say m. So o_i for every honest party P_i in $\mathcal{P}_{\mathsf{sm}}$ is equal to m. Now consider an arbitrary *honest* party $P_j \in \mathcal{P}_{\mathsf{hmsm}}$. If $P_j \in \mathcal{P}_{\mathsf{sm}}$, then $o_i = m_i = m$. If $P_j \notin \mathcal{P}_{\mathsf{sm}}$ but in $\mathcal{P}_{\mathsf{hmsm}}$, then P_j must have broadcasted $\mathtt{happy}_j = 1$. This implies that P_j had received some message, say m' from $\phi(P_j) \in \mathcal{P}_{\mathsf{sm}}$ and $h = \mathsf{Hash}(m')$ where h is the hash of the common output message m held by the honest parties in $\mathcal{P}_{\mathsf{sm}}$. This implies $m = m'$ with high probability by collision resistance of the hash function. Since P_j sets $o_i = m'$, the output messages of all the honest parties in $\mathcal{P}^i_{\mathsf{sm}}$ and $\mathcal{P}^i_{\mathsf{hmsm}}$ are the same message.

(c) We will show that every honest party P_j outputs $o_j = m$, where m is the common input message of the honest parties in $\mathcal{P}_{\mathsf{sm}}$. Consider an arbitrary honest party P_j. We have already proved that if $P_j \in \mathcal{P}_{\mathsf{sm}}$ or $P_j \in \mathcal{P}_{\mathsf{hmsm}}$, then $o_j = m$. If P_j belongs neither to $\mathcal{P}_{\mathsf{sm}}$ nor to $\mathcal{P}_{\mathsf{hmsm}}$, then it implies that $\mathtt{happy}_j$ must be 0 and P_j has received a message that is not equal to m from $\phi(P_j)$. We now show that P_j will retrieve m from the parties in $\mathcal{P}_{\mathsf{hmsm}}$. Recall that $\mathcal{P}^j_{\mathsf{hmsm}}$ has honest majority and all the honest parties in it have $o_i = m$. This implies that every honest party P_i's transformed polynomial f_i are identical and correspond to m. We refer to the polynomial as f. If P_j receives d correct y_i values on f, then it can reconstruct f that is a $d-1$ degree polynomial. There are at least d honest parties in $\mathcal{P}_{\mathsf{hmsm}}$ whose hash vectors will be the hash values of $f(1), \ldots, f(n)$. So a piece y_i that is same as $f(i)$ will be accepted by P_j. Whereas y'_i that is not $f(i)$ will be rejected by P_j with high probability. Since there are at least d honest parties in $\mathcal{P}_{\mathsf{hmsm}}$, P_j will always receive d y_i pieces and will reconstruct f and m.

□

Theorem 4.1. *The protocol* $(\frac{\mathsf{n}}{2})$-BA *satisfies the following properties (except with negligible probability in* κ*):*

Agreement: All honest parties output the same value.

Validity: If every honest party P_i *hold the same message* $m_i = m$*, then all the honest parties output* m*.*

Complexity: The protocol has a round complexity of 5*, seed-round complexity of* 2*, a communication complexity of* $\mathcal{O}\left(\ell n + n^3\kappa + n\kappa\mathcal{B}(1)\right)$ *bits and seed-communication complexity of* $\mathcal{O}(n\kappa)$ *bits.*

Proof.

Agreement: If the protocol aborts in the checking phase, then all the parties output $\perp$. Otherwise, all the parties output the same message at the end of agreement phase (Lemma 4.2(c)).

Validity: By Lemma 4.1(a), if all honest parties hold the same message, then all honest parties are in $\mathcal{P}_{\mathsf{sm}}$ and output the same message m.

Complexity: The checking phase communicates $n\mathcal{B}(\kappa)$ bits. In the agreement phase, at most t parties who are outside $\mathcal{P}_{\mathsf{sm}}$ receive messages from the parties in $\mathcal{P}_{\mathsf{sm}}$ as per the mapping ϕ. This requires $\mathcal{O}(\ell n)$ bits of communication. After this step, every party in $\mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ broadcasts a bit $\mathtt{happy}_i$. This requires $n\mathcal{B}(1)$ bits of communication. Finally, for a party

P_i, the set of parties in $\mathcal{P}_{\mathsf{hmsm}}$ makes $\mathcal{O}(\ell + n^2\kappa)$ bits of communication. Since there can be at most t parties in $\mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$, the total communication required is $\mathcal{O}(\ell n + n^3\kappa)$ bits. The total communication complexity of $(\frac{\mathsf{n}}{2})$-BA protocol is $\mathcal{O}\left(\ell n + n^3\kappa + n\kappa\mathcal{B}(1)\right)$ bits. The seed-communication complexity is $\mathcal{O}(n\kappa)$.
The checking phase requires one round which is also a seed round. The agreement phase requires four rounds of communication, in which one round is a seed round. Therefore, the round complexity and the seed-round complexity of $(\frac{\mathsf{n}}{2})$-BA are 5 and 2 respectively.

□

5 Byzantine Broadcast Extension for $t < n$

We present a BB extension protocol for ℓ bit message in the dishonest majority setting with: (i) communication complexity: $\mathcal{O}(\ell n)$ bits, (ii) round complexity: $\mathcal{O}(n)$. The protocol has a seed-round complexity of $\mathcal{O}(n)$. In the same setting, the first communication-optimal BB extension protocol was given by [HR14] (will be referred as HR protocol henceforth). Both the round complexity and seed-round complexity of the HR protocol are $\mathcal{O}(n^2)$. More recently, [CO18] presents an information-theoretic BB extension protocol with the same round and seed-round complexity, extending the works of [GP16,HR14]. Our protocol is the first protocol that optimizes two significant parameters of a BB extension protocol simultaneously.

Like the HR protocol, our construction broadcasts the long message block by block sequentially using dispute control framework [BH06]. While overall communication complexity is guaranteed to be optimal, broadcasting each block may not be done with optimal communication complexity. Our dispute control framework ensures that a corrupted party gets a single chance to misbehave with an honest party. Once it is detected for wrong behaviour by an honest party in some execution of block broadcast, the corrupted party will be ignored by the honest party for the rest of the protocol. By maintaining a history of deviating behaviours across block broadcasts, the dispute control framework help save expensive block communications between parties in dispute. At the heart of dispute control framework lies the art of maintaining history and using it carefully. Such a paradigm calls for sequential execution, and hence, often, the framework trades round complexity for communication complexity. When both the round and the communication overheads are of concern, the framework may not seem to be a wise choice. However, we hit the optimal communication and round complexity at the same time by implementing dispute control framework with overlapped sequential execution. At this point, we note a clear difference between our construction and the HR construction. While the HR block broadcasts run sequentially without any overlap, our construction intertwines the block broadcasts cleverly, yet maintaining the optimal communication complexity. The result is our construction runs just for $\mathcal{O}(n)$ rounds beating the non-optimal $\mathcal{O}(n^2)$ round complexity of the HR protocol.

Specifically, we divide the input message into n blocks. A seed broadcast round is then used to broadcast the hash values of the n blocks. The usage of broadcast ensures all the *honest* parties hold the same copies for the hash values. In an honest block broadcast protocol where everyone behaves honestly, it just requires $n-1$ point-to-point communication of the block in order to propagate it from the sender to the rest of the parties given that the hash value of the message block is already agreed upon and the underlying hash function is collision resistant. But the corrupted parties may behave arbitrarily. Although it may result in requiring more than $n-1$ point-to-point block communications, the honest parties will identify the corrupted parties who misbehaved and will

ignore them for the rest of the protocol. Our protocol (n)-BB is given in Figure 4 which constitutes of two phases: (i) Hash Agreement Phase: It consists of just one round where P_s broadcasts the hashes $h^{(1)}, \ldots, h^{(n)}$ of the n message blocks using oracle access to broadcast bits so that the parties agree on the hash values of the blocks.; (ii) Block Agreement Phase: In this phase, the parties try to obtain the blocks from parties who already received it using *only* point-to-point communication such that the blocks verify against the agreed hash values.

We now concentrate on the block agreement phase where the agreement of blocks are done sequentially yet in an overlapped fashion. Specifically, a party P_i starts requesting for kth message block only when it has received the $(k-1)$th one. This reflects the sequential nature. The overlapping comes from the fact that the kth block agreement for P_i may run in parallel with the $(k-1)$th block agreement for P_j. This stems from the fact that a party P_i proceeds to kth block agreement once it receives the $(k-1)$th block and without waiting for others to receive the same. The protocol runs for $t+n$ rounds, where the earliest round to start asking for kth message block and the latest round to reach agreement on kth block are set to k and $t+k$ respectively. If no block matching the kth hash is received by the end of round $(t+k)$ round, then an honest party exits the while loop and outputs $\perp$. Our protocol guarantees that either all or none of the honest parties will get the message.

Every party maintains three kinds of sets. A local corrupt set $\mathcal{C}_i$ is used by party P_i to log the corrupted parties discovered across the blocks. For the kth block, every party P_i locally maintains a set of happy parties $\mathcal{H}_i^{(k)}$ and a set of unhappy parties $\overline{\mathcal{H}}_i^{(k)}$. From P_i's point of view, a party is happy if it declares to hold/receive a message block matching with the hash value $h^{(k)}$ and it broadcasts certain 'proof' along with the declaration. A party P_i promotes P_j to its happy set only when 'the proof verifies correctly'. Only P_s is considered to be happy initially for all the blocks.

Now let us understand how a particular block agreement is done. Consider the kth block. A party P_i who has received $(k-1)$th block and is still unhappy for the kth block checks if there is a party in its kth happy set $\mathcal{H}_i^{(k)}$ who it can ask for the kth message block. An unhappy (and *honest*) party P_i never requests a party for the message block more than once. Similarly, a happy (and *honest*) P_j does not entertain anyone more than once. To prevent the corrupted parties from making block request from multiple honest parties in a round, every party is made to broadcast the identity of its chosen party. A happy (and *honest*) P_j sends its message block to P_i over the point-to-point channel if and only if it receives a request from P_i via broadcast for the first time for a block. A matrix $\mathbf{T}^{(k)}$ is maintained by each party to detect repeated message send requests. A party who asks for the message block more than once from the same party is identified as corrupted. Next P_i, on receiving a message block from P_j can check if it matches with $h^{(k)}$. As $h^{(k)}$ is generated from a collision-resistant hash function, a corrupted P_j cannot trick an honest P_i by sending a wrong message block and yet pass the consistency check with $h^{(k)}$. Once P_i is happy, it prepares its proof and broadcasts the same. Intuitively, when P_s is corrupted, the proof enables to reach agreement on a block within the round limit fixed for the block. Namely, for the kth round, it is $k+t$. It ensures that the more the adversary delays the receipt of the block by the honest parties, the more it needs to expose the identities of the corrupted parties. We show that an honest party that moves to the kth happy set in round r will know at least $r-k$ corrupted parties. The proof further ensures that the delay cannot be beyond a limit. For the kth block, the first honest party must be allowed to be happy before round $k+t$. Otherwise all the honest parties will remain unhappy and would output a $\perp$ at the end of round $k+t$. More details follow.

Specifically, the proof of P_i for kth block in rth round is a happy set and corrupt set such that

$|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r - k + 1$ (excluding itself). If P_s is honest, then every honest party will get promoted to the kth happy sets of all the honest parties in the kth round itself as each one of them can prepare a proof where the happy sets consists of P_s. For a corrupted P_s, the proof ensures that the first entry of an honest party in the kth happy sets of the honest parties *cannot* be in $(k+t)$th round. It must be in one of the previous rounds, because the proof in $(k+t)$th round requires $t+1$ distinct parties in the union of happy set and the corrupt set. Since an honest party will not belong to the corrupt set of another honest party, apart from self, the first honest entrant must include another honest party in its kth happy set. Therefore, there must be some other honest party who has become happy before $(k+t)$th round. We then show that if an honest party becomes happy in round r for the kth block such that $k \leq r < k + t$, then all the honest parties will be happy after running round $(k + t)$.

Lastly, to ensure that a corrupted party does not fake its proof by including parties that are not happy or corrupt, the proof is verified by checking if it is a subset of the union of the happy set and corrupt set of the verifier. An honest party's proof will always get verified by another honest party. We show that there is a one-to-one correspondence between the sets $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ and $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j$ of two honest parties P_i and P_j.

We now proceed to the proofs.

Lemma 5.1 (Complexity)**.** *Protocol* (n)-BB *has:*
(i) *a round complexity of* $\mathcal{O}(n)$*,*
(ii) *a seed-round complexity of* $\mathcal{O}(n)$*,*
(iii) *a communication complexity of* $\mathcal{O}\big(\ell n + (n\kappa + n^3 \log n)\mathcal{B}(1)\big)$ *bits and*
(iv) *a seed-communication complexity of* $\mathcal{O}(n\kappa + n^3 \log n)$ *bits.*
for a message of ℓ*-bit length.*

Proof. It is easy to verify the round complexity and the seed-round complexity of the protocol.

We now compute the communication complexity by considering the communication received by each party. First, let us consider the communication received by an honest party. For every block of message, if an honest party eventually moves to the happy set, then it receives $\frac{\ell}{n}$ bits of message in order to move from the unhappy set to happy set. Across all the message blocks, it may in addition receive $\frac{\ell}{n}$ bits of wrong message block from each of the t corrupted parties. However, each such receipt will reveal the identity of one corrupted party. Thus, in total, the amount of communication received by an honest party in the protocol is $\mathcal{O}(\ell)$ bits. Now consider the communication received by a corrupted party. We need to focus on the communication made by the honest parties since a corrupted party can communicate as many bits as it wants to another corrupted party. A corrupted party cannot make an honest party to communicate any message block twice. Every honest party P_i keeps track of the list of parties it sends a message block to in the array $\mathbf{T}$. Once it sends the kth message block to P_x, it sets $\mathbf{T}^{(k)}[i, x]$ to 0. Thus, a corrupted party either moves to the happy set or creates a conflict with the honest party P_i. Irrespective of the case, the honest party P_i will not communicate the same message block to the same corrupted party for the second time. Thus, across all message blocks, a corrupted party can receive $\frac{\ell}{n}$ bits of message from each honest party to create conflicts with them. To move from unhappy set to happy set for each message block, it may receive $\frac{\ell}{n}$ bits of message from some honest party. Overall it counts to receipt of $\mathcal{O}(\ell)$ bits. Now counting over all parties, we get that the overall received message complexity is $\mathcal{O}(\ell n)$ bits. Since the communication complexity is same as the received bit complexity, we conclude that the communication complexity is $\mathcal{O}(\ell n)$ bits.

We now count the part of the communication complexity that is independent of ℓ and is broadcasted. In the Hash Agreement phase, P_s broadcasts n hash values corresponding to n message blocks. This requires a communication of $\mathcal{B}(n|h|)$ bits. Next, we consider the communication involved in the Block Agreement phase. For kth block, every party P_i will broadcast $(\texttt{happy}, \mathcal{H}_i^{(k)}, \mathcal{C}_i, k)$ only in the round when it moves from its unhappy set to happy set. In the worst case, it may broadcast $(\texttt{send}, j, \mathsf{c}_i)$ and $(\texttt{unhappy}, \mathsf{c}_i)$ in every round. We may assume the size of $\mathcal{H}_i^{(k)}$ and $\mathcal{C}_i$ to be $\mathcal{O}(t)$. This results in a communication of $\mathcal{B}(n \log n)$ bits per party per message block. Summing over all the blocks and all the parties, we get that a communication of $\mathcal{B}(n^3 \log n)$ bits is required. Assuming $|h^{(k)}| = \kappa$, the communication that is independent of ℓ turns out to be $\mathcal{B}(n\kappa + n^3 \log n)$ bits. Thus the ℓ independent communication complexity is $\mathcal{O}((n\kappa + n^3 \log n)\mathcal{B}(1))$ bits.

The communication complexity of protocol (n)-BB is, thus, $\mathcal{O}\big(\ell n + (n\kappa + n^3 \log n)\mathcal{B}(1)\big)$ bits and the seed-communication complexity is $\mathcal{O}(n\kappa + n^3 \log n)$ bits. □

Lemma 5.2 (Validity)**.** *Assume that* Hash *is a collision-resistant hash function. In protocol* (n)-BB*, if P_s is honest then every honest party will output sender's message.*

Proof. The validity follows from the fact that all the honest parties ask for and receive the kth message block from P_s in the kth round of the *while* loop and move to the kth happy set $\mathcal{H}^{(k)}$ of all the honest parties. Formally, the proof goes as follows. Before the start of the *while* loop, $\mathcal{H}_i^{(k)}$ is set to P_s for all $k \in \{1, \ldots, n\}$. Consider the kth message block. In the kth round of the *while* loop, the condition $|\mathcal{H}_i^{(k)}| \geq r - k + 1$ will hold for any honest party P_i, since both $|\mathcal{H}_i^{(k)}|$ and $r - k + 1$ are equal to 1. The latter is true because we are considering the kth round and hence $r = k$. Every honest party P_i does the following in the given order (i) It asks for the kth message block from P_s and will receive the same; (ii) It broadcasts $(\texttt{happy}, \mathcal{H}_i^{(k)}, \mathcal{C}_i, k)$ where, $\mathcal{H}_i^{(k)} = \{P_s\}$, sets $o_i^{(k)}$ to $m^{(k)}$ and increments block count c_i by one; (iii) At the end of kth round, it outputs $o_i^{(k)}$ which is the same as $m^{(k)}$, since P_s is honest. At the end of protocol (n)-BB, every honest P_i will output $m^{(1)}| \ldots |m^{(n)}$ which is same as the sender's message m. □

We now prove the agreement property via a sequence of lemmas on the (n)-BB protocol.

Lemma 5.3. *At any step during round r for $1 \leq r \leq n + t$, the condition $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i \subseteq \mathcal{H}_j^{(k)} \cup \mathcal{C}_j$ is true for any two honest parties P_i and P_j for any $k \in \{1, \ldots, n\}$.*

Proof. For any round r, we proceed step by step and show that in each step the sets $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i$, $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j$ grow together.

Step 2a: Step 2a does not update any of the sets.

Step 2b: In step 2b, only the corrupt sets may grow. If a party P_α enters the set $\mathcal{C}_i$ in step 2b, then it enters set $\mathcal{C}_j$ too in the same step. P_α enters $\mathcal{C}_i$ for an honest P_i because of the following reason. P_i has received P_α's broadcast $(\texttt{send}, x, y)$ such that one of the following is true. Either $\mathbf{T}_i^{(y)}[x, \alpha]$ is already set to 0 (i.e. P_α has asked for the kth message block from P_x earlier) or there is more than one broadcast request initiated by P_α in this round. In either case, an honest P_j will also add P_α to its corrupt set $\mathcal{C}_j$.

Step 2c: In step 2c, only the corrupt sets may grow. Here an honest P_i includes P_α in its corrupt set because of the following reason. P_i had requested P_α for the kth message block but did not receive one matching $h^{(k)}$. Then it must be the case that P_α belongs to the set $\mathcal{H}_i^{(k)}$ of P_i in step 2d of one of the rounds $\{1, \ldots, r-1\}$, say r'. Below, we show that for party P_j, P_α is in set $\mathcal{H}_j \cup \mathcal{C}_j$ in step 2d of r'.

Step 2d: Here both the happy and corrupt sets may grow.

Consider any party P_α that enters $\mathcal{H}_i^{(k)}$ in this step of any round r. We show that P_α will belong to $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j$ in the same round r. Since P_α enters $\mathcal{H}_i^{(k)}$ in round r, the following must be true: (i) P_i has received P_α's broadcast $(\texttt{send}, x, y)$ such that $\mathbf{T}_i^{(y)}[x, \alpha] = 1$ and there is only one broadcast initiated by P_α in this round; and (ii) P_i has received P_α's broadcast $(\texttt{happy}, \mathcal{H}_\alpha^{(k)}, \mathcal{C}_\alpha, k)$ such that the conditions $\mathcal{H}_\alpha^{(k)} \cup \mathcal{C}_\alpha \subseteq \mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ and $|\mathcal{H}_\alpha^{(k)} \cup \mathcal{C}_\alpha| \geq r - k + 1$ hold. P_j will also receive P_α's broadcast $(\texttt{send}, x, y)$ such that $\mathbf{T}_i^{(y)}[x, \alpha] = 1$ and this will be the only broadcast initiated by P_α in this round. Furthermore, P_j also receives P_α's broadcast $(\texttt{happy}, \mathcal{H}_\alpha^{(k)}, \mathcal{C}_\alpha, k)$. Now, if $\mathcal{H}_\alpha^{(k)} \cup \mathcal{C}_\alpha \subseteq \mathcal{H}_j^{(k)} \cup \mathcal{C}_j$, then P_j will move P_α to its happy set. Otherwise, P_j will add P_α to $\mathcal{C}_i$. It can never happen that P_α is moved to $\mathcal{H}_i^{(k)}$ by P_i, but P_j has neither moved it to $\mathcal{H}_j^{(k)}$ nor added to $\mathcal{C}_j$.

Now consider any party P_α that enters $\mathcal{C}_i$ of P_i in this step of any round r. This will happen when P_i has received P_α's broadcast $(\texttt{happy}, \mathcal{H}_\alpha^{(k)}, \mathcal{C}_\alpha, k)$ such that either of the conditions $\mathcal{H}_\alpha^{(k)} \cup \mathcal{C}_\alpha \subseteq \mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ and $|\mathcal{H}_\alpha^{(k)} \cup \mathcal{C}_\alpha| \geq r - k + 1$ are not true. P_j will also receive the same broadcast. Based on whether the conditions are true for P_j or not, P_j will either add P_α to $\mathcal{C}_j$ or move it to $\mathcal{H}_j^{(k)}$. But it cannot happen that P_j has not added P_α to either of the two sets.

□

Definition 5.1 (Validity of $\mathcal{C}_i$)**.** *We say that a set $\mathcal{C}_i$ is valid if for honest P_i, and for all honest parties P_j, it holds that $P_j \notin \mathcal{C}_i$.*

Below we prove that protocol (n)-BB ensures $\mathcal{C}_i$ remains valid throughout for an honest P_i.

Lemma 5.4. *The set $\mathcal{C}_i$ is valid for an honest party P_i.*

Proof. We note that an honest party adds a party P_j in corrupt set $\mathcal{C}_i$ when either of the following are true: (i) it receives multiple broadcasts from P_j of the form $(\texttt{send}, x, y)$ in different rounds or in the same round or (ii) it receives multiple broadcasts from P_j of the form $(\texttt{send}, x, y)$ and $(\texttt{send}, x', y')$ in the same round where $x \neq x'$ or (iii) it did not receive a message block from P_j upon request that matches with the corresponding hash value or (iv) it receives $(\texttt{happy}, \mathcal{H}_j^{(k)}, \mathcal{C}_j, k)$ for some k from P_j in round r such that either $|\mathcal{H}_j^{(k)} \cup \mathcal{C}_j| < r - k + 1$ or $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j \not\subseteq \mathcal{H}_i^{(k)} \cup \mathcal{C}_i$. P_j is clearly corrupted when any of the conditions stated in (i)-(iii) hold. Now, for the conditions in (iv), if P_j was honest it would not broadcast $(\texttt{happy}, \mathcal{H}_j^{(k)}, \mathcal{C}_j, k)$ such that $|\mathcal{H}_j^{(k)} \cup \mathcal{C}_j| < r - k + 1$. By Lemma 5.3, for any two honest parties P_i and P_j, $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j \subseteq \mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ is true at the end of step 2c or in the beginning of step 2d when $(\texttt{happy}, \mathcal{H}_j^{(k)}, \mathcal{C}_j, k)$ is broadcasted. If it is not true, then P_j must be corrupt and has not broadcasted the correct set. □

The next lemma makes the following statements equivalent: For any k, in the kth block broadcast (i) an honest party moves from its kth unhappy set to kth happy set in round r (ii) an honest party moves from the kth unhappy set to the kth happy set of *every* honest party in round r.

Lemma 5.5. *For any k, if an honest party P_i moves from $\overline{\mathcal{H}}_\alpha^{(k)}$ to $\mathcal{H}_\alpha^{(k)}$ for an honest party P_α in the rth round, then it moves from $\overline{\mathcal{H}}_\beta^{(k)}$ to $\mathcal{H}_\beta^{(k)}$ for every other honest party P_β in the same round.*

Proof. We note that the test done by P_α and P_β for promoting P_i to their happy sets are identical: P_α checks $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i \subseteq \mathcal{H}_\alpha \cup \mathcal{C}_\alpha$ while P_β checks $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i \subseteq \mathcal{H}_\beta^{(k)} \cup \mathcal{C}_\beta$. It follows from Lemma 5.3 that both will hold. □

We now prove the following lemma: For any k, if an honest and unhappy party moves a party to its kth happy set in the rth round, then it will know at least $r - k + 1$ corrupted parties by the end of the rth round where $k \leq r \leq k + t$.

Lemma 5.6. *For any k, if an honest party P_i is in $\mathcal{H}_i^{(k-1)}$ before rth round but in $\overline{\mathcal{H}}_i^{(k)}$ till the end of rth round and moved a party from $\overline{\mathcal{H}}_i^{(k)}$ to $\mathcal{H}_i^{(k)}$ in the rth round where $k \leq r \leq k + t$, then $|\mathcal{C}_i| \geq r - k + 1$ in the end of rth round.*

Proof. We prove the lemma using strong induction on r. We start with the following observation. Let P_j be the party that moved from $\overline{\mathcal{H}}_i^k$ to $\mathcal{H}_i^k$ in the rth round. As per the protocol steps, this implies that the conditions $\mathcal{H}_j^{(k)} \cup \mathcal{C}_j \subseteq \mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ and $|\mathcal{H}_j^{(k)} \cup \mathcal{C}_j| \geq r - k + 1$ are true at Step 2d of round r. From these two conditions, we can conclude that $|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r - k + 1$ at Step 2d of round r (excluding P_j).

Base Case. Assume r to be the first round when P_i moved at least one party from $\overline{\mathcal{H}}_i^{(k)}$ to $\mathcal{H}_i^{(k)}$ and P_i is in $\overline{\mathcal{H}}_i^k$ till the end of rth round. We have $k \leq r \leq k + t$. Since before rth round no party has moved to $\mathcal{H}_i^{(k)}$, $\mathcal{H}_i^{(k)}$ consists of only P_s. Furthermore, P_s must be corrupted as P_i is still unhappy. Therefore, all the parties in $\mathcal{H}_i^{(k)}$ are corrupted. Since we have $|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r - k + 1$ at Step 2d of rth round, we can conclude that $|\mathcal{C}_i| \geq r - k + 1$ in round r.

Induction Step. Assume the lemma statement is true for all the rounds starting from k to $r - 1$. We will prove the statement for round r.

Now, assume that r' is the last round before r when P_i moved at least one party from $\overline{\mathcal{H}}_i^{(k)}$ to $\mathcal{H}_i^{(k)}$. We have $r' < r$. Since P_i must have been unhappy in round r' too, we have $|\mathcal{C}_i| \geq r' - k + 1$ by the end of round r' via induction hypothesis. According to the protocol, the following condition will be satisfied for P_i in the end of round r': $|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r' - k + 2$ after moving P_j to the happy set. Now we have two cases to consider.

(i) First, if P_i asks for the kth message block from some party in its happy set $\mathcal{H}_i^{(k)}$ in every round starting from round r' until round r, then P_i discovers $r - (r' - 1) + 1 = r - r'$ additional corrupt parties since it remains unhappy in the end of rth round. We therefore conclude that $|\mathcal{C}_i| \geq (r' - k + 1) + (r - r') = r - k + 1$ in this case.

(ii) The other possibility is that P_i asks for the kth message block for some rounds starting from round $r' + 1$ but stops asking before round r. By assumption, the happy set $\mathcal{H}_i^{(k)}$ of P_i did

not grow between r'th and rth round. In this case, we can conclude that all the parties that belong to $\mathcal{H}_i^{(k)}$ until the beginning of round r are corrupted since none of them delivered P_i a message block that is consistent with $h^{(k)}$. Since we have $|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r - k + 1$ at Step 2d of rth round and all the parties in $\mathcal{H}_i^{(k)}$ until the beginning of round r are corrupted, we have $|\mathcal{C}_i| \geq r - k + 1$.

□

We now prove a lemma that captures the fact that, the more an honest party is delayed in receiving a message block, the more the number of corrupted parties it discovers.

Lemma 5.7. *For any k, if an honest party P_i moves to $\mathcal{H}_i^{(k)}$ in rth round where $k \leq r \leq k + t$, then $|\mathcal{C}_i| \geq r - k$ in the end of rth round.*

Proof. We first prove the lemma assuming that P_i has not moved any party to $\mathcal{H}_i^{(k)}$ before round r. This means $\mathcal{H}_i^{(k)}$ contains P_s alone and P_i must have become happy on receiving the kth message block from P_s. This implies two things: first, it must have received all the previous message blocks from P_s too; second, r must be k. In this case, P_i knows zero corrupted parties. Since $r = k$, we have $|\mathcal{C}_i| \geq r - k$ at the end of round r. We now prove the lemma for the case when P_i does not receive the message blocks from P_s and has moved some party to $\mathcal{H}_i^{(k)}$ before rth round. We prove the lemma using induction on k.

Base Case. Assume $k = 1$. That is, we are considering the first message block. P_i has moved some party to $\mathcal{H}_i^{(1)}$ before rth round. Let r' be the last round when P_i does so. We have $1 \leq r' < r$. By Lemma 5.6, P_i knows at least $r' - k + 1 = r'$ corrupted parties by the end of r'th round. According to the protocol, by the end of round r', $|\mathcal{H}_i^{(1)} \cup \mathcal{C}_i| \geq r' - k + 2 = r' + 1$. By the assumption of the statement P_i is honest and it has not added any party to its happy set after round r'. Now, it has at least one party to ask for the message starting in round $(r' + 1)$ and it moves to its happy set only in round r. Therefore, P_i will have parties in $\mathcal{H}_i^{(1)}$ that are not in $\mathcal{C}_i$ to ask for the message block. Since P_i has become happy in rth round, it must have been asking distinct parties for the message block, starting from round $(r' + 1)$ and upto round r. None of the parties that it asked for the message until round $r - 1$ delivered it a message block matching $h^{(1)}$. This implies it discovers $(r - 1) - (r' + 1) + 1 = (r - r' - 1)$ corrupted parties starting from round $(r' + 1)$ and until round $(r - 1)$. These identities are distinct from the r' parties it knew by the end of round r'. In total, P_i knows at least $r' + (r - r' - 1) = r - 1$ corrupted parties by the end of round r. Since $k = 1$, we have $|\mathcal{C}_i| \geq r - 1 = r - k$ at the end of round r.

Induction Step. Assume the lemma is true for all the message blocks upto $k - 1$. We now prove the lemma for the kth message block. Say, P_i moves to $\mathcal{H}_i^{(k-1)}$ in r_{k-1}th round where $(k - 1) \leq r_{k-1} \leq (k - 1 + t)$. By induction hypothesis, $|\mathcal{C}_i| \geq r_{k-1} - (k - 1)$ by the end of r_{k-1}th round. Now we consider two cases based on when P_i moves some party to $\mathcal{H}_i^{(k)}$ before round r.

(i) The last time P_i moves some party to its happy set $\mathcal{H}_i^{(k)}$ is in or before r_{k-1}th round. We know that P_i had been asking for kth message block in every round starting from round $(r_{k-1} + 1)$ to round r. We also know that P_i has become happy only in round r. Until round $(r - 1)$, it

discovers $(r-1)-(r_{k-1}+1)+1 = r-r_{k-1}-1$ corrupted parties which are different from $r_{k-1}-k+1$ corrupted parties it knew by the end of round r_{k-1}. In total P_i now knows $(r-r_{k-1}-1)+(r_{k-1}-k+1) = r-k$ corrupted parties at the end of rth round.

(ii) The last time P_i moves some party to its happy set $\mathcal{H}_i^{(k)}$ is after r_{k-1}th round, say, in round r' such that $r_{k-1} < r' < r$. By Lemma 5.6, P_i knows at least $r'-k+1$ corrupted parties by the end of r'th round. According to the protocol, by the end of round r', $|\mathcal{H}_i^{(k)} \cup \mathcal{C}_i| \geq r'-k+2$. Hence, P_i will have parties in $\mathcal{H}_i^{(k)}$ that are not in $\mathcal{C}_i$ to ask for the message block starting from round $(r'+1)$. Now, since P_i became happy in rth round, it must have been asking distinct parties for the message block starting from round $(r'+1)$ and upto round r. None of the parties that it asked for the message until round $r-1$ delivered it a message block matching with $h^{(k)}$. This implies it discovers $(r-1)-(r'+1)+1 = (r-r'-1)$ corrupted parties starting from round $(r'+1)$ and until round $(r-1)$. These identities are distinct from the r' parties it knew by the end of round r'. In total, P_i knows at least $(r'-k+1)+(r-r'-1) = r-k$ corrupted parties by the end of round r. We have $|\mathcal{C}_i| \geq r-k$ at the end of round r.

□

The following lemma states that for any k, if one honest party moves to the kth happy sets of the honest parties in rth round where $k \leq r < k+t$, then every honest party will move to the kth happy sets of the honest parties before round $(k+t)$. This lemma will let us prove that either all or none of the honest parties will be happy for the kth message block which in turn will lead us to the proof of the agreement property of our protocol (n)-BB.

Lemma 5.8. *For any k, if some honest party P_i moves from $\overline{\mathcal{H}}_j^{(k)}$ to $\mathcal{H}_j^{(k)}$ for every honest party P_j in round r such that $k \leq r < k+t$, then every honest party will move from $\overline{\mathcal{H}}_j^{(k)}$ to $\mathcal{H}_j^{(k)}$ before $(k+t)$th round.*

Proof. Let P_i be the first honest party that moves to the kth happy set $\mathcal{H}_j^{(k)}$ of honest P_j. Also let P_i move to the happy set $\mathcal{H}_j^{(k)}$ of P_j in round r such that $k \leq r < k+t$. We now show that P_j will move to the kth happy sets of all the parties in or before $(k+t)$th round. Let r_{k-1} be the round number when P_j has moved to $\mathcal{H}_j^{(k-1)}$ where $(k-1) \leq r_{k-1} \leq (k-1)+t$. Now we complete our proof by considering the following two possible cases:

(i) $r_{k-1} < r$*:* By Lemma 5.6, P_j knows at least $r-k+1$ corrupted parties by the end of rth round. At the end of rth round, honest P_j will have $|\mathcal{H}_j^{(k)} \cup \mathcal{C}_i| \geq r-k+2$ after including P_i. Therefore, P_j will have at least one party to ask for the message block in $(r+1)$th round. Now, if P_j talks to P_i in some round in or before $(k+t)$th round, then it moves to its happy set $\mathcal{H}_j^{(k)}$ after receiving the message block. So let us assume that P_j does not ask for the kth message block from P_i. This implies that P_j was requesting for the message block from parties other than P_i in each of the rounds starting from the $(r+1)$th round. It is not possible that P_j has not asked for the block from anyone in some round after the rth round. This is because P_i will be in its happy set post rth round and by Lemma 5.4, P_i will not be in $\mathcal{C}_j$ as both P_i and P_j are honest. We therefore conclude that P_j talks to some party in its kth happy set in each of the rounds starting from the $(r+1)$th round and until $(k+t)$th round.

Now P_j talks to $(k+t)-(r+1)+1 = k+t-r$ parties in its happy set starting from $(k+x)$th round to $(k+t)$th round. These $k+t-r$ parties are different from the $r-k+1$ corrupted parties P_j had in its corrupt list $\mathcal{C}_j$ at the end of rth round. If all the $k+t-r$ parties are corrupted, then total number of corrupted parties will be $x+t-x+1 = t+1$. This is a contradiction. One out of the $k+t-r$ parties must, therefore, be honest and belong to P_j's happy set $\mathcal{H}_j^{(k)}$. After getting the message block from the honest party in its happy set $\mathcal{H}_j^{(k)}$, P_j will move to its happy set too.

(ii) $r_{k-1} \geq r$*:* By Lemma 5.7, P_j will know at least $r_{k-1}-(k-1)$ corrupt parties by the end of round r_{k-1}. At the start of round $(r_{k-1}+1)$, P_i has $|\mathcal{C}_j| \geq r_{k-1}-(k-1)$ and its happy set $\mathcal{H}_j^{(k)}$ contains honest P_i that cannot belong to $\mathcal{C}_j$ by Lemma 5.4. So clearly $|\mathcal{H}_j^{(k)} \cup \mathcal{C}_j| \geq r_{k-1}-(k-1)+1 = (r_{k-1}+1)-k+1$ holds good in the beginning of round $(r_{k-1}+1)$. This implies P_j will ask for kth message block starting from round $(r_{k-1}+1)$. Now if it asks for the kth message block from P_i in some round before $k+t$, then it gets promoted to $\mathcal{H}_j^{(k)}$. If it does not ask P_i, then it must be asking other parties in its kth happy set in every round starting from round $(r_{k-1}+1)$ and until round $(k+t)$. By the end of $k+t$ rounds, if it has not got promoted to the kth happy set, then it discovers $(k+t)-(r_{k-1}+1)+1 = k+t-r_{k-1}$ corrupted parties. These are different from $r_{k-1}-(k-1)$ corrupt parties P_j knew by the end of round r_{k-1}. Therefore, the total corrupted parties will have to be $k+t-r_{k-1}+r_{k-1}-(k-1) = t+1$. This is a contradiction. Hence, we conclude P_j will talk to some honest party and get promoted to $\mathcal{H}_j^{(k)}$ by the end of round $(k+t)$.

Recall that by Lemma 5.5, an honest party will be promoted to the kth happy set respectively by all honest parties in the same round. P_j, will therefore, be promoted to the kth happy sets of all the honest parties. Now *every* honest P_j will move to the kth happy sets of all the parties since honest P_i will move to the kth happy set of every honest P_j in the same round (again due to Lemma 5.5). □

Lemma 5.9. *For any k, either all or none of the honest parties will be in the kth happy sets of the honest parties.*

Proof. By Lemma 5.8, if one honest party moves to the kth happy sets of the honest parties in rth round where $k \leq r < k+t$, then every honest party will move to the kth happy sets of the honest parties on or before $(k+t)$th round of the *while* loop. We now show that if at all an honest party moves to the kth happy set of the honest parties, the first such move happens in round r where $k \leq r < k+t$. Assume that P_i is the first honest party to enter to the kth happy sets of the honest parties. Also assume that P_i is moved to the kth happy sets in the $(k+t)$th round. Then P_i must have at least $(k+t)-k+1 = t+1$ parties in $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i$ excluding itself during the broadcast of $(\texttt{happy}, \mathcal{H}_i^{(k)}, \mathcal{C}_i, k)$. Since there are at most t corrupted parties, there is already one honest party included in $\mathcal{H}_i^{(k)} \cup \mathcal{C}_i$. By Lemma 5.4, the honest party cannot belong to $\mathcal{C}_i$. So it must be in $\mathcal{H}_i^{(k)}$. This contradicts our assumption that P_i is the first honest party to move to the kth happy set of every honest party.

We conclude that if one honest party moves to the kth happy set of the honest parties before $(k+t)$th round, then all the honest parties move to the kth happy sets by the end of $(k+t)$th round. Otherwise, none of the honest parties enter the kth happy sets of the honest parties. □

Lemma 5.10. *Assume that* Hash *is a collision-resistant hash function. For any k, all the honest parties in $\mathcal{H}_i^{(k)}$ for an honest P_i hold the same value.*

Proof. This follows from the fact that Hash is a collision-resistant hash function and every honest party has an identical copy of $h^{(k)}$ (since it is broadcasted). Even a corrupted party cannot find a different image of $h^{(k)}$. Therefore, even if an honest party receives the kth message block from a corrupted party, the message block must be the same as the kth message block possessed by the honest parties in the happy set of an honest party (with very high probability). □

We are now ready to prove the agreement property of (n)-BB protocol.

Lemma 5.11 (Agreement)**.** *In protocol* (n)-BB*, every honest party will output the same message.*

Proof. By Lemma 5.9, either all or none of the honest parties are present in the kth happy sets of all the honest parties for any k. If all of them are present, then by Lemma 5.10 they will hold the same block. If none of them are present, then they will hold $\perp$. Agreement, therefore, holds for every block of message. Agreement over the entire message follows easily. □

Theorem 5.1. *Assume that* Hash *is a collision-resistant hash function. The protocol* (n)-BB *satisfies (except with negligible probability in κ):*

Agreement: Every honest party will output the same message.

Validity: If the sender is honest, all honest parties output the sender's message m.

Complexity: A round complexity and seed-round complexity of $\mathcal{O}(n)$. A communication complexity and a seed-communication complexity of $\mathcal{O}\big(\ell n + (n\kappa + n^3 \log n)\mathcal{B}(1)\big)$ and $\mathcal{O}(n\kappa + n^3 \log n)$ bits respectively.

Proof. The agreement, validity and complexity follow from Lemma 5.11, Lemma 5.2 and Lemma 5.1 respectively. □

6 Conclusion

We studied extension protocols that are optimal in terms of communication and round complexity. We introduce *seed-round complexity* as a measure of the number of rounds in which a single-bit broadcast protocol is invoked, and *seed-communication complexity* as a measure of the number of bits communicated through seed broadcasts. We presented a BA extension protocol in the $t < n/3$ setting that is simultaneously communication and round optimal and requires a single seed round. Our protocol does not require any set-up, is error-free and information-theoretically secure. The existing protocols in this realm achieve only communication optimality and incur $\Omega(n^3)$ round as well as seed-round complexity. Then, in the setting with a set-up assumption, we gave constructions of two extension protocols. Our extension protocol in the $t < n/2$ setting also achieves optimal communication and improves the round complexity to 5 from 6 rounds of the best known extension protocol in this setting. Finally, in the $t < n$ setting, we presented a BB extension protocol optimal in both communication and round complexity. Our $t < n$ extension protocol has a round and seed-round complexity of $\mathcal{O}(n)$ that improves the state-of-the-art in this setting which is only communication optimal. We leave as open questions, designing protocols that are optimal in communication and round complexity while minimizing the seed-round complexity to 1 for the more interesting and challenging $t < n$ setting.

Acknowledgements. The authors would like to thank Ling Ren and Zhuolun Xiang for pointing out a minor bug in the BA protocol with $t < n/2$ presented in the conference version [GP16]. The bug allowed the adversary to make the honest parties communicate more than $\mathcal{O}(n\ell)$ bits, and has been fixed in this version.

References

[BGP92] Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science*, pages 313–321. Springer, 1992.

[BH06] Zuzana Beerliová-Trubíniová and Martin Hirt. Efficient multi-party computation with dispute control. In *Theory of Cryptography*, pages 305–328. Springer, 2006.

[Blu90] Norbert Blum. A new approach to maximum matching in general graphs. In *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, pages 586–597, 1990.

[BOCG93] 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. ACM, 1993.

[BOEY03] Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[BSGH13] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 57–64. ACM, 2013.

[Can96] Ran Canetti. *Studies in secure multiparty computation and applications*. PhD thesis, The Weizmann Institute of Science, 1996.

[CCGZ16] Ran Cohen, Sandro Coretti, Juan Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In *Proceedings, Part III, of the 36th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO 2016-Volume 9816*, pages 240–269. Springer-Verlag New York, Inc., 2016.

[CCGZ17] Ran Cohen, Sandro Coretti, Juan Garay, and Vassilis Zikas. Round-Preserving Parallel Composition of Probabilistic-Termination Cryptographic Protocols. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:15, 2017.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT 1997*, pages 103–118. Springer, 1997.

[CO18] Wutichai Chongchitmate and Rafail Ostrovsky. Information-theoretic broadcast with dishonest majority for long messages. Cryptology ePrint Archive, Report 2018/829, 2018. `https://eprint.iacr.org/2018/829`.

[CW92] Brian A Coan and Jennifer L Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, 1992.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[FH06] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued byzantine agreement. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 163–168. ACM, 2006.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[GGOR13] Juan Garay, Clint Givens, Rafail Ostrovsky, and Pavel Raykov. Broadcast (and round) efficient verifiable secret sharing. In *International Conference on Information Theoretic Security*, pages 200–219. Springer, 2013.

[GJ79] MR Garey and DS Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979.

[GKKO07] Juan Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *Foundations of Computer Science, 2007. FOCS'07*, pages 658–668. IEEE, 2007.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[GP16] Chaya Ganesh and Arpita Patra. Broadcast extensions with optimal communication and round complexity. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 371–380. ACM, 2016.

[HMP00] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. *Advances in CryptologySIACRYPT 2000*, pages 143–161, 2000.

[HR14] Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: The $t < n$ case. In *Advances in Cryptology–ASIACRYPT 2014*, pages 448–465. Springer, 2014.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology-CRYPTO 2003*, pages 145–161. Springer, 2003.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Advances in Cryptology - CRYPTO 2006*, pages 445–462, 2006.

[KK07] Jonathan Katz and Chiu-Yuen Koo. Round-efficient secure computation in point-to-point networks. *Advances in Cryptology-EUROCRYPT 2007*, pages 311–328, 2007.

[KKK08] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of vss in point-to-point networks. In *International Colloquium on Automata, Languages, and Programming*, pages 499–510. Springer, 2008.

[KLST11] Valerie King, Steven Lonargan, Jared Saia, and Amitabh Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. In *Distributed Computing and Networking - 12th International Conference, ICDCN 2011, Bangalore, India, January 2-5, 2011. Proceedings*, pages 203–214, 2011.

[KS09] Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{O}(n^{3/2})$ bits. In *International Symposium on Distributed Computing*, pages 464–478. Springer, 2009.

[KS11] Valerie King and Jared Saia. Breaking the $o(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. *Journal of the ACM (JACM)*, 58(4):18, 2011.

[KSSV06] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 990–999. Society for Industrial and Applied Mathematics, 2006.

[KY86] Anna Karlin and Andrew Chi-Chih Yao. Probabilistic lower bounds for byzantine agreement. Technical report, Manuscript, 1986.

[LF82] Leslie Lamport and Michael Fischer. Byzantine generals and transaction commit protocols. Technical report, Technical Report 62, SRI International, 1982.

[LLR02] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*, pages 203–212, 2002.

[LLR06] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. *Journal of the ACM (JACM)*, 53(6):881–917, 2006.

[LV11] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 11–20. ACM, 2011.

[Lyn96] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[Mic17] Silvio Micali. Very simple and efficient byzantine agreement. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 6:1–6:1, 2017.

[MS78] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.

[PR] Arpita Patra and C Pandu Rangan. Communication optimal multi-valued asynchronous broadcast protocol. *Progress in Cryptology–LATINCRYPT 2010*, page 162.

[PR11] Arpita Patra and C Pandu Rangan. Communication optimal multi-valued asynchronous byzantine agreement with optimal resilience. In *Proceedings of the 5th international conference on Information theoretic security*, pages 206–226. Springer-Verlag, 2011.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[PW96] Birgit Pfitzmann and Michael Waidner. *Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$.* Technical Report RZ 2882, IBM Research, 1996.

[RS60] 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.

[TC84] Russell Turpin and Brian A Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing(preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 209–213. ACM, 1979.

$(\frac{\mathsf{n}}{2})$-BA

– **Input of every P_i:** An ℓ-bit message m_i.

– **Oracle:** Broadcast oracle for bits.

– **Cryptographic Assumption:** A collision resistant hash function Hash.

Checking Phase. Every party P_i does the following:

1. Compute a hash of the message m_i as $h_i = \mathsf{Hash}(m_i)$ and broadcast h_i.
2. Check if at least $n - t$ broadcasted hashes are equal. If no $n - t$ broadcasted hashes are equal, output $o_i = \perp$ and terminate. Otherwise, let h denote the common hash value broadcasted by at least $n - t$ parties. Then form $\mathcal{P}_{\mathsf{sm}}$ as the set of parties broadcasting h.

Agreement Phase. Every party P_i does the following

1. If $P_i \in \mathcal{P}_{\mathsf{sm}}$, set output message $o_i = m_i$.
2. Form an injective function from $\mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ to $\mathcal{P}_{\mathsf{sm}}$, by say, mapping the party with the smallest index in $\mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ to the party with the smallest index in $\mathcal{P}_{\mathsf{sm}}$ i.e. $\phi : \mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}} \to \mathcal{P}_{\mathsf{sm}}$.
3. If $P_i \in \mathcal{P}_{\mathsf{sm}}$ and $P_i = \phi(P_j)$, then send o_i to P_j.
4. If $P_i \in \mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ and received a value say, o'_j from $P_j \in \mathcal{P}_{\mathsf{sm}}$ in the previous round such that $P_j = \phi(P_i)$, then check if $\mathsf{Hash}(o'_j) = h$. If the test passes, set $\mathtt{happy}_i = 1$ and assign output message $o_i = o'_j$, else set $\mathtt{happy}_i = 0$. Broadcast $\mathtt{happy}_i$.
5. Construct a set $\mathcal{P}_{\mathsf{conflict}}$ consisting of the parties $P_j, \phi(P_j)$ such that $\mathtt{happy}_j$ received from P_j in the previous step is 0 and P_j belongs to $\mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$. Set $\mathcal{P}_{\mathsf{hmsm}} = \mathcal{P} \setminus \mathcal{P}_{\mathsf{conflict}}$, $d = \lceil (|\mathcal{P}_{\mathsf{hmsm}}| + 1)/2 \rceil$.
6. Transform the message o_i into a polynomial over $GF(2^c)$, for $c = \lceil (\ell+1)/d \rceil$ denoted by f_i with degree $d-1$. Compute the c-bit piece $y_i = f_i(i)$, $H_i = (\mathsf{Hash}(f_i(1)), \cdots, \mathsf{Hash}(f_i(n)))$ and sends (y_i, H_i) to P_j for every $P_j \in \mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ that broadcasted $\mathtt{happy}_j = 0$.
7. If $P_i \in \mathcal{P} \setminus \mathcal{P}_{\mathsf{sm}}$ and $\mathtt{happy}_i = 0$, check each piece y_j received from each $P_j \in \mathcal{P}_{\mathsf{hmsm}}$ against the jth entry of every hash value vector H_k received from $P_k \in \mathcal{P}_{\mathsf{hmsm}}$. If at least d of the hash values match a piece y_j, then accept y_j, otherwise reject it. Interpolate the polynomial f from the d accepted pieces y_j, and compute the message m corresponding to the polynomial f. Set $o_i = m$.
8. Output o_i and terminate.

Figure 3: Honest Majority BA

(n)-BB

– **Input of P_s:** An ℓ bit message m.

– **Oracle:** Broadcast oracle for bits.

– **Cryptographic Assumption:** A collision resistant hash function Hash.

Hash Agreement Phase: The sender P_s does the following:

1. Break the message m into n pieces by padding the end of the message if necessary so that all pieces are of the same length. Denote the n messages by $m^{(1)}, \cdots, m^{(n)}$.
2. For $k = 1, \ldots, n$, compute $h^{(k)} = \mathsf{Hash}(m^{(k)})$ and broadcast $h^{(k)}$ to all parties.

Block Agreement Phase: Each party P_i does the following:

1. Initialize
   - $\mathcal{C}_i$ to $\emptyset$, c_i to 1 and r to 1.
   - $\mathbf{T}_i^{(k)}[j, l] = 1$ for $j, l, k \in \{1, \ldots, n\}$
   - $\mathcal{H}_i^{(k)}, \overline{\mathcal{H}}_i^{(k)}$ to P_s and $\mathcal{P} \setminus P_s$ respectively for $k \in \{1, \ldots, n\}$
   - $o_i^{(k)}$ to $\perp$ if $P_i \neq P_s$, $o_i^{(k)} = m_i^{(k)}$ otherwise for $k \in \{1, \ldots, n\}$
2. While $r \leq (n + t)$
   (a) If $P_i \in \overline{\mathcal{H}}_i^{(\mathsf{c}_i)}$, $\exists\, P_j \in \mathcal{H}_i^{(\mathsf{c}_i)} \setminus \mathcal{C}_i$ and $|\mathcal{H}_i^{(\mathsf{c}_i)} \cup \mathcal{C}_i| \geq r - \mathsf{c}_i + 1$, then broadcast $(\mathtt{send}, j, \mathsf{c}_i)$.
   (b) Let $(\mathtt{send}, x, y)$ be the output of the broadcast initiated by $P_j \notin \mathcal{C}_i$.
      i. if $\mathbf{T}_i^{(y)}[x, j] = 1$ and there is only one broadcast initiated by P_j, then set $\mathbf{T}_i^{(y)}[x, j] = 0$. If $x = i$ and $P_i \in \mathcal{H}_i^{(y)}$, then send $o_i^{(y)}$ to P_j over point-to-point channel.
      ii. else add P_j to $\mathcal{C}_i$.
   (c) If P_i broadcasted $(\mathtt{send}, j, \mathsf{c}_i)$ in step 2a: let $\overline{o_j^{(\mathsf{c}_i)}}$ denote the message block received from P_j over point-to-point channel.
      i. if $h^{(\mathsf{c}_i)} = \mathsf{Hash}(\overline{o_j^{(\mathsf{c}_i)}})$, then increment c_i by one, set $o_i^{(\mathsf{c}_i)} = \overline{o_j^{(\mathsf{c}_i)}}$ and finally broadcast $(\mathtt{happy}, \mathcal{H}_i^{(\mathsf{c}_i)}, \mathcal{C}_i, \mathsf{c}_i)$.
      ii. otherwise broadcast $(\mathtt{unhappy}, \mathsf{c}_i)$, add P_j to $\mathcal{C}_i$.
   (d) Let v be the output of the broadcast done by $P_j \notin \mathcal{C}_i$ in step 2c who broadcasted $(\mathtt{send}, *, x)$ in step 2a earlier this round:
      i. if $v = (\mathtt{happy}, \mathcal{H}_j^{(x)}, \mathcal{C}_j, x)$, $\mathcal{H}_j^{(x)} \cup \mathcal{C}_j \subseteq \mathcal{H}_i^{(x)} \cup \mathcal{C}_i$ and $|\mathcal{H}_j^{(x)} \cup \mathcal{C}_j| \geq r - x + 1$, then move P_j from $\overline{\mathcal{H}}_i^{(x)}$ to $\mathcal{H}_i^{(x)}$ and set $\mathcal{H}_i^{(x)} = \mathcal{H}_i^{(x)} \cup \mathcal{H}_j^{(x)}$.
      ii. if $v = (\mathtt{unhappy}, x)$, then do nothing.
      iii. else add P_j to $\mathcal{C}_i$.
   (e) If $r = \mathsf{c}_i + t$ and $P_i \in \overline{\mathcal{H}}_i^{(\mathsf{c}_i)}$, then exit *while* loop.
   (f) Set $r = r + 1$.
3. If $o_i^{(k)} \neq \perp$ for all $1 \leq k \leq n$, then output $o_i^{(1)} | \cdots | o_i^{(n)}$. Else, output $\perp$.

Figure 4: Broadcast Protocol in Dishonest Majority Setting