Optimal Error-Free Multi-Valued Byzantine Agreement

Jinyuan Chen
Louisiana Tech University, Ruston, LA, USA

Abstract

Byzantine agreement (BA) is a distributed consensus problem where n processors want to reach agreement on an ℓ-bit message or value, but up to t of the processors are dishonest or faulty. The challenge of this BA problem lies in achieving agreement despite the presence of dishonest processors who may arbitrarily deviate from the designed protocol. In this work by using coding theory, together with graph theory and linear algebra, we design a **co**ded BA protoc**ol** (termed as COOL) that achieves consensus on an ℓ-bit message with optimal *resilience*, asymptotically optimal *round complexity*, and asymptotically optimal *communication complexity* when $\ell \geq t \log t$, simultaneously. The proposed COOL is a *deterministic* BA protocol that is guaranteed to be correct in all executions (*error free*) and does not rely on cryptographic technique such as signatures, hashing, authentication and secret sharing (*signature free*). It is secure against computationally unbounded adversary who takes full control over the dishonest processors (*information-theoretic secure*). The main idea of the proposed COOL is to use a carefully-crafted error correction code that provides an efficient way of exchanging "compressed" information among distributed nodes, while keeping the ability of detecting errors, masking errors, and making a consistent and validated agreement at honest distributed nodes. We show that our results can also be extended to the setting of Byzantine broadcast, aka Byzantine generals problem, where the honest processors want to agree on the message sent by a leader who is potentially dishonest. The results reveal that *coding* is an effective approach for achieving the fundamental limits of Byzantine agreement and its variants. Our protocol analysis borrows tools from coding theory, graph theory and linear algebra.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Byzantine agreement, information-theoretic security, error correction codes

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.17

Related Version *Full Version*: `https://arxiv.org/pdf/2009.10965.pdf`

Funding *Jinyuan Chen*: This work was supported in part by the NSF under Grant OIA-1946231 and in part by the Louisiana Board of Regents under Grant 32-4121-40336.

1 Introduction

Byzantine agreement (BA), as originally proposed by Pease, Shostak and Lamport in 1980, is a distributed consensus problem where n processors want to reach agreement on some message (or value), but up to t of the processors are dishonest or faulty [52]. The challenge of this BA problem lies in achieving agreement despite the presence of dishonest processors who may arbitrarily deviate from the designed protocol. One variant of the problem is Byzantine broadcast (BB), aka Byzantine generals problem, where the honest processors want to agree on the message sent by a leader who is potentially dishonest [39]. Byzantine agreement and its variants are considered to be the fundamental building blocks for distributed systems and cryptography including Byzantine-fault-tolerant (BFT) distributed computing, distributed storage, blockchain protocols, state machine replication and voting, just to name a few [52, 39, 26, 45, 23, 41, 25, 42, 48, 50, 9, 43, 1, 13, 57, 49, 53, 27, 58].

Table 1 Comparison of the proposed and some other error-free synchronous BA protocols.

Protocols	Resilience	Communication	Round	Error Free	Signature Free
ℓ-1-bit	$n \geq 3t+1$	$\Omega(n^2\ell)$	$\Omega(\ell t)$	yes	yes
[41]	$n \geq 3t+1$	$O(n\ell + n^4\sqrt{\ell} + n^6)$	$\Omega(\sqrt{\ell} + n^2)$	yes	yes
[25]	$n \geq 3t+1$	$O(n\ell + n^4)$	$O(t)$	yes	yes
[42]	$n \geq 3t+1$	$O(n\ell + n^4)$	$O(t)$	yes	yes
[48]	$n \geq 3t+1$	$O(n\ell + n^3)$	$O(t)$	yes	yes
Proposed	$n \geq 3t+1$	$O(\max\{n\ell, nt\log t\})$	$O(t)$	yes	yes

To solve the Byzantine agreement problem, a designed protocol needs to satisfy the following conditions: every honest processor eventually outputs a message and terminates (*termination*); all honest processors output the same message (*consistency*); and if all honest processors hold the same initial message then they output this initial message (*validity*). A protocol that satisfies the above three conditions in all executions is said to be *error free*. The quality of a BA protocol is measured primarily by using three parameters:

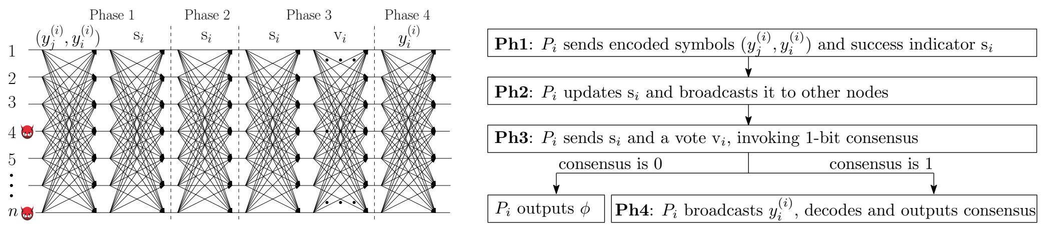
- Resilience: the number of processors n as a function of t allowed.
- Round complexity: the number of rounds of exchanging information, denoted by r.
- Communication complexity: the total number of communication bits, denoted by b.

For any error-free BA protocol, the known lower bounds on those parameters are respectively

$$n \geq 3t+1 \quad \text{(cf. [52, 39])}, \quad r \geq t+1 \quad \text{(cf. [22, 20])}, \quad b \geq \Omega(\max\{n\ell, nt\}) \text{ (cf. [19, 23])}$$

where ℓ denotes the length of message. It is worth mentioning that, in practice the consensus is often required for *multi-valued* message rather than just single-bit message [23, 41, 25, 42, 48, 50, 51]. For example, in BFT consensus protocols of Libra (or Diem) and Hyperledger Fabric proposed by Facebook and IBM respectively, the message being agreed upon could be a transaction or transaction block with size scaled from $1KB$ to $1MB$ [4, 59, 28, 33, 60, 5, 3]. Also, in practice the consensus is often expected to be 100% secure and error free in mission-critical applications such as online banking and smart contracts [8, 30, 44, 12, 18].

The multi-valued BA problem of achieving consensus on an ℓ-bit message could be solved by invoking ℓ instances of 1-bit consensus in sequence, which is termed as ℓ-1-bit scheme. However, this scheme will result in communication complexity of $\Omega(n^2\ell)$ bits, because $\Omega(n^2)$ is the lower bound on communication complexity of 1-bit consensus given $n \geq 3t+1$ [16, 7, 19]. In 2006, Fitzi and Hirt provided a *probabilistically correct* multi-valued BA protocol by using a hashing technique, which results in communication complexity of $O(n\ell + n^3(n+\kappa))$ bits for some constant κ [23]. In 2011, Liang and Vaidya provided an *error-free* BA protocol with communication complexity $O(n\ell + n^4\sqrt{\ell} + n^6)$ bits, which is optimal when $\ell \geq n^6$ [41]. However, in the regime of $\ell < n^6$, this communication complexity is sub-optimal. The result of [41] was improved recently in [25], [42] and [48]. Specifically, the communication complexities of the BA protocols proposed in [25], [42] and [48] are $O(n\ell+n^4)$ bits, $O(n\ell + n^4)$ bits, and $O(n\ell + n^3)$ bits, respectively. Although the communication complexity has been improved in [25], [42] and [48], the achievable performance is still sub-optimal in the regime of $\ell < n^2$. In some previous work, randomized algorithms were proposed to reduce communication and round complexities but the termination cannot be 100% guaranteed [21, 50, 2, 11, 15, 35, 46, 47]. In some other work, the protocols were designed with cryptographic technique such as signatures, hashing, authentication and secret sharing [54, 35, 1, 20]. However, the protocols with such cryptographic technique are vulnerable to attacks from the adversary with very high computation power, e.g., using



Phase 1
Phase 2
Phase 3
Phase 4
$(y_j^{(i)}, y_i^{(i)})$
s_i
s_i
s_i
v_i
$y_i^{(i)}$
1
2
3
4
5
n
Ph1: P_i sends encoded symbols $(y_j^{(i)}, y_i^{(i)})$ and success indicator s_i
Ph2: P_i updates s_i and broadcasts it to other nodes
Ph3: P_i sends s_i and a vote v_i, invoking 1-bit consensus
consensus is 0
consensus is 1
P_i outputs ϕ
Ph4: P_i broadcasts $y_i^{(i)}$, decodes and outputs consensus

Figure 1 Four-phase operation and block diagram at Processor i (P_i) of the proposed COOL.

supercomputer or quantum computer possibly available in the future, and hence not error free. A protocol that doesn't rely on cryptographic technique mentioned above is said to be *signature free*. A protocol is said to be *information-theoretic secure* if it is secure against computationally unbounded adversary who takes full control over the dishonest processors.

In this work we focus on the fundamental limits of error-free multi-valued Byzantine agreement. Specifically by using coding theory, together with graph theory and linear algebra, we are able to design an error-free signature-free information-theoretic-secure multi-valued BA protocol (named as COOL) with optimal *resilience*, asymptotically optimal *round complexity*, and asymptotically optimal *communication complexity* when $\ell \geq t \log t$, simultaneously (see Table 1), focusing on the BA setting with synchronous communication network. In a nutshell, carefully-crafted error correction codes provide an efficient way of exchanging "compressed" information among distributed processors, while keeping the ability of detecting errors, masking errors, and making a consistent agreement at honest distributed processors.

The main difference between the protocols of [41, 25, 42, 48] and our proposed protocol is that, while coding is also used, in the protocols of [41, 25, 42, 48] each distributed processor needs to send an n-bit information to all other processors after generating a graph based on the exchanged information, which results in a communication complexity for this step of at least $\Omega(n^3)$ bits. This is a limitation in the protocols of [41, 25, 42, 48]. Our proposed protocol, which is carefully designed by using coding theory, graph theory and linear algebra, avoids the limitation appeared in [41, 25, 42, 48]. Specifically, our proposed protocol consists of at most four phases (see Fig. 1 and Section 4). After the third phase, it is guaranteed that at most one group of honest processors output the same non-empty value and the size of this group is bigger than the group size of dishonest processors. In this way honest processors can calibrate their values based on majority rule and error-correction decoding (see Section 4). The *high-level* insights and used tools are described below.

Coding theory: Error correction code is used here to reduce the communication complexity in a way that the distributed processors exchange the encoded ("compressed") information symbols but not the initial ℓ-bit messages, where the encoded information symbol is a projection of a message on an encoding vector, e.g., $\boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_1$ is a projection of a message $\bar{\boldsymbol{w}}_1$ on an encoding vector $\boldsymbol{h}_i$ (see Fig. 2-(a)). The challenge is that the encoded symbols could not reveal enough information about the original messages, which might lead to some illusions at distributed processors and result in an inconsistent consensus. For example, as shown in Fig. 2-(a), when Processor i having an initial message as $\bar{\boldsymbol{w}}_1$ sends out an encoded symbol $\boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_1$, this encoded symbol might be expressed as $\boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_1 = \boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_2$ if $\bar{\boldsymbol{w}}_1 - \bar{\boldsymbol{w}}_2$ is in the null space of $\boldsymbol{h}_i$. In this case, it creates an illusion at other processors that Processor i might initially hold the message as $\bar{\boldsymbol{w}}_2$ but not $\bar{\boldsymbol{w}}_1$. Our goal is to control the number of the aforementioned illusions to be small in the designed protocol and finally remove those illusions, by using graph theory and linear algebra.

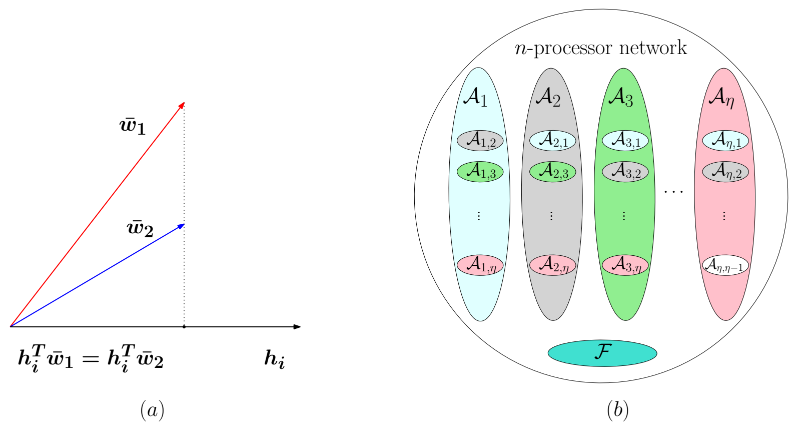

$\bar{\boldsymbol{w}}_1$
$\bar{\boldsymbol{w}}_2$
$\boldsymbol{h}_i^T\bar{\boldsymbol{w}}_1 = \boldsymbol{h}_i^T\bar{\boldsymbol{w}}_2$
$\boldsymbol{h}_i$
(a)
n-processor network
$\mathcal{A}_1$ $\mathcal{A}_2$ $\mathcal{A}_3$ $\mathcal{A}_\eta$
$\mathcal{A}_{1,2}$ $\mathcal{A}_{1,3}$ $\mathcal{A}_{1,\eta}$
$\mathcal{A}_{2,1}$ $\mathcal{A}_{2,3}$ $\mathcal{A}_{2,\eta}$
$\mathcal{A}_{3,1}$ $\mathcal{A}_{3,2}$ $\mathcal{A}_{3,\eta}$
$\mathcal{A}_{\eta,1}$ $\mathcal{A}_{\eta,2}$ $\mathcal{A}_{\eta,\eta-1}$
$\mathcal{F}$
(b)

Figure 2 (a) Two vectors $\bar{\boldsymbol{w}}_1$ and $\bar{\boldsymbol{w}}_2$ have the same projection on an encoding vector $\boldsymbol{h}_i$. (b) An illustration of n-processor network. Group $\mathcal{F}$ denotes the indices of all dishonest processors. Group $\mathcal{A}_l$ denotes the indices of honest processors whose initial messages are all equal to a value $\bar{\boldsymbol{w}}_l$, for some $\bar{\boldsymbol{w}}_l$ and for $l \in [1:\eta]$. $\mathcal{A}_{l,j}$ denotes a subset of $\mathcal{A}_l$ such that $\boldsymbol{h}_i^\intercal\bar{\boldsymbol{w}}_l = \boldsymbol{h}_i^\intercal\bar{\boldsymbol{w}}_j, \forall i \in \mathcal{A}_{l,j}$, which means that the encoded information symbols sent from the processors in $\mathcal{A}_{l,j}$ seem to be projected from the value $\bar{\boldsymbol{w}}_j$ but actually projected from the value $\bar{\boldsymbol{w}}_l$, for $j \neq l, j, l \in [1:\eta]$.

Graph theory and linear algebra: To control the number of the aforementioned illusions to be small, we first classify the n-processor network into different groups. As shown in Fig. 2-(b), Group $\mathcal{A}_l$ denotes a set of honest processors holding the same value of initial messages, i.e., $\bar{\boldsymbol{w}}_l$, for $l \in [1:\eta]$, while $\mathcal{A}_{l,j}$ denotes a subset of $\mathcal{A}_l$ such that the encoded symbols sent from $\mathcal{A}_{l,j}$ seem to be projected from the value $\bar{\boldsymbol{w}}_j$, but actually projected from the value $\bar{\boldsymbol{w}}_l$. Therefore, the encoded information symbols sent from $\cup_{l=1,l\neq j}^{\eta}\mathcal{A}_{l,j}$ seem to be projected from the value $\bar{\boldsymbol{w}}_j$, but actually not. The idea of our design is to control the size of $\cup_{l=1,l\neq j}^{\eta}\mathcal{A}_{l,j}$ to be small for any j. Specifically, in our protocol the parameter η is controlled to be small by using graph theory, while the size of each $\mathcal{A}_{l,j}$ is controlled to be bounded by using linear algebra. For the use of graph theory, our approach is to map the n-node network into a specific type of graph, with one graph example depicted in Fig. 3, and bound the value of η based on this defined graph such that η cannot be more than 2 at the end of the second phase (see Lemma 22 and Lemma 18). For the use of linear algebra, our approach is to construct a set of linear equations based on the constraints related to $\mathcal{A}_{l,j}$, as well as the encoding matrix property, and then bound the size of $\mathcal{A}_{l,j}$ (see Lemma 21).

Coding has been used previously as an exciting approach in network communication and cooperative data exchange for improving throughput and tolerating attacks or failures [40, 37, 24, 10, 31, 34, 36, 38, 63, 32, 29, 62, 61, 17]. However, one common assumption in those previous works is that the source of the data is always *fault-free*, i.e., the source node is always honest or trustworthy, or the initial messages of the honest nodes are consistent and generated from the trustworthy source node. This is very different from the BA and BB problems considered here, in which the leader (or the source node) can be dishonest and the original messages of the distributed nodes can be controlled by the Byzantine adversary. One of the main difficulties of the BA and BB problems lies in the unknown knowledge about the leader (honest or dishonest) and about initial messages (consistent or inconsistent).

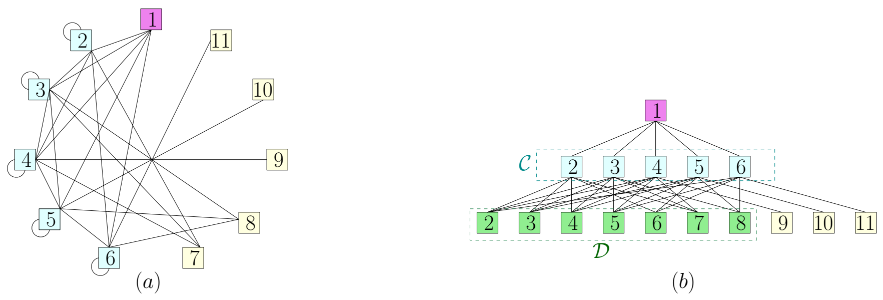


Figure 3 (a) One example of a type of graph, where each vertex in $\mathcal{C} = \{2, 3, 4, 5, 6\}$ is connected with at least 6 edges: one of the edges is connected to vertex $i^\star = 1$ and the rest are connected to the vertices in $\mathcal{D} = \{2, 3, 4, 5, 6, 7, 8\}$. (b) A three-layer representation of the graph in (a).

2 System models

In the BA problem, n processors want to reach agreement on an ℓ-bit message (or value), but up to t of the processors are dishonest (or faulty). Processor i holds an ℓ-bit initial message $\boldsymbol{w}_i$, $\forall i \in [1:n]$. To solve this BA problem, a designed protocol needs to satisfy the *termination*, *consistency* and *validity* conditions mentioned in the previous section. We consider the synchronous BA, in which every two processors are connected via a reliable and private communication channel, and the messages sent on a channel are guaranteed to reach to the destination on time. We assume that a Byzantine adversary takes full control over the dishonest processors and has complete knowledge of the state of the other processors, including the ℓ-bit initial messages.

As mentioned, a protocol that satisfies the termination, consistency and validity conditions in all executions is said to be *error free*. A protocol that doesn't rely on the cryptographic technique such as signatures, hashing, authentication and secret sharing is said to be *signature free*. A protocol that is secure (satisfying the termination, consistency and validity conditions) against computationally unbounded adversary is said to be *information-theoretic secure*. In the BB problem, the validity condition requires that, if the leader is honest then all honest processors should agree on the message sent by the leader. Other definitions follow similarly from that of the BA problem.

3 Main results

The main results of this work are summarized in the following theorems.

▶ **Theorem 1** (BA problem). *The proposed COOL is an error-free signature-free information-theoretic-secure multi-valued BA protocol that achieves the consensus on an ℓ-bit message with optimal resilience, asymptotically optimal round complexity, and asymptotically optimal communication complexity when $\ell \geq t \log t$, simultaneously.*

Proof. The description of the proposed COOL is provided in Section 4. The proposed COOL achieves the consensus on an ℓ-bit message with resilience of $n \geq 3t + 1$ (optimal), round complexity of $O(t)$ rounds (asymptotically optimal), and communication complexity of $O(\max\{n\ell, nt \log t\})$ bits (asymptotically optimal when $\ell \geq t \log t$), simultaneously. Note that, for any error-free BA protocol, the known lower bounds on resilience, round complexity and communication complexity are $3t+1$ (cf. [52, 39]), $t+1$ (cf. [22, 20]), and $\Omega(\max\{n\ell, nt\})$ (cf. [19, 23]), respectively. ◀

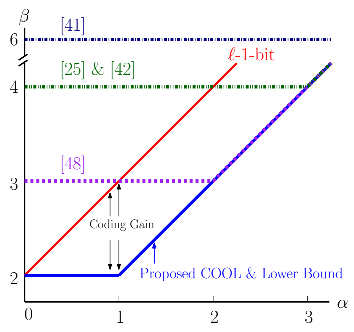


Figure 4 Communication complexity exponent β vs. message size exponent α of the proposed COOL, the protocols in [41, 25, 42, 48], and ℓ-1-bit scheme, focusing on the case with $\delta = 1$.

▶ **Theorem 2** (BB problem). *The proposed adapted COOL is an error-free signature-free information-theoretic-secure multi-valued BB protocol that achieves the consensus on an ℓ-bit message with optimal resilience, asymptotically optimal round complexity, and asymptotically optimal communication complexity when $\ell \geq t \log t$, simultaneously.*

Proof. The proposed COOL designed for the BA setting can be adapted into the BB setting, which achieves the consensus on an ℓ-bit message with the same performance of resilience, round complexity and communication complexity as in the BA setting. Note that the known lower bounds on resilience, round complexity and communication complexity for error-free BA protocols, can also be applied to any error-free BB protocols. The description of adapted COOL for the BB setting is provided in Appendix A. ◀

For an error-free BA (or BB) protocol, we define the notions of *communication complexity exponent*, *message size exponent* and *faulty size exponent* as $\beta(\alpha, \delta) \triangleq \lim_{n\to\infty} \frac{\log b(n,\delta,\alpha)}{\log n}$, $\alpha \triangleq \lim_{n\to\infty} \frac{\log \ell}{\log n}$, and $\delta \triangleq \lim_{n\to\infty} \frac{\log t}{\log n}$, respectively. Intuitively, β (resp. α and δ) captures the exponent of communication complexity b (resp. message size ℓ and faulty size t) with n as the base, when n is large. In this work we characterize the *optimal* communication complexity exponent $\beta^*(\alpha, \delta)$ achievable by *any* error-free BA protocol when $n \geq 3t + 1$.

▶ **Theorem 3** (communication complexity exponent). *The optimal communication complexity exponent $\beta^*(\alpha, \delta)$ achievable by* any *error-free BA (or BB) protocol when $n \geq 3t + 1$, is*

$$\beta^*(\alpha, \delta) = \max\{1 + \alpha, 1 + \delta\}. \tag{1}$$

Proof. Based on the known lower bound, the optimal communication complexity exponent is lower bounded by $\beta^*(\alpha, \delta) \geq \lim_{n\to\infty} \frac{\log \Omega(\max\{n\ell, nt\})}{\log n} = \max\{1+\alpha, 1+\delta\}$. This lower bound is achievable by the proposed COOL, as the communication complexity exponent of COOL, denoted by $\beta^{[cool]}$, is $\beta^{[cool]}(\alpha, \delta) = \lim_{n\to\infty} \frac{\log O(\max\{n\ell, nt \log t\})}{\log n} = \max\{1 + \alpha, 1 + \delta\}$. ◀

As shown in Fig. 4, the proposed COOL achieves the *optimal* communication complexity exponent. Compared to the protocols in [41, 25, 42, 48], COOL provides additive gains up to 4, 2, 2, 1, respectively, in terms of communication complexity exponent. Compared to ℓ-1-bit scheme without coding, COOL provides a communication complexity exponent gain of 1 for any $\alpha \geq 1$, which can be considered as a coding gain resulted from carefully-crafted coding.

4 COOL: coded Byzantine agreement protocol

This section describes the proposed COOL: **co**ded Byzantine agreement protoc**ol**. Error correction code is used in COOL. The (n,k) Reed-Solomon error correction code encodes k data symbols from Galois Field $GF(2^c)$ into a codeword consisting of n symbols from $GF(2^c)$, for $n \leq 2^c - 1$ (cf. [55])[1]. We can use c bits to represent each symbol from $GF(2^c)$, which implies that a vector consisting of k symbols from $GF(2^c)$ can be represented using kc bits of data. The error correction code can be constructed by *Lagrange polynomial interpolation*. An (n,k) error correction code can correct up to $\lfloor \frac{n-k}{2} \rfloor$ errors by applying some efficient decoding algorithms for Reed-Solomon code, such as, Berlekamp-Welch algorithm and Euclid's algorithm[56, 6, 55].

In the proposed COOL, at first the parameters k and c are designed as

$$k \triangleq \left\lfloor \frac{t}{5} \right\rfloor + 1, \quad c \triangleq \left\lceil \frac{\max\{\ell,\ (t/5+1)\cdot \log(n+1)\}}{k} \right\rceil. \tag{2}$$

As will be shown later, our design of k and c as above is one of the key elements in COOL, which guarantees that the proposed COOL satisfies the termination, consistency and validity conditions. With the above values of k and c, it holds true that the condition of the Reed-Solomon error correction code, i.e., $n \leq 2^c - 1$, is satisfied. The (n,k) Reed-Solomon error correction code is used to encode the ℓ-bit initial message $\boldsymbol{w}_i$, $\forall i \in [1:n]$. When ℓ is less than kc bits, the ℓ-bit message $\boldsymbol{w}_i$ will be first extended to a kc-bit data by adding $(kc-\ell)$ bits of redundant zeros (zero padding). The proposed COOL will work as long as $t \leq \frac{n-1}{3}$. In the description of the proposed COOL, t is considered such that $t \leq \frac{n-1}{3}$ and $t = \Omega(n)$. Later on we will discuss the case when t is relatively small compared to n.

The proposed COOL consists of at most four phases (see Fig. 1), which are described in the following sub-sections. The proposed COOL is also described in Algorithm 1 later. Let us first define $\boldsymbol{w}^{(i)}$ as the updated message at Processor i, $i \in [1:n]$. $\boldsymbol{w}^{(i)}$ can be updated via decoding, or via comparing its own information and the obtained information. In this proposed protocol, the decoding is required at Phase 4 only. The value of $\boldsymbol{w}^{(i)}$ is initially set as $\boldsymbol{w}^{(i)} = \boldsymbol{w}_i$, $i \in [1:n]$.

4.1 Phase 1: exchange compressed information and update message

Phase 1 has three steps. The idea is to exchange "compressed" information and learn it.

1) Exchange compressed information: Processor i, $i \in [1:n]$, first encodes its ℓ-bit initial message $\boldsymbol{w}_i$ into ℓ/k-bit symbols as

$$y_j^{(i)} \triangleq \boldsymbol{h}_j^{\intercal} \boldsymbol{w}_i, \quad j \in [1:n] \tag{3}$$

where $\boldsymbol{h}_j$ is defined as $\boldsymbol{h}_j \triangleq [h_{j,1}, h_{j,2}, \cdots, h_{j,k}]^{\intercal}$ and $h_{j,m} \triangleq \prod_{p=1, p\neq m}^{k} \frac{j-p}{m-p}$, $m \in [1:k]$. Then, Processor i sends coded symbols $(y_j^{(i)}, y_i^{(i)})$ to Processor j for $i,j \in [1:n], j \neq i$.

2) Update information: Processor i, $i \in [1:n]$, compares the observation $(y_i^{(j)}, y_j^{(j)})$ received from Processor j with its observation $(y_i^{(i)}, y_j^{(i)})$ and sets a binary indicator for the link between Processor i and Processor j, denoted by $\mathrm{u}_i(j)$, as

$$\mathrm{u}_i(j) = \begin{cases} 1 & \text{if } \ (y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)}) \\ 0 & \text{else} \end{cases} \tag{4}$$

[1] For extended Reed-Solomon codes, the constraint can be relaxed to $n \leq 2^c + 2$ in some cases.

for $j \in [1:n]$. $\mathrm{u}_i(j)$ can be considered as a *link indicator* for Processor i and Processor j. The value of $\mathrm{u}_i(j) = 0$ reveals that Processor i and Processor j have mismatched messages, i.e., $\boldsymbol{w}^{(i)} \neq \boldsymbol{w}^{(j)}$. However, the value of $\mathrm{u}_i(j) = 1$ does *not* mean that Processor i and Processor j have matched messages; it just means that Processor i and Processor j share a common information at a certain degree, i.e., $(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})$. When $\mathrm{u}_i(j) = 0$, the observation of $(y_i^{(j)}, y_j^{(j)})$ received from Processor j is considered as a *mismatched observation* at Processor i. In this step Processor i, $i \in [1:n]$, checks if its own initial message successfully matches the majority of other processors' initial messages, by counting the number of mismatched observations. Specifically, Processor i sets a binary *success indicator*, denoted by s_i, as

$$\mathrm{s}_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} \mathrm{u}_i(j) \geq n - t \\ 0 & \text{else .} \end{cases} \tag{5}$$

The event of $\mathrm{s}_i = 0$ means that the number of mismatched observations is more than t, which implies that the initial message of Processor i doesn't match the majority of other processors' initial messages. If $\mathrm{s}_i = 0$, Processor i updates the message as $\boldsymbol{w}^{(i)} = \phi$ (a default value), else keeps the original value of $\boldsymbol{w}^{(i)}$.

3) Exchange success indicators: Processor i, $i \in [1:n]$, sends the binary value of success indicator s_i to all other processors. Based on the received success indicators $\{\mathrm{s}_i\}_{i=1}^{n}$, each processor creates the following two sets:

$$\mathcal{S}_1 \triangleq \{i : \mathrm{s}_i = 1, i \in [1:n]\}, \quad \mathcal{S}_0 \triangleq \{i : \mathrm{s}_i = 0, i \in [1:n]\}. \tag{6}$$

Note that different processors might have different views on $\mathcal{S}_1$ and $\mathcal{S}_0$, due to the inconsistent information possibly sent from dishonest processors.

▶ Remark 4. *Since $y_j^{(i)}$ defined in (3) has only c bits, the total communication complexity among the network for the first step of Phase 1, denoted by b_1, is $b_1 = 2cn(n-1)$ bits. If Processor i, $i \in [1:n]$, sends the whole message $\boldsymbol{w}_i$ to other processors, then the total communication complexity would be $\ell n(n-1)$ bits. Compared to the whole message $\boldsymbol{w}_i$, the value of $y_j^{(i)}$ can be considered as a compressed information. By exchanging compressed information, instead of whole messages, the communication complexity is significantly reduced in this step. Since the success indicator s_i has only 1 bit, the communication complexity for the third step of Phase 1, denoted by b_2, is $b_2 = n(n-1)$ bits.*

▶ Remark 5. *In Phase 1, exchanging "compressed" information reduces the communication complexity, however, it also creates some potential issues due to the lack of full original information. Fig. 5 describes an example with three disjoint groups in n-processor network: Group $\mathcal{F}$, Group $\mathcal{A}_1$ and Group $\mathcal{A}_2$, where Group $\mathcal{A}_i$ is a set of honest processors holding the same initial message $\bar{\boldsymbol{w}}_i$, given $|\mathcal{A}_1| = t+1$, $|\mathcal{A}_2| = t$, and $\bar{\boldsymbol{w}}_1 \neq \bar{\boldsymbol{w}}_2$. To attack the protocol, in Phase 1 each dishonest processor could send inconsistent information to two different groups of honest processors, that is, sending symbols $(\boldsymbol{h}_j^{\intercal}\bar{\boldsymbol{w}}_1, \boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_1)$ to Processor j for $j \in \mathcal{A}_1$ and sending different symbols $(\boldsymbol{h}_{j'}^{\intercal}\bar{\boldsymbol{w}}_2, \boldsymbol{h}_i^{\intercal}\bar{\boldsymbol{w}}_2)$ to Processor j' for $j' \in \mathcal{A}_2$, respectively, for $i \in \mathcal{F}$. Due to this inconsistent information, together with the condition of $\boldsymbol{h}_1^{\intercal}\bar{\boldsymbol{w}}_1 = \boldsymbol{h}_1^{\intercal}\bar{\boldsymbol{w}}_2$ and $\boldsymbol{h}_{12}^{\intercal}\bar{\boldsymbol{w}}_1 = \boldsymbol{h}_{12}^{\intercal}\bar{\boldsymbol{w}}_2$, the honest processors from different groups consequently have different updated messages, which might lead to inconsistent consensus outputs. In Phase 1 Processor i, $i \in [13:21] \subset \mathcal{A}_2$, sets $\mathrm{u}_i(j) = 0, \forall j \in [1:11]$ and $\mathrm{s}_i = 0$, from which it identifies that its initial message doesn't match the majority of other processors' initial messages and then updates its message as $\boldsymbol{w}^{(i)} = \phi$. However, Processor 12 still thinks that its initial message does match the majority of other processors' initial messages because of the condition*

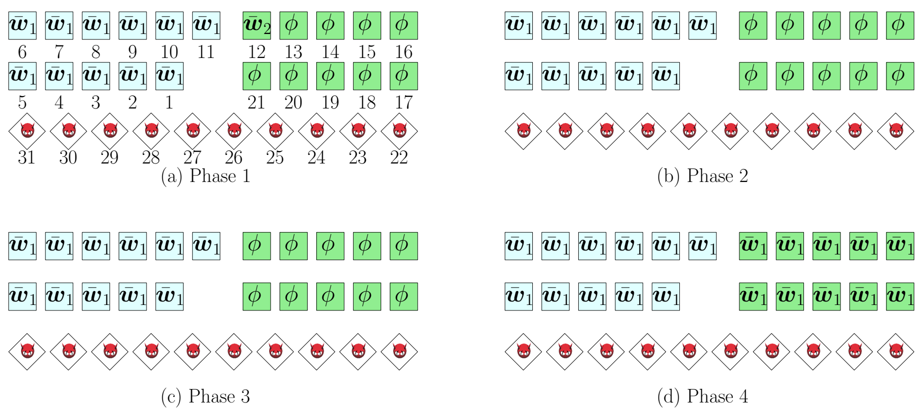


Figure 5 Illustration of COOL for an example with $(t = 10, n = 31)$. The number under the node indicates the identity. The value inside the ith node denotes the value of updated message $\boldsymbol{w}^{(i)}$ at the end of the corresponding phase. Here $\mathcal{A}_1 = [1 : 11]$, $\mathcal{A}_2 = [12 : 21]$, $\mathcal{F} = [22 : 31]$, $\boldsymbol{w}_i = \bar{\boldsymbol{w}}_1, \forall i \in \mathcal{A}_1$ and $\boldsymbol{w}_{i'} = \bar{\boldsymbol{w}}_2, \forall i' \in \mathcal{A}_2$. It is assumed that $\boldsymbol{h}_1^\intercal \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_1^\intercal \bar{\boldsymbol{w}}_2$ and $\boldsymbol{h}_{12}^\intercal \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_{12}^\intercal \bar{\boldsymbol{w}}_2$, for $\bar{\boldsymbol{w}}_1 \neq \bar{\boldsymbol{w}}_2$. All honest processors eventually make the same consensus output.

$\boldsymbol{h}_1^\intercal \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_1^\intercal \bar{\boldsymbol{w}}_2$ *and* $\boldsymbol{h}_{12}^\intercal \bar{\boldsymbol{w}}_1 = \boldsymbol{h}_{12}^\intercal \bar{\boldsymbol{w}}_2$. *This condition implies that* $(y_{12}^{(j)}, y_j^{(j)}) = (y_{12}^{(12)}, y_j^{(12)})$ *for any* $j \in \{1\} \cup \mathcal{A}_2$ *from the view of Processor* 12*, and hence results in a "wrong" (i.e., mismatched) output of* $\mathrm{s}_{12} = 1$ *at Processor* 12*. In the next phases the effort is to detect errors (mismatched information), mask errors, and identify "trusted" information.*

4.2 Phase 2: mask errors, and update success indicator and message

Phase 2 has three steps. The goal is to mask errors from the honest processors.

1) Mask errors identified in the previous phase: Processor i, $i \in \mathcal{S}_1$, sets

$$\mathrm{u}_i(j) = 0, \ \forall j \in \mathcal{S}_0. \tag{7}$$

2) Update and send success indicator: Processor i, $i \in \mathcal{S}_1$, updates s_i as in (5) using updated values of $\{\mathrm{u}_i(1), \cdots, \mathrm{u}_i(n)\}$. If the updated value of success indicator is $\mathrm{s}_i = 0$, then Processor i sends the updated success indicator of $\mathrm{s}_i = 0$ to others and updates the message as $\boldsymbol{w}^{(i)} = \phi$.

3) Update $\mathcal{S}_1$ and $\mathcal{S}_0$: Processor i, $i \in [1 : n]$, updates the sets of $\mathcal{S}_1$ and $\mathcal{S}_0$ as in (6) based on the newly received success indicators $\{\mathrm{s}_i\}_{i=1}^n$.

▶ Remark 6. *Since the success indicator* s_i *has only* 1 *bit, the total communication complexity for the second step of Phase 2, denoted by* b_3*, is bounded by* $b_3 \leq n(n-1)$ *bits.*

▶ Remark 7. *The idea of Phase 2 is to mask errors from honest processors whose initial messages don't match the majority of other processors' initial messages, but could not be detected out in Phase 1. For the example in Fig. 5, at the second step of Phase 2, Processor* 12 *updates the message as* $\boldsymbol{w}^{(12)} = \phi$*. This is because at Step 2 of Phase 2, from the view of Processor* 12*, the number of mismatched observations is at least* 19*, since* $\mathrm{u}_{12}(j) = 0, \forall j \in [2 : 11] \cup [13 : 21]$ *based on the updated information in* (7)*.*

4.3 Phase 3: mask errors, update information, and vote

Phase 3 has five steps. The goal is to mask the rest of errors from the honest processors, and then vote for going to next phase or stopping in this phase.

1) Mask errors identified in the previous phase: Processor i, $i \in \mathcal{S}_1$, sets

$$\mathrm{u}_i(j) = 0, \ \forall j \in \mathcal{S}_0. \tag{8}$$

2) Update and send success indicator: Processor i, $i \in \mathcal{S}_1$, updates s_i as in (5) using updated values of $\{\mathrm{u}_i(1), \cdots, \mathrm{u}_i(n)\}$. If the updated value of success indicator is $\mathrm{s}_i = 0$, then Processor i sends $\mathrm{s}_i = 0$ to others and updates the message as $\boldsymbol{w}^{(i)} = \phi$.

3) Update $\mathcal{S}_1$ and $\mathcal{S}_0$: Processor i, $i \in [1:n]$, updates the sets of $\mathcal{S}_1$ and $\mathcal{S}_0$ as in (6) based on the newly received success indicators $\{\mathrm{s}_i\}_{i=1}^n$.

4) Vote: Processor i, $i \in [1:n]$, sets a binary vote as

$$\mathrm{v}_i = \begin{cases} 1 & \text{if } \sum_{j=1}^n \mathrm{s}_j \geq 2t+1 \\ 0 & \text{else .} \end{cases} \tag{9}$$

The indicator v_i can be considered as a vote for going to next phase or stopping in this phase.

5) One-bit consensus on the n votes: In this step the system runs one-bit consensus [7, 16] on the n votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ from all processors. If the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 1, then every honest processor *goes to next phase*, else every honest processor sets $\boldsymbol{w}^{(i)} = \phi$ and considers it as a final consensus and *stops here.*

▶ Remark 8. *Since s_i has only 1 bit, the total communication complexity for the second step of Phase 3, denoted by b_4, is bounded by $b_4 \leq n(n-1)$ bits.*

▶ Remark 9. *Since the system runs the one-bit consensus from [7, 16], the total communication complexity for the last step of Phase 3, denoted by b_5, is $b_5 = O(nt)$ bits, while the round complexity of this step is $O(t)$ rounds, which dominates the round complexity of COOL.*

▶ Remark 10. *The goal of the first two steps of Phase 3 is to mask the remaining errors from honest processors. As shown in Lemma 17 later, it is guaranteed that at the end of Phase 3 there exists* at most *1 group of honest processors, where the honest processors within this group have the same* non-empty *updated message (like $\mathcal{A}_1$ in Fig. 5).*

4.4 Phase 4: identify trusted information and make consensus

This phase is taken place only when one-bit consensus of $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 1.

1) Update information with majority rule: Processor i, $i \in \mathcal{S}_0$, updates $y_i^{(i)}$ as

$$y_i^{(i)} \leftarrow \mathrm{Majority}(\{y_i^{(j)} : j \in \mathcal{S}_1\}) \tag{10}$$

where the symbols of $y_i^{(j)}$ were received in Phase 1. Majority(•) is a function that returns the most frequent value in the list, based on majority rule. For example, $\mathrm{Majority}(1,2,2) = 2$.

2) Broadcast updated information: Processor i, $i \in \mathcal{S}_0$, sends the updated value of $y_i^{(i)}$ to Processor j, $\forall j \in \mathcal{S}_0, j \neq i$.

3) Decode the message: Processor i, $i \in \mathcal{S}_0$, decodes its message using the observations $\{y_1^{(1)}, y_2^{(2)}, \cdots, y_n^{(n)}\}$, where $\{y_j^{(j)} : j \in \mathcal{S}_0\}$ were updated and received in this phase and $\{y_j^{(j)} : j \in \mathcal{S}_1\}$ were received in the Phase 1. The value of $\boldsymbol{w}^{(i)}$ is updated as the decoded message at Processor i, $i \in \mathcal{S}_0$. For Processor i, $i \in \mathcal{S}_1$, it keeps the original value of $\boldsymbol{w}^{(i)}$.

4) Stop: Processor i, $i \in [1:n]$, outputs consensus as the message $\boldsymbol{w}^{(i)}$ and stops.

▶ Remark 11. *Since $y_i^{(i)}$ has only c bits, the total communication complexity for the second step of Phase 4, denoted by b_6, is upper bounded by $b_6 \leq cn(n-1)$ bits.*

▶ Remark 12. *The idea of Phase 4 is to identify "trusted" information from the set of honest processors whose initial messages match the majority of other processors' initial messages. In this way, the set of honest processors whose initial messages don't match the majority of other processors' initial messages could calibrate and update their information. For the example in Fig. 5, since the size of $\mathcal{A}_1$ is bigger than the size of $\mathcal{F}$, it guarantees that Group $\mathcal{A}_2$ could calibrate and update their information successfully in the first step of Phase 4, eventually making the same consensus output as Group $\mathcal{A}_1$.*

5 Provable performance of COOL

In this section we analyze the performance of COOL. At first we define some groups of processors in the n-processor network. For the ease of notation, let us use $\mathrm{s}_i^{[1]}$, $\mathrm{s}_i^{[2]}$ and $\mathrm{s}_i^{[3]}$ to denote the values of s_i updated in Phase 1, Phase 2 and Phase 3, respectively. Similarly, let us use $\mathrm{u}_i^{[1]}(j)$, $\mathrm{u}_i^{[2]}(j)$ and $\mathrm{u}_i^{[3]}(j)$ to denote the values of $\mathrm{u}_i(j)$ updated in Phase 1, Phase 2 and Phase 3, respectively. Let us first define Group $\mathcal{F}$ as the indices of all of the dishonest processors. Without loss of generality, *in the analysis* we just focus on the case with t dishonest nodes, i.e., $|\mathcal{F}| = t$ (no matter how the dishonest nodes act). Note that when $|\mathcal{F}| = t'$ for some $t' < t$, this case is indistinguishable from the case with $|\mathcal{F}| = t$ in which $t - t'$ out of t dishonest nodes act normally like honest nodes. Therefore, if a protocol is correct in the extreme case with $|\mathcal{F}| = t$, it is also correct in the case with $|\mathcal{F}| < t$.

Let us then define some groups of honest processors as

$$\mathcal{A}_l \triangleq \{i : \boldsymbol{w}_i = \bar{\boldsymbol{w}}_l,\ i \notin \mathcal{F},\ i \in [1:n]\}, \quad l \in [1:\eta] \tag{11}$$

$$\mathcal{A}_l^{[p]} \triangleq \{i : \mathrm{s}_i^{[p]} = 1, \boldsymbol{w}_i = \bar{\boldsymbol{w}}_l,\ i \notin \mathcal{F},\ i \in [1:n]\}, \quad l \in [1:\eta^{[p]}], \quad p \in \{1,2,3\} \tag{12}$$

$$\mathcal{B}^{[p]} \triangleq \{i : \mathrm{s}_i^{[p]} = 0,\ i \notin \mathcal{F},\ i \in [1:n]\}, \quad p \in \{1,2,3\} \tag{13}$$

for some different non-empty ℓ-bit values $\bar{\boldsymbol{w}}_1, \bar{\boldsymbol{w}}_2, \cdots, \bar{\boldsymbol{w}}_\eta$ and some non-negative integers $\eta, \eta^{[1]}, \eta^{[2]}, \eta^{[3]}$ such that $\eta^{[3]} \leq \eta^{[2]} \leq \eta^{[1]} \leq \eta$. Group $\mathcal{A}_l$ is a subset of honest processors who have the same value of initial messages. $\mathcal{A}_l^{[p]}$ (resp. $\mathcal{B}^{[p]}$) is a subset of honest processors who have the same non-empty (resp. empty) value of updated messages at the end of Phase p, for $p \in \{1,2,3\}$. As shown in Fig. 2-(a), two different messages might have the same projection on an encoding vector. With this motivation, Group $\mathcal{A}_l$ (and Group $\mathcal{A}_l^{[p]}$) can be divided into some possibly overlapping sub-groups defined as

$$\mathcal{A}_{l,j} \triangleq \{i :\ i \in \mathcal{A}_l,\ \boldsymbol{h}_i^\intercal \bar{\boldsymbol{w}}_l = \boldsymbol{h}_i^\intercal \bar{\boldsymbol{w}}_j\}, \quad j \neq l,\ j,l \in [1:\eta] \tag{14}$$

$$\mathcal{A}_{l,l} \triangleq \mathcal{A}_l \setminus \{\cup_{j=1, j\neq l}^{\eta} \mathcal{A}_{l,j}\}, \quad l \in [1:\eta] \tag{15}$$

$$\mathcal{A}_{l,j}^{[p]} \triangleq \{i :\ i \in \mathcal{A}_l^{[p]},\ \boldsymbol{h}_i^\intercal \bar{\boldsymbol{w}}_l = \boldsymbol{h}_i^\intercal \bar{\boldsymbol{w}}_j\}, \quad j \neq l,\ j,l \in [1:\eta^{[p]}], \quad p \in \{1,2,3\} \tag{16}$$

$$\mathcal{A}_{l,l}^{[p]} \triangleq \mathcal{A}_l^{[p]} \setminus \{\cup_{j=1, j\neq l}^{\eta^{[p]}} \mathcal{A}_{l,j}^{[p]}\}, \quad l \in [1:\eta^{[p]}], \quad p \in \{1,2,3\}. \tag{17}$$

The provable performance of COOL is summarized in the following Lemmas 13-16.

▶ **Lemma 13** (termination). *Given $n \geq 3t+1$, all honest processors eventually output messages and terminate in COOL.*

Proof. Given $n \geq 3t+1$, it is guaranteed in COOL that all honest processors eventually terminate together at the last step of Phase 3 or Phase 4. It is also guaranteed that every honest processor eventually outputs a message when it terminates. ◀

▶ **Lemma 14** (validity). *Given $n \geq 3t+1$, if all honest processors have the same initial message, then at the end of COOL all honest processors agree on this initial message.*

Proof. When all honest processors have the same initial message, in the first three phases all honest processors will set their success indicators as ones and keep their updated messages exactly the same as the initial message, no matter what information sent by the dishonest processors. In this scenario, all honest processors will go to Phase 4 and eventually make the consensus outputs exactly the same as the initial message at the last step of Phase 4. ◀

▶ **Lemma 15** (consistency). *Given $n \geq 3t+1$, all honest processors reach the same agreement.*

Proof. We prove this lemma by considering each of the following two cases.

- Case (a): In Phase 3, the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 0.
- Case (b): In Phase 3, the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 1.

Analysis for Case (a): In Phase 3 of COOL, if the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 0, then each honest processor will set the updated message as ϕ and consider ϕ as a final consensus and stop here. In this case, all of the honest processors agree on the same message.

Analysis for Case (b): In Phase 3 of COOL, if the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 1, then all honest processors will go to Phase 4. In this case, at least one of the honest processors votes $\mathrm{v}_i = 1$, for some $i \notin \mathcal{F}$. Otherwise, all of the honest processors vote the same value such that $\mathrm{v}_i = 0$, $\forall i \notin \mathcal{F}$ and the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ should be 0, contradicting the condition of this case. In COOL, the condition of voting $\mathrm{v}_i = 1$ (see (9)) is that Processor i receives no less than $2t+1$ number of ones from n success indicators $\{\mathrm{s}_1^{[3]}, \mathrm{s}_2^{[3]}, \cdots, \mathrm{s}_n^{[3]}\}$, i.e., $\sum_{j=1}^{n} \mathrm{s}_j^{[3]} \geq 2t+1$. Since t dishonest processors might send the success indicators as ones, the above outcome implies that

$$\sum_{j \in \cup_{l=1}^{\eta^{[3]}} \mathcal{A}_l^{[3]}} \mathrm{s}_j^{[3]} \geq t+1. \tag{18}$$

Lemma 17 (see Section 5.1) reveals that at the end of Phase 3 there exists *at most* 1 group of honest processors, where the honest processors in this group have the same non-empty updated message, that is, $\eta^{[3]} \leq 1$. Then, for this Case (b), the conclusions in (18) and Lemma 17 imply that at the end of Phase 3 there exists *exactly* 1 group of *honest* processors with group size bigger than or equal to $t+1$, where the honest processors within this group have the same non-empty updated message. In other words, for this Case (b), we have

$$\eta^{[3]} = 1, \quad |\mathcal{A}_1^{[3]}| \geq t+1, \quad \boldsymbol{w}_i = \bar{\boldsymbol{w}}_1, \quad \forall i \in \mathcal{A}_1^{[3]} \tag{19}$$

by following from (18) and Lemma 17, as well as the definition of $\mathcal{A}_l^{[3]}$.

In Phase 4 Processor i, $i \in \mathcal{S}_0$, updates the value of $y_i^{(i)}$ as $y_i^{(i)} \leftarrow \mathrm{Majority}(\{y_i^{(j)} : j \in \mathcal{S}_1\})$ based on the majority rule, where the symbols of $y_i^{(j)}$ were received in Phase 1. In this step it is true that $\mathcal{A}_1^{[3]} \subseteq \mathcal{S}_1$ and $|\mathcal{A}_1^{[3]}| > |\mathcal{F}|$ (see (19)), which guarantees that Processor i, $i \in \mathcal{S}_0 \setminus \mathcal{F}$, could use the majority rule to update the value of $y_i^{(i)}$ as $y_i^{(i)} \leftarrow \mathrm{Majority}(\{y_i^{(j)} : j \in \mathcal{S}_1\}) = \boldsymbol{h}_i^{\intercal} \bar{\boldsymbol{w}}_1$. At the end of this step, for any honest Processor i, $i \notin \mathcal{F}$, the value of $y_i^{(i)}$ becomes $y_i^{(i)} = \boldsymbol{h}_i^{\intercal} \bar{\boldsymbol{w}}_1$, which is encoded with $\bar{\boldsymbol{w}}_1$. In the second step of Phase 4, Processor i, $i \in \mathcal{S}_0$, sends the updated value of $y_i^{(i)}$ to Processor j, $\forall j \in \mathcal{S}_0, j \neq i$. After this, Processor i, $i \in \mathcal{S}_0$, decodes its message using the updated observations $\{y_1^{(1)}, y_2^{(2)}, \cdots, y_n^{(n)}\}$, where $n-t$ observations of which are guaranteed to be $y_j^{(j)} = \boldsymbol{h}_j^{\intercal} \bar{\boldsymbol{w}}_1, \forall j \notin \mathcal{F}$. Since the number of mismatched observations, i.e., the observations not encoded with the message $\bar{\boldsymbol{w}}_1$, is no more than t, then Processor i, $i \in \mathcal{S}_0 \setminus \mathcal{F}$, decodes its message and outputs $\boldsymbol{w}^{(i)} = \bar{\boldsymbol{w}}_1$. At the same time, Processor i, $i \in \mathcal{S}_1 \setminus \mathcal{F}$, outputs the original value as $\boldsymbol{w}^{(i)} = \bar{\boldsymbol{w}}_1$. Thus, all of the honest processors successfully agree on the same message, i.e., $\boldsymbol{w}^{(i)} = \bar{\boldsymbol{w}}_1$, $\forall i \notin \mathcal{F}$. With this we complete the proof of Lemma 15. ◀

▶ **Lemma 16** (complexity). *When $n \geq 3t+1$, COOL achieves the consensus on an ℓ-bit message with the communication complexity of $O(\max\{n\ell, nt\log t\})$ bits and $O(t)$ rounds.*

Proof. The total communication complexity of COOL, denoted by b, is computed as $b = \sum_{i=1}^{6} b_i = O(cn(n-1)+n^2) = O\Big(\Big\lceil \frac{\max\{\ell,\ (t/5+1)\cdot\log(n+1)\}}{\lfloor\frac{t}{5}\rfloor+1}\Big\rceil \cdot n(n-1)+n^2\Big) = O(\max\{\ell n^2/t, n^2\log n\})$ bits, where $b_1, b_2, \cdots, b_6$ are expressed in Remarks 4, 6, 8, 9 and 11, and the parameters c and k are defined in (2). In the description of the proposed protocol, t is considered such that $t \leq (n-1)/3$ and $t = \Omega(n)$. In this case, the total communication complexity computed as above can be rewritten in the form of $b = O(\max\{\ell n, nt\log t\})$ bits. For the case when t is relatively small, we show that COOL can be slightly modified to achieve the communication complexity as $b = O(\max\{\ell n, nt\log t\})$ bits (see the long version [14] for more details). The round complexity of COOL is dominated by the round complexity of the one-bit consensus in Phase 3, which is $O(t)$ rounds. ◀

From Lemmas 13-15, it reveals that given $n \geq 3t+1$ the termination, validity and consistency conditions are all satisfied in COOL in all executions (*error free*). Note that Lemmas 13-15 hold true without using any assumptions on the cryptographic technique (*signature-free*). Lemmas 13-15 also hold true even when the adversary, who takes full control over the dishonest processors, has unbounded computational power (*information-theoretic-secure*). The results of Lemmas 13-16 serve as the achievability proof of Theorem 1.

5.1 Some lemmas

Below we provide some lemmas that are used in the protocol analysis.

▶ **Lemma 17.** *Given $n \geq 3t+1$, at the end of Phase 3 of COOL there exists* at most *1 group of honest processors, where honest processors in this group have the same non-empty updated message, and honest processors outside this group have the same empty updated message.*

Proof. Based on (12), it is equivalent to prove $\eta^{[3]} \leq 1$. In the first step we prove that $\eta^{[2]} \leq 2$ (see Lemma 18 below). Clearly, it is true $\eta^{[3]} \leq 1$ when $\eta^{[2]} \leq 1$, using the fact that $\eta^{[3]} \leq \eta^{[2]}$ (see (11) and (12)). Then in the second step we prove that $\eta^{[3]} \leq 1$ when $\eta^{[2]} = 2$ (see Lemma 19). Thus, it is concluded that $\eta^{[3]} \leq 1$ for COOL with $n \geq 3t+1$. ◀

▶ **Lemma 18.** *For the proposed COOL with $n \geq 3t+1$, it holds true that $\eta^{[2]} \leq 2$.*

Proof. Proof by contradiction is used in this proof. Let us first assume that the claim in Lemma 18 is false. Specifically let us assume that

$$\eta^{[2]} \geq 3. \tag{20}$$

From Lemma 22 that will be shown later, we have $|\mathcal{A}_l| \geq n - 9t/4, l \in [1:\eta^{[2]}]$. By combining the above $\eta^{[2]}$ bounds together we have

$$\sum_{l=1}^{\eta^{[2]}} |\mathcal{A}_l| \geq \eta^{[2]}(n-9t/4) \geq 3(n-9t/4) = (n-t)+(2n-23t/4) > (n-t) \tag{21}$$

where the second inequality uses the assumption in (20); and the last inequality stems from the derivation that $2n - 23t/4 \geq 6t+2-23t/4 > 0$ by using the condition of $n \geq 3t+1$. One can see that the conclusion in (21) contradicts with the identify of $\sum_{l=1}^{\eta^{[2]}} |\mathcal{A}_l| \leq \sum_{l=1}^{\eta} |\mathcal{A}_l| = n-t$, i.e., the total number of honest processors should be $n-t$. Therefore, the assumption in (20) leads to a contradiction and thus $\eta^{[2]}$ should be bounded by $\eta^{[2]} \leq 2$. ◀

▶ **Lemma 19.** *For COOL with $n \geq 3t+1$, it holds true that $\eta^{[3]} \leq 1$ when $\eta^{[2]} = 2$.*

Proof. Given $\eta^{[2]} = 2$, it is concluded from Lemma 20 (shown below) that $\eta^{[1]} = 2$. Furthermore, given $\eta^{[1]} = 2$, it is concluded from Lemma 23 (shown later) that $\eta^{[3]} \leq 1$. We then conclude that $\eta^{[3]} \leq 1$ when $\eta^{[2]} = 2$. ◀

▶ **Lemma 20.** *For the proposed COOL with $n \geq 3t+1$, it is true that $\eta^{[1]} = 2$ when $\eta^{[2]} = 2$.*

Proof. Proof by contradiction is also used here. Given $\eta^{[2]} = 2$, let us assume that $\eta^{[1]} > 2$. Given $\eta^{[2]} = 2$ and Lemma 22 (shown later), we have $|\mathcal{A}_l| \geq n - 9t/4$, $\forall l \in [1:2]$ and have

$$\sum_{l=3}^{\eta} |\mathcal{A}_l| = n - |\mathcal{F}| - |\mathcal{A}_1| - |\mathcal{A}_2| \leq n - t - 2(n - 9t/4) \leq t/2 - 1. \tag{22}$$

If $\eta^{[1]} > 2$, there exists an $i^\star \in \mathcal{A}_{l^\star}^{[1]} \subseteq \mathcal{A}_{l^\star}$ for $l^\star \geq 3$, such that $\mathrm{s}_{i^\star}^{[1]} = 1$ (see (12)) and that $\sum_{j \in \cup_{l=1}^{\eta} \mathcal{A}_l} \mathrm{u}_{i^\star}^{[1]}(j) \geq n - t - t$ (see (5)). Based on the definition of $\mathcal{A}_{l,j}$ in (14), it is true that $\mathrm{u}_{i^\star}^{[1]}(j) = 0$, $\forall j \in \{\cup_{l=1}^{\eta} \mathcal{A}_l\} \setminus \{\mathcal{A}_{l^\star} \cup \{\cup_{l=1, l\neq l^\star}^{\eta} \mathcal{A}_{l,l^\star}\}\}$. Therefore, the aforementioned inequality resulted from (5) can be rewritten as

$$\sum_{j \in \mathcal{A}_{l^\star} \cup \{\cup_{l=1, l\neq l^\star}^{\eta} \mathcal{A}_{l,l^\star}\}} \mathrm{u}_{i^\star}^{[1]}(j) \geq n - t - t. \tag{23}$$

On the other hand, the size of $\mathcal{A}_{l^\star} \cup \{\cup_{l=1, l\neq l^\star}^{\eta} \mathcal{A}_{l,l^\star}\}$ can be upper bounded by

$$\begin{aligned}
|\mathcal{A}_{l^\star} \cup \{\cup_{l=1, l\neq l^\star}^{\eta} \mathcal{A}_{l,l^\star}\}| &\leq |\mathcal{A}_{1,l^\star}| + |\mathcal{A}_{2,l^\star}| + \sum_{l=3}^{\eta} |\mathcal{A}_l| && (24)\\
&\leq |\mathcal{A}_{1,l^\star}| + |\mathcal{A}_{2,l^\star}| + t/2 - 1 && (25)\\
&\leq (k-1) + (k-1) + t/2 - 1 && (26)\\
&= 2(\lfloor t/5 \rfloor + 1 - 1) + t/2 - 1 && (27)\\
&< n - 2t && (28)
\end{aligned}$$

for $i^\star \in \mathcal{A}_{l^\star}^{[1]} \subseteq \mathcal{A}_{l^\star}$ and $l^\star \geq 3$, where (24) uses the fact that $\mathcal{A}_{l^\star} \cup \{\cup_{l=1, l\neq l^\star}^{\eta} \mathcal{A}_{l,l^\star}\} \subseteq \mathcal{A}_{1,l^\star} \cup \mathcal{A}_{2,l^\star} \cup \{\cup_{l=3}^{\eta} \mathcal{A}_l\}$; (25) is from (22); (26) stems from the result in Lemma 21 shown below; (27) uses the definition of k as in (2); (28) follows from the condition of $n \geq 3t+1$.

One can see that the conclusion in (28) contradicts with (23). Therefore, the assumption of $\eta^{[1]} > 2$ leads to a contradiction and thus $\eta^{[1]}$ should be $\eta^{[1]} = 2$, given $\eta^{[2]} = 2$. ◀

▶ **Lemma 21.** *For $\eta \geq \eta^{[1]} \geq 2$, the following inequalities hold true*

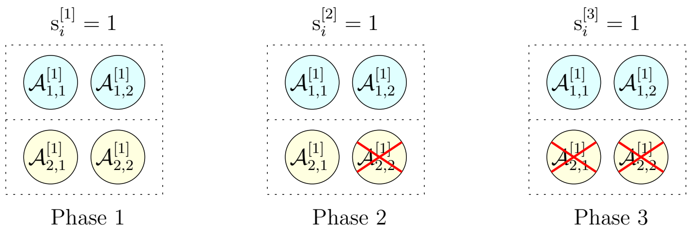
$$|\mathcal{A}_{l,j}| + |\mathcal{A}_{j,l}| < k, \quad \forall j \neq l,\ j, l \in [1:\eta] \tag{29}$$

$$|\mathcal{A}_{l,j}^{[1]}| + |\mathcal{A}_{j,l}^{[1]}| < k, \quad \forall j \neq l,\ j, l \in [1:\eta^{[1]}] \tag{30}$$

where k is defined in (2)*.*

Proof. This proof borrows tool from linear algebra. Our approach is to construct a set of linear equations based on $\mathcal{A}_{l,j}$ and $\mathcal{A}_{j,l}$, as well as the full rank property of encoding matrix, and then bound the size of $\mathcal{A}_{l,j}$ and $\mathcal{A}_{j,l}$. Details are provided in the long version [14]. ◀

▶ **Lemma 22.** *When $\eta^{[2]} \geq 1$, it holds true that $|\mathcal{A}_l| \geq n - 9t/4$, for any $l \in [1:\eta^{[2]}]$.*



$s_i^{[1]} = 1$
$s_i^{[2]} = 1$
$s_i^{[3]} = 1$
$\mathcal{A}_{1,1}^{[1]}$
$\mathcal{A}_{1,2}^{[1]}$
$\mathcal{A}_{2,1}^{[1]}$
$\mathcal{A}_{2,2}^{[1]}$
Phase 1
Phase 2
Phase 3

Figure 6 An example with $\eta^{[3]} \leq 1$ given $\eta^{[1]} = 2$.

Proof. This proof borrows tool from graph theory. It consists of the following steps:

- Step (a): Transform the network into a graph that is within the family of graphs, for a fixed $i^\star$ in $\mathcal{A}_{l^\star}^{[2]}$ and $l^\star \in [1 : \eta^{[2]}]$. One graph example is depicted in Fig. 3.
- Step (b): Bound the size of a group of honest processors, denoted by $\mathcal{D}'$, using the result of a derived lemma based on the defined graph, i.e., $|\mathcal{D}'| \geq n - 9t/4 - 1$.
- Step (c): Argue that every processor in $\mathcal{D}'$ has the same initial message as Processor $i^\star$.
- Step (d): Conclude from Step (c) that $\mathcal{D}'$ is a subset of $\mathcal{A}_{l^\star}$, i.e., $\mathcal{D}' \cup \{i^\star\} \subseteq \mathcal{A}_{l^\star}$ and conclude that the size of $\mathcal{A}_{l^\star}$ is bounded by the number determined in Step (b), i.e., $|\mathcal{A}_{l^\star}| \geq |\mathcal{D}'| + 1 \geq n - 9t/4 - 1 + 1$, for $l^\star \in [1 : \eta^{[2]}]$.

More details are provided in the long version [14] due to the lack of space here. ◀

▶ **Lemma 23.** *For COOL with $n \geq 3t + 1$, if $\eta^{[1]} = 2$ then it holds true that $\eta^{[3]} \leq 1$.*

Proof. Given two groups of honest processors with two *non-empty* updated messages at the end of Phase 1 (i.e., $\eta^{[1]} = 2$), we prove that at least one group will be reduced to the one with *empty* updated message after error detecting and masking in Phases 2 and 3 (i.e., $\eta^{[3]} \leq 1$, see one example in Fig. 5). The high-level proof procedure is described in Fig. 6 for one scenario. Specifically, given $\eta^{[1]} = 2$, at the end of Phase 1 Groups $\mathcal{A}_1$ and $\mathcal{A}_2$ all have their success indicators as ones. After error detecting and masking in Phase 2 (resp. Phase 3), then Sub-group $\mathcal{A}_{2,2}$ (resp. $\mathcal{A}_{2,1}$) will update their success indicators as zeros. In this case, Group $\mathcal{A}_2$ will be reduced to the one with zero indicators at the end of Phase 3 (i.e., $\eta^{[3]} \leq 1$). Details are provided in the long version [14]. ◀

6 Conclusion and discussion

In this work we proposed COOL, a deterministic error-free signature-free BA protocol designed from coding theory, together with graph theory and linear algebra, with optimal or near optimal performance in resilience, round complexity, and communication complexity, simultaneously. The results reveal that coding is an effective approach for achieving the fundamental limits of Byzantine agreement and its variants. To see the advantages of coding, let us compare three schemes below.

Scheme with full data transmission (DT): The multi-valued consensus problems could be solved by invoking ℓ instances of 1-bit consensus in sequence (ℓ-1-bit scheme). Since distributed processors need to exchange the whole message data, it is not surprising that this scheme results in a high communication complexity (see Fig. 4).

Non-coding scheme with reduced DT: To reduce the communication complexity, one possible solution is to let distributed processors exchange only one piece of data, instead of the whole data. However, if the data is *uncoded*, this reduction in data exchange could possibly lead to a consensus error. This is because honest processors might not get enough information from others and hence this protocol is more vulnerable to the Byzantine attacks.

Coding scheme with reduced DT: If the data is *coded* with error correction code, the distributed processors could possibly be able to detect and correct errors, even with reduced data transmission. Hence, the coding scheme with reduced DT could be robust against attacks from dishonest processors and could be error free. One example of the coding scheme with reduced DT is the proposed COOL in the BA and BB settings. Table 2 provides some comparison between the above three schemes. We believe that the coding schemes could be applied to broader settings of distributed algorithms.

Table 2 Comparison of three consensus schemes.

Schemes	communication complexity	error free
Scheme with full DT	high	yes
Non-coding scheme with reduced DT	low	no
Coding scheme with reduced DT	low	yes

References

1 I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Efficient synchronous Byzantine consensus. Available on ArXiv: `arXiv:1704.02397`, 2017.

2 I. Abraham, D. Dolev, and J. Halpern. An almost-surely terminating polynomial protocol for asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 405–414, 2008.

3 S. Ahmed. A performance study of Hyperledger Fabric in a smart home and IoT environment, 2019. Available on https://www.duo.uio.no/handle/10852/70940.

4 Z. Amsden et al. The libra blockchain. *Libra White Paper*, 2019.

5 E. Androulaki et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, 2018.

6 E. Berlekamp. Nonbinary BCH decoding (abstr.). *IEEE Trans. Inf. Theory*, 14(2):242–242, 1968.

7 P. Berman, J. Garay, and K. Perry. Bit optimal distributed consensus. *Computer Science*, pages 313–321, 1992.

8 V. Buterin. A next-generation smart contract and decentralized application platform. *Ethereum White Paper*, 2015.

9 C. Cachin and S. Tessaro. Asynchronous verifiable information dispersal. In *IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2005.

10 N. Cai and R. Yeung. Network error correction, II: Lower bounds. *Communications in Information and Systems*, 6(1):37–54, 2006.

11 R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience. In *25th Annual ACM Symposium on the Theory of Computing*, pages 42–51, 1993.

12 F. Casino, T. Dasaklis, and C. Patsakis. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics*, 36:55–81, March 2019.

13 B. Chan and E. Shi. Streamlet: Textbook streamlined blockchains. In *2nd ACM Conference on Advances in Financial Technologies(AFT)*, pages 1–11, October 2020.

14 J. Chen. Fundamental limits of Byzantine agreement. Available on ArXiv: `arXiv:10965`, 2020.

15 B. Chor and B. Coan. A simple and efficient randomized Byzantine agreement algorithm. *IEEE Transactions on Software Engineering*, 11(6):531–539, June 1985.

16 B. Coan and J. Welch. Modular construction of a Byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, March 1992.

17 T. Courtade and T. Halford. Coded cooperative data exchange for a secret key. *IEEE Trans. Inf. Theory*, 62(7):3785–3795, 2016.

18 M. Demir, M. Alalfi, O. Turetken, and A. Ferworn. Security smells in smart contracts. In *IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, July 2019.

19 D. Dolev and R. Reischuk. Bounds on information exchange for Byzantine agreement. *Journal of the ACM*, 32(1):191–204, 1985.

20 D. Dolev and H. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

21 P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, August 1997.

22 M. Fischer and N. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, June 1982.

23 M. Fitzi and M. Hirt. Optimally efficient multi-valued Byzantine agreement. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 163–168, 2006.

24 C. Fragouli, D. Lun, M. Medard, and P. Pakzad. On feedback for network coding. In *2007 41st Annual Conference on Information Sciences and Systems*, March 2007.

25 C. Ganesh and A. Patra. Optimal extension protocols for Byzantine broadcast and agreement. In *Distributed Computing*, July 2020.

26 J. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.

27 G. Goodson, J. Wylie, G. Ganger, and M. Reiter. Efficient Byzantine-tolerant erasure-coded storage. In *International Conference on Dependable Systems and Networks*, June 2004.

28 C. Gorenflo, S. Lee, L. Golab, and S. Keshav. FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.

29 W. Halbawi, T. Ho, H. Yao, and I. Duursma. Distributed reed-solomon codes for simple multiple access networks. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2014.

30 M. Hammoudeh, I. Ghafir, A. Bounceur, and T. Rawlinson. Continuous monitoring in mission-critical applications using the internet of things and blockchain. In *3rd International Conference on Future Networks and Distributed Systems*, July 2019.

31 T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger. Byzantine modification detection in multicast networks using randomized network coding. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, June 2004.

32 D. Huang and D. Medhi. A Byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks. In *19th IEEE International Parallel and Distributed Processing Symposium*, April 2005.

33 Hyperledger. Hyperledger-fabricdocs documentation, 2020.

34 S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros. Resilient network coding in the presence of Byzantine adversaries. *IEEE Trans. Inf. Theory*, 54(6):2596–2603, June 2008.

35 J. Katz and C. Koo. On expected constant-round protocols for Byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91–112, 2009.

36 S. Kim, T. Ho, M. Effros, and S. Avestimehr. New results on network error correction: Capacities and upper bounds. In *Proc. Inf. Theory and App. Workshop (ITA)*, 2010.

37 R. Koetter and M. Medard. An algebraic approach to network coding. In *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, June 2001.

38 M. Krohn, M. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *IEEE Symposium on Security and Privacy (SP)*, May 2004.

39 L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, July 1982.

40 S. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE Trans. Inf. Theory*, 49(2):371–381, February 2003.

41 G. Liang and N. Vaidya. Error-free multi-valued consensus with Byzantine failures. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 11–20, June 2011.

42 A. Loveless, R. Dreslinski, and B. Kasikci. Optimal and error-free multi-valued Byzantine consensus through parallel execution. Available on : https://eprint.iacr.org/2020/322, 2020.

43 A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The Honey Badger of BFT protocols. In *2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

44 B. Mohanta, D. Jena, S. Panda, and S. Sobhanayak. Blockchain technology: A survey on applications and security privacy challenges. *Internet of Things*, 8, 2019.

45 Y. Moses and O. Waarts. Coordinated traversal: $(t+1)$-round Byzantine agreement in polynomial time. *Journal of Algorithms*, 17(1):110–156, July 1994.

46 A. Mostéfaoui, H. Moumen, and M. Raynal. Signature-free asynchronous binary Byzantine consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time. *Journal of the ACM*, 62(4), 2015.

47 A. Mostéfaoui and M. Raynal. Signature-free asynchronous Byzantine systems: from multivalued to binary consensus with $t < n/3$, $O(n^2)$ messages, and constant time. In *Proc. 22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO'15)*, July 2015.

48 K. Nayak, L. Ren, E. Shi, N. Vaidya, and Z. Xiang. Improved extension protocols for Byzantine broadcast and agreement. In *International Symposium on Distributed Computing (DISC)*, October 2020.

49 R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Advances in Cryptology – EUROCRYPT 2018. Lecture Notes in Computer Science*, volume 10821, pages 3–33, March 2018.

50 A. Patra. Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity. In *International Conference on Principles of Distributed Systems (OPODIS)*, pages 34–49, 2011.

51 A. Patra and P. Rangan. Communication optimal multi-valued asynchronous Byzantine agreement with optimal resilience. In *International Conference on Information Theoretic Security*, pages 206–226, 2011.

52 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

53 X. Qi, Z. Zhang, C. Jin, and A. Zhou. BFT-Store: Storage partition for permissioned blockchain via erasure coding. In *IEEE 36th International Conference on Data Engineering*, April 2020.

54 M. Rabin. Randomized Byzantine generals. In *24th Annual Symposium on Foundations of Computer Science*, November 1983.

55 I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

56 R. Roth. *Introduction to coding theory.* Cambridge University Press, 2006.

57 E. Shi. Streamlined blockchains: A simple and elegant approach (a tutorial and survey). *Advances in Cryptology – ASIACRYPT 2019, Lecture Notes in Computer Science*, 11921:3–17, November 2019.

58 J. Sousa and A. Bessani. From Byzantine consensus to BFT state machine replication: A latency-optimal transformation. In *2012 Ninth European Dependable Computing Conference*, May 2012.

59 J. Sousa, A. Bessani, and M. Vukolic. A Byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018.

60 The Hyperledger White Paper Working Group. An introduction to Hyperledger. *Hyperledger White Paper*, 2018.

61 M. Yan and A. Sprintson. On error correcting algorithms for the cooperative data exchange problem. In *2014 International Symposium on Network Coding (NetCod)*, June 2014.

62 H. Yao and T. Ho. Distributed reed-solomon codes for simple multiple access networks, September 2015. US Patent 9,148,173.
63 Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *IEEE International Conference on Computer Communications*, April 2008.

A The extension of COOL to the Byzantine broadcast problem

For the BB setting, at first the leader sends each processor an ℓ-bit message that can be considered as the initial message. Then, COOL is applied into this BB setting from this step to achieve the consistent and validated consensus, with similar performance as in BA setting.

B Algorithm of the proposed COOL protocol

Algorithm 1 : COOL protocol, code for Processor i, $i \in [1:n]$.

```
1: Initially set $\boldsymbol{w}^{(i)} = \boldsymbol{w}_i$.
2: Processor $i$ encodes its message into $n$ symbols as $y_j^{(i)} \triangleq \boldsymbol{h}_j^{\intercal}\boldsymbol{w}_i,\ j \in [1:n]$.
   Phase 1
3: Processor $i$ sends $(y_j^{(i)}, y_i^{(i)})$ to Processor $j$, $\forall j \in [1:n], j \neq i$.
4: for $j = 1:n$ do
5:     if $((y_i^{(j)}, y_j^{(j)}) == (y_i^{(i)}, y_j^{(i)}))$ then
6:         Processor $i$ sets $\mathrm{u}_i(j) = 1$.
7:     else
8:         Processor $i$ sets $\mathrm{u}_i(j) = 0$.
9: if $(\sum_{j=1}^{n} \mathrm{u}_i(j) >= n - t)$ then
10:    Processor $i$ sets its success indicator as $\mathrm{s}_i = 1$.
11: else
12:    Processor $i$ sets $\mathrm{s}_i = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
13: Processor $i$ sends the value of $\mathrm{s}_i$ to all other processors.
14: Processor $i$ creates sets $\mathcal{S}_p = \{j : \mathrm{s}_j = p, j \in [1:n]\}$, $p \in \{0,1\}$, from received $\{\mathrm{s}_j\}_{j=1}^{n}$.
   Phase 2
15: if $(\mathrm{s}_i == 1)$ then
16:    Processor $i$ sets $\mathrm{u}_i(j) = 0, \forall j \in \mathcal{S}_0$.
17:    if $(\sum_{j=1}^{n} \mathrm{u}_i(j) < n - t)$ then
18:        Processor $i$ sets $\mathrm{s}_i = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
19:        Processor $i$ sends the value of $\mathrm{s}_i$ to all other processors.
20: Processor $i$ updates $\mathcal{S}_0$ and $\mathcal{S}_1$ based on the newly received success indicators.
   Phase 3
21: if $(\mathrm{s}_i == 1)$ then
22:    Processor $i$ sets $\mathrm{u}_i(j) = 0, \forall j \in \mathcal{S}_0$.
23:    if $(\sum_{j=1}^{n} \mathrm{u}_i(j) < n - t)$ then
24:        Processor $i$ sets $\mathrm{s}_i = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
25:        Processor $i$ sends the value of $\mathrm{s}_i$ to all other processors.
26: Processor $i$ updates $\mathcal{S}_0$ and $\mathcal{S}_1$ based on the newly received success indicators.
27: if $(\sum_{j=1}^{n} \mathrm{s}_j >= 2t + 1)$ then
28:    Processor $i$ sets the binary vote as $\mathrm{v}_i = 1$.
29: else
30:    Processor $i$ sets the binary vote as $\mathrm{v}_i = 0$.
31: Processor $i$ runs the one-bit consensus with all other processors on votes $\{\mathrm{v}_j\}_j$ using one-bit
    consensus from [7, 16].
32: if (the consensus of the votes $\{\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_n\}$ is 1) then
33:    Processor $i$ goes to next phase.
34: else
35:    Processor $i$ sets $\boldsymbol{w}^{(i)} = \phi$ and considers it as a final consensus and stops here.
   Phase 4
36: if $(\mathrm{s}_i == 0)$ then
37:    Processor $i$ updates $y_i^{(i)} \leftarrow \mathrm{Majority}(\{y_i^{(j)} : \mathrm{s}_j = 1, j \in [1:n]\})$.
38:    Processor $i$ sends updated $y_i^{(i)}$ to Processor $j$, $\forall j \in \mathcal{S}_0, j \neq i$.
39:    Processor $i$ decodes message with new observations $\{y_1^{(1)}, \cdots, y_n^{(n)}\}$ and updates $\boldsymbol{w}^{(i)}$.
40: Processor $i$ outputs consensus as the updated message $\boldsymbol{w}^{(i)}$ and stops.
```