Communication-Efficient Signature-Free Asynchronous Byzantine Agreement

Fan Li and Jinyuan Chen
Department of Electrical Engineering
Louisiana Tech University
Email: {fli005, jinyuan}@latech.edu

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518010

***Abstract*—In this work, we focus on the problem of byzantine agreement (BA), in which n distributed processors seek to reach an agreement on an ℓ-bit value, but up to t processors might be corrupted by a Byzantine adversary and act as dishonest nodes. In particular, we consider the communication-efficient BA in an asynchronous setting, where the network communication might have arbitrarily time delay. The primary challenge of designing the BA protocol in this setting is that we need to handle both the message delay from honest nodes and the Byzantine behavior from dishonest nodes simultaneously. In this work we propose a new signature-free asynchronous byzantine agreement (ABA) protocol, which achieves the optimal communication complexity of $O(n\ell)$ when $\ell \geq t \log t$, given $n \geq 5t+1$. A protocol is said to be signature-free if the protocol design does not depend on the cryptographic machinery such as hashing and signature. To the best of our knowledge, this is the first *signature-free* ABA protocol that achieves the optimal communication complexity of $O(n\ell)$ when ℓ is almost linearly scaled with t.**

I. Introduction

Byzantine agreement (BA), as a 40-year-old distributed consensus problem (cf. [1], [2]), is one of the fundamental building blocks for distributed computing and cryptography (e.g. [1]–[12]). In the BA problem, n distributed processors seek to reach an agreement on some value, but up to t processors might be corrupted by a Byzantine adversary and act as dishonest nodes. The BA problem has been broadly studied in varying settings, such as authenticated BA (cf. [5]–[12]), unauthenticated BA (cf. [13]–[16]), BA with synchronous communication (cf. [17]–[20]), BA with asynchronous communication (cf. [21]–[33]), binary BA (cf. [13]–[15], [30], [34]), and multi-valued BA (cf. [35]–[45]). In this work, we focus on the *communication-efficient* multi-valued asynchronous BA (ABA) in an unauthenticated (signature-free) setting. A protocol is said to be signature-free if the protocol design does not depend on the cryptographic machinery such as hashing and signature.

In the research line of signature-free multi-valued ABA, the work of [39] proposed an ABA protocol with communication complexity $O(n\ell+n^5 \log n)$, given $n \geq 3t+1$. This result was recently improved in [45] to the communication complexity of $O(n\ell + n^4 \log n)$. The communication complexity of [39] and [45] is optimal when $\ell \geq n^4 \log n$ and $\ell \geq n^3 \log n$, respectively. A recent work of [16] has shown that the optimal communication complexity $O(n\ell)$ can be achieved when $\ell \geq t \log t$ in the synchronous setting. This inspires us to raise the following question.

Is it possible to achieve the optimal communication complexity $O(n\ell)$ when $\ell \geq t \log t$ in the signature-free asynchronous setting?

In this work, we seek to investigate the above question. Specifically, we propose a new signature-free ABA protocol that achieves the communication complexity of $O(\max\{n\ell, nt \log t\})$, given $n \geq 5t+1$. It implies that the optimal communication complexity of $O(n\ell)$ can be achieved when $\ell \geq t \log t$. To the best of our knowledge, this is the first signature-free ABA protocol that achieves the optimal communication complexity of $O(n\ell)$ when ℓ is almost linearly scaled with t.

The proposed protocol (called as A-COOL) is extended from the COOL protocol introduced in the synchronous setting [16]. Note that, in the asynchronous setting the message transmission might be delayed with arbitrary time. Therefore, we need to handle both the message delay from honest nodes and the Byzantine behavior from dishonest nodes simultaneously. This is the primary challenge of designing the BA protocol in this asynchronous setting.

We provide some comparison between different ABA protocols in Table I and Fig. 1. Note that Fig. 1 focuses on the comparison in the communication complexity exponent, which is defined in (1), capturing the exponent of communication complexity performance. As shown in Fig. 1, compared to the protocols in [39] and [45], A-COOL protocol provides additive gains up to 3 and 2, respectively, in terms of communication complexity exponent performance.

TABLE I
Comparison of different ABA protocols

Model	Resilience	Communication complexity	Round complexity	Signature -free
[39]	$t < \frac{n}{3}$	$O(n\ell + n^5 \log n)$	$O(1)$	yes
[45]	$t < \frac{n}{3}$	$O(n\ell + n^4 \log n)$	$O(1)$	yes
Proposed	$t < \frac{n}{5}$	$O(\max\{n\ell, nt \log t\})$	$O(1)$	yes

This work was supported in part by the NSF EPSCoR-Louisiana Materials Design Alliance (LAMDA) Program under Grant OIA-1946231 and in part by the Louisiana Board of Regents Support Fund (BoRSF) Research Competitiveness Subprogram under Grant 32-4121-40336.

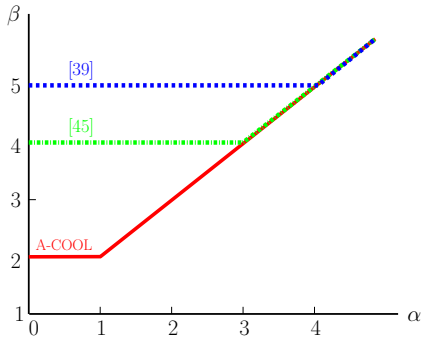

Fig. 1. Communication complexity exponent β vs. message size exponent α of the proposed A-COOL, the protocols in [39] and [45], focusing on the case with $\delta = 1$.

This paper is organized as follows. The system model is provided in Section II, while the main results are presented in Section III. The proposed A-COOL is detailed in Section IV. The conclusion is drawn in Section V.

II. System Models

We consider the ABA problem, in which a set of n distributed processors $\{P_1, P_2, \cdots, P_n\}$ seek to reach an agreement on an ℓ-bit value, but up to t processors might be corrupted by a Byzantine adversary and act as dishonest nodes. The processors are communicated over an asynchronous network. Each pair of processors is connected via a reliable and private communication channel. We consider a static adversary, denoted by $\mathcal{A}_t$, which selects the set of corrupted processors before starting the protocol. We assume that the adversary has complete knowledge of the state of the other processors. The processors that are not corrupted by the adversary are honest processors. We provide the formal definition of ABA as below.

Definition 1 (ABA)**.** *A protocol for a set of processors $\{P_1, P_2, \cdots, P_n\}$, where P_i holds an initial ℓ-bit input value $\boldsymbol{w}_i$, $i \in [1:n]$, is said to be an ABA protocol tolerating an adversary $\mathcal{A}_t$, if the following properties hold*

- *Termination: Every honest processor eventually almost-surely*[1] *terminates.*
- *Consistency: All of the honest processors eventually decide on the same value.*
- *Validity: If every honest processor holds an initial value $\boldsymbol{w}$, then all of the honest processors eventually decide on this initial value $\boldsymbol{w}$.*

In this work we use the following three parameters to measure the quality of a BA protocol:

- Resilience: the maximum number of dishonest nodes allowed in the protocol, denoted by t.
- Communication complexity: the total number of communication bits during the protocol execution, denoted by b.
- Round complexity: the expected number of rounds taken by the protocol to terminate, denoted by r.

[1]The probability that an honest processor is undecided after r rounds approaches 0 as r grows to infinity [46].

We define the notion of *communication complexity exponent* of the ABA protocol as

$$\beta(\alpha, \delta) \triangleq \lim_{n\to\infty} \frac{\log b(n, \delta, \alpha)}{\log n} \tag{1}$$

where $\alpha \triangleq \lim_{n\to\infty} \frac{\log \ell}{\log n}$ (*message size exponent*) and $\delta \triangleq \lim_{n\to\infty} \frac{\log t}{\log n}$ (*faulty size exponent*). β, α and δ capture the exponents of communication complexity b, the message size ℓ and faulty size t respectively with n as the base, when n is large.

Recall that a protocol is said to be signature-free if the protocol design does not depend on the cryptographic machinery such as hashing and signature. A protocol is said to be information-theoretic secure if the protocol can tolerate the computationally unbounded adversary.

III. Main Results

In this section we will provide the main results of this work.

Theorem 1 (ABA)**.** *For the ABA problem, the proposed A-COOL is a signature-free multi-valued ABA protocol that reaches an agreement on an ℓ-bit value with communication complexity $O(\max\{n\ell, nt\log t\})$, given $n \geq 5t+1$.*

Proof. The proposed A-COOL protocol is provided in Section IV. □

Theorem 1 implies that the proposed A-COOL protocol achieves the optimal communication complexity $O(n\ell)$ when $\ell \geq t\log t$. The following result is directly derived from Theorem 1.

Corollary 1 (Communication complexity exponent)**.** *For the proposed A-COOL protocol, when $n \geq 5t+1$, the communication complexity exponent is*

$$\beta(\alpha, \delta) = \max\{1+\alpha, 1+\delta\}. \tag{2}$$

The proposed A-COOL protocol does not assume the cryptographic machinery such as hashing and signature (signature free). Also, the proposed protocol can tolerate the computationally unbounded adversary (information-theoretic secure).

IV. The Proposed A-COOL Protocol

In this section, we will provide the proposed A-COOL protocol. This proposed A-COOL protocol seeks to reach an agreement on an ℓ-bit value $\boldsymbol{w}$ over n distributed processors $\{P_i\}_{i=1}^n$ under asynchronous communication. In the design of the A-COOL protocol, coding scheme is used to reduce the communication complexity during the information exchange steps. Specifically, the $(n,\ k)$ Reed-Solomon error correction code will be used in the protocol design. We will show that the proposed A-COOL is a signature-free multi-valued ABA protocol that reaches an agreement on an ℓ-bit value with communication complexity $O(\max\{n\ell, nt\log t\})$, when $n \geq 5t+1$. This result serves as the proof of Theorem 1.

The proposed A-COOL protocol is composed of at most four phases, as shown in Algorithm 1. The proposed protocol is guaranteed to satisfy the termination, consistency, and

validity conditions. In what follows, we will describe the four phases in detail. In the description of the proposed A-COOL, we will just consider the case of $t \leq \frac{n-1}{5}$ and $t = \Omega(n)$.

We first define some notations that will be used in the protocol design. Let $\boldsymbol{w}^{(i)}$ be the updated value at P_i, $i \in [1:n]$. Let $\mathrm{u}_i^{[p]}(j)$ be the binary link indicator for P_i and P_j, for $p \in [1:3]$ and $i, j \in [1:n]$. Let $\mathrm{s}_i^{[p]}$ be the binary success indicator at P_i, for $p \in [1:3]$ and $i \in [1:n]$. In our design the initial value of $\boldsymbol{w}^{(i)}$ is set as $\boldsymbol{w}^{(i)} = \boldsymbol{w}_i$, $i \in [1:n]$. The binary link indicator is initially set as $\mathrm{u}_i^{[1]}(j) = 0$, $i, j \in [1:n]$.

A. Phase 1: exchange symbols and update information

The main idea of Phase 1 is to exchange coded symbols and update information.

1) Encode and exchange symbols: By applying Reed-Solomon error correction code, $P_i, i \in [1:n]$, first encodes its ℓ-bit initial value $\boldsymbol{w}_i$ into ℓ/k-bit symbols as

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

where $\boldsymbol{h}_j$ is defined as $\boldsymbol{h}_j \triangleq [h_{j,1}, h_{j,2}, \cdots, h_{j,k}]^{\mathsf{T}}$, for

$$h_{j,m} \triangleq \prod_{\substack{p=1 \\ p \neq m}}^{k} \frac{j-p}{m-p}, \quad j \in [1:n], \quad m \in [1:k]. \tag{4}$$

In our protocol design, the parameter k is designed as $k \triangleq \left\lfloor \frac{t}{5} \right\rfloor + 1$. Note that for the (n, k) Reed-Solomon error correction code constructed in the Galois Field $GF(2^c)$, the condition of $n \leq 2^c - 1$ needs to be satisfied (cf. [47]). In this protocol, the parameter c is designed as

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

It can be verified that the condition of $n \leq 2^c - 1$ is satisfied with the designed parameters k and c. In this protocol, each coded symbol $y_j^{(i)}$ has c bits. After this encoding process, P_i, $i \in [1:n]$, sends a pair of coded symbols $(y_j^{(i)}, y_i^{(i)})$ to P_j for $j \in [1:n] \setminus i$.

2) Update information: Upon receiving $n - t$ pairs of symbols $\{(y_i^{(j)}, y_j^{(j)})\}_j$, P_i compares the received $(y_i^{(j)}, y_j^{(j)})$ with its symbol pair $(y_i^{(i)}, y_j^{(i)})$ and sets a link indicator $\mathrm{u}_i^{[1]}(j)$ as

$$\mathrm{u}_i^{[1]}(j) = \begin{cases} 1 & \text{if } (y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)}) \\ 0 & \text{else} \end{cases} \tag{6}$$

for $i \in [1:n]$. The value of $\mathrm{u}_i^{[1]}(j) = 0$ implies that P_i and P_j might have mismatched value, i.e., $\boldsymbol{w}^{(i)} \neq \boldsymbol{w}^{(j)}$. If $\mathrm{u}_i^{[1]}(j) = 0$, we consider the pair of $(y_i^{(j)}, y_j^{(j)})$ from P_j as a mismatched symbol pair at P_i. By counting the number of mismatched symbol pairs, P_i sets a success indicator $\mathrm{s}_i^{[1]}$ as

$$\mathrm{s}_i^{[1]} = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} \mathrm{u}_i^{[1]}(j) \geq n - 2t \\ 0 & \text{else} \end{cases} \tag{7}$$

for $i \in [1:n]$. The event of $\mathrm{s}_i^{[1]} = 0$ means that the number of mismatched symbol pairs is more than $2t$, which implies that the initial value of P_i does not match the majority of other processors' initial values. If $\mathrm{s}_i^{[1]} = 0$, P_i updates $\boldsymbol{w}^{(i)}$ as $\boldsymbol{w}^{(i)} = \phi$ (a default value), else keeps the original value of $\boldsymbol{w}^{(i)}$.

3) Exchange success indicators: P_i, $i \in [1:n]$, sends the value of success indicator $\mathrm{s}_i^{[1]}$ to $P_j, j \in [1:n] \setminus i$. Upon receiving $n - t$ success indicators $\{\mathrm{s}_j^{[1]}\}_j$, P_i creates the sets:

$$\mathcal{S}_1 \triangleq \{j : \mathrm{s}_j^{[1]} = 1, j \in [1:n]\}, \quad \mathcal{S}_0 \triangleq [1:n] \setminus \mathcal{S}_1 \tag{8}$$

based on $\{\mathrm{s}_j^{[1]}\}_j$. In this step, different processors might have different views on $\mathcal{S}_0$ and $\mathcal{S}_1$, due to the inconsistent value possibly sent from dishonest processors.

Remark 1. In Phase 1, since $y_j^{(i)}$ defined in (3) has c bits, the communication complexity of exchanging coded data (see Line 3 in Algorithm 1), denoted by b_1, is $b_1 = 2cn(n-1)$ bits. Since the success indicator $\mathrm{s}_i^{[1]}$ has only 1 bit, the communication complexity of exchanging success indicators (see Line 14), denoted by b_2, is $b_2 = n(n-1)$ bits.

Remark 2. Note that dishonest processors could send arbitrary (inconsistent) value to different honest processors, which could make honest processors output their updated values differently in this phase. Phase 2 will further handle the issue of inconsistent updated values among honest processors.

B. Phase 2: mask errors, and update success indicator

In this phase, the main goal is to mask errors from honest processors, based on the result of Phase 1.

1) Mask errors: If $\mathrm{s}_i^{[1]} = 1$, P_i sets $\mathrm{u}_i^{[2]}(j) = \mathrm{u}_i^{[1]}(j), \forall j \in \mathcal{S}_1$ and $\mathrm{u}_i^{[2]}(j) = 0,\ \forall j \in \mathcal{S}_0$.

2) Exchange success indicators: If $\mathrm{s}_i^{[1]} = 1$, P_i updates $\mathrm{s}_i^{[2]}$ as in (7) using updated values of $\{\mathrm{u}_i^{[2]}(j)\}_{j=1}^n$, else sets $\mathrm{s}_i^{[2]} = 0$. Then P_i sends $\mathrm{s}_i^{[2]}$ to $P_j, j \in [1:n] \setminus i$. If $\mathrm{s}_i^{[2]} = 0$, P_i updates $\boldsymbol{w}^{(i)}$ as $\boldsymbol{w}^{(i)} = \phi$, else keeps the original value of $\boldsymbol{w}^{(i)}$.

3) Update $\mathcal{S}_0$ and $\mathcal{S}_1$: Upon receiving $n - t$ success indicators $\{\mathrm{s}_j^{[2]}\}_j$, P_i updates the sets of $\mathcal{S}_0$ and $\mathcal{S}_1$ as in (8) for $i \in [1:n]$.

Remark 3. Since the success indicator $\mathrm{s}_i^{[2]}$ has 1 bit, the communication complexity of exchanging success indicators in Phase 2 (see Line 26 in Algorithm 1), denoted by b_3, is $b_3 = n(n-1)$ bits.

C. Phase 3: mask errors, update information, and vote

In Phase 3, the main goal is to mask the rest of the errors from the honest processors, and then vote for stopping in this phase or going to Phase 4.

1) Mask errors: If $\mathrm{s}_i^{[2]} = 1$, P_i sets $\mathrm{u}_i^{[3]}(j) = \mathrm{u}_i^{[2]}(j), \forall j \in \mathcal{S}_1$ and $\mathrm{u}_i^{[3]}(j) = 0,\ \forall j \in \mathcal{S}_0$.

2) Exchange success indicator: If $\mathrm{s}_i^{[2]} = 1$, P_i updates $\mathrm{s}_i^{[3]}$ as in (7) using updated values of $\{\mathrm{u}_i^{[3]}(j)\}_{j=1}^n$, else sets $\mathrm{s}_i^{[3]} = 0$. Then P_i sends $(\mathrm{s}_i^{[3]}, y_j^{(i)})$ to $P_j, j \in [1:n] \setminus i$. If $\mathrm{s}_i^{[3]} = 0$, P_i updates $\boldsymbol{w}^{(i)}$ as $\boldsymbol{w}^{(i)} = \phi$, else keeps the original value of $\boldsymbol{w}^{(i)}$.

Algorithm 1 : A-COOL protocol, code for P_i, $i \in [1:n]$

1: Set $\boldsymbol{w}^{(i)} = \boldsymbol{w}_i$; $\mathrm{u}_i^{[1]}(j) = 0$, $j \in [1:n]$.

Phase 1

2: P_i encodes $\boldsymbol{w}^{(i)}$ into n symbols as $y_j^{(i)} \triangleq \boldsymbol{h}_j^{\intercal}\boldsymbol{w}_i, j \in [1:n]$.
3: P_i sends $(y_j^{(i)}, y_i^{(i)})$ to P_j, $\forall j \in [1:n] \setminus i$.
4: **upon** receiving $n-t$ pairs of symbols $\{(y_i^{(j)}, y_j^{(j)})\}_j$
5: **for** each received $(y_i^{(j)}, y_j^{(j)})$ **do**
6: **if** $\big((y_i^{(j)}, y_j^{(j)}) == (y_i^{(i)}, y_j^{(i)})\big)$ **then**
7: P_i sets $\mathrm{u}_i^{[1]}(j) = 1$.
8: **else**
9: P_i sets $\mathrm{u}_i^{[1]}(j) = 0$.
10: **if** $\big(\sum_{j=1}^{n} \mathrm{u}_i^{[1]}(j) >= n - 2t\big)$ **then**
11: P_i sets $\mathrm{s}_i^{[1]} = 1$.
12: **else**
13: P_i sets $\mathrm{s}_i^{[1]} = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
14: P_i sends $\mathrm{s}_i^{[1]}$ to $P_j, j \in [1:n] \setminus i$.
15: **upon** receiving $n-t$ success indicators $\{\mathrm{s}_j^{[1]}\}_j$
16: P_i sets $\mathcal{S}_1 = \{j : \mathrm{s}_j^{[1]} = 1, j \in [1:n]\}$, $\mathcal{S}_0 = [1:n] \setminus \mathcal{S}_1$.

Phase 2

17: **if** $(\mathrm{s}_i^{[1]} == 1)$ **then**
18: P_i sets $\mathrm{u}_i^{[2]}(j) = \mathrm{u}_i^{[1]}(j), \forall j \in \mathcal{S}_1$.
19: P_i sets $\mathrm{u}_i^{[2]}(j) = 0, \forall j \in \mathcal{S}_0$.
20: **if** $\big(\sum_{j=1}^{n} \mathrm{u}_i^{[2]}(j) >= n - 2t\big)$ **then**
21: P_i sets $\mathrm{s}_i^{[2]} = 1$.
22: **else**
23: P_i sets $\mathrm{s}_i^{[2]} = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
24: **else**
25: P_i sets $\mathrm{s}_i^{[2]} = 0$.
26: P_i sends $\mathrm{s}_i^{[2]}$ to $P_j, j \in [1:n] \setminus i$.
27: **upon** receiving $n-t$ success indicators $\{\mathrm{s}_j^{[2]}\}_j$
28: P_i updates $\mathcal{S}_0$ and $\mathcal{S}_1$, based on $\{\mathrm{s}_j^{[2]}\}_j$.

Phase 3

29: **if** $(\mathrm{s}_i^{[2]} == 1)$ **then**
30: P_i sets $\mathrm{u}_i^{[3]}(j) = \mathrm{u}_i^{[2]}(j), \forall j \in \mathcal{S}_1$.
31: P_i sets $\mathrm{u}_i^{[3]}(j) = 0, \forall j \in \mathcal{S}_0$.
32: **if** $\big(\sum_{j=1}^{n} \mathrm{u}_i^{[3]}(j) >= n - 2t\big)$ **then**
33: P_i sets $\mathrm{s}_i^{[3]} = 1$.
34: **else**
35: P_i sets $\mathrm{s}_i^{[3]} = 0$ and $\boldsymbol{w}^{(i)} = \phi$.
36: **else**
37: P_i sets $\mathrm{s}_i^{[3]} = 0$.
38: P_i sends $(\mathrm{s}_i^{[3]}, y_j^{(i)})$ to $P_j, j \in [1:n] \setminus i$.
39: **upon** receiving $n-t$ success indicators $\{\mathrm{s}_j^{[3]}\}_j$
40: P_i updates $\mathcal{S}_0$ and $\mathcal{S}_1$, based on $\{\mathrm{s}_j^{[3]}\}_j$.
41: **if** $(\sum_j \mathrm{s}_j^{[3]} >= n - 2t)$ **then**
42: P_i sets $\mathrm{v}_i = 1$.
43: **else**
44: P_i sets $\mathrm{v}_i = 0$.
45: P_i runs the one-bit consensus with all other processors on the n votes $\{\mathrm{v}_i\}_{i=1}^n$, by using the one-bit consensus protocol from [30].
46: **if** (the consensus of the votes $\{\mathrm{v}_i\}_{i=1}^n$ is 1) **then**
47: P_i goes to next phase.
48: **else**
49: P_i outputs $\boldsymbol{w}^{(i)} = \phi$ as a final consensus and stops.

Phase 4

50: **if** $(\mathrm{s}_i^{[3]} == 0)$ **then**
51: $y_i^{(i)} \leftarrow \text{Majority}(\{y_i^{(j)} : j \in \mathcal{S}_1\})$.
52: P_i sends $y_i^{(i)}$ to P_j, $\forall j \in [1:n] \setminus i$.
53: **if** $(\mathrm{s}_i^{[3]} == 0)$ **then**
54: **upon** receiving $n-t$ symbols $\{y_j^{(j)}\}_j$
55: P_i decodes message and updates it into $\boldsymbol{w}^{(i)}$ based on the received symbols.
56: P_i outputs $\boldsymbol{w}^{(i)}$ as the final consensus and stops.

3) Update $\mathcal{S}_0$ and $\mathcal{S}_1$: Upon receiving $n-t$ success indicators $\{\mathrm{s}_j^{[3]}\}_j$, P_i updates the sets of $\mathcal{S}_0$ and $\mathcal{S}_1$ as in (8) for $i \in [1:n]$.

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

$$\mathrm{v}_i = \begin{cases} 1 & \text{if } \sum_j \mathrm{s}_j^{[3]} \geq n - 2t \\ 0 & \text{else} \end{cases} \tag{9}$$

based on the $n-t$ received success indicators $\{\mathrm{s}_j^{[3]}\}_j$. v_i denotes a vote from P_i for stopping in this phase or going to Phase 4.

5) One-bit consensus on the n votes: $P_i, i \in [1,n]$, runs the one-bit consensus with all other processors on the n votes $\{\mathrm{v}_i\}_{i=1}^n$, by using the one-bit consensus protocol from [30]. The consensus can be reached with $O(n^2)$ bits of communication complexity, and $O(1)$ rounds of round complexity, for $t < n/5$. If the consensus of the votes $\{\mathrm{v}_i\}_{i=1}^n$ is 0, then every honest processor P_i outputs $\boldsymbol{w}^{(i)} = \phi$ as a final consensus and stops, else every honest processor goes to Phase 4.

Remark 4. In Phase 3, since $\mathrm{s}_i^{[3]}$ and $y_j^{(i)}$ has 1 bit and c bits respectively, the communication complexity of exchanging pairs of $(\mathrm{s}_i^{[3]}, y_j^{(i)})$, $i \in [1:n], j \in [1:n] \setminus i$ (see Line 38 in Algorithm 1), denoted by b_4, is $b_4 = (1+c)n(n-1)$ bits.

Remark 5. The work of [30] provides the most efficient unauthenticated binary ABA protocol that achieves the communication complexity $O(n^2)$ and round complexity $O(1)$, assuming a common coin oracle. Since we run the one-

bit consensus from [30], the communication complexity of invoking one-bit consensus (see Line 45 in Algorithm 1), denoted by b_5, is $b_5 = O(n^2)$ bits. Since the round complexity of this step is constant (i.e., $O(1)$) and the round complexity of other steps is also constant, it gives that the round complexity of the proposed protocol is constant, i.e., $O(1)$.

Remark 6. It can be proved that at the end of this phase, there exists at most 1 group of honest processors, where the honest processors within this group have the same non-empty updated value ($\boldsymbol{w}^{(i)} \neq \phi$), and the honest processors outside this group have the same empty updated value ($\boldsymbol{w}^{(i)} = \phi$).

D. Phase 4: exchange coded symbols and make consensus

The idea of this phase is to calibrate and update the final value of the honest processors in the set of $\mathcal{S}_0$ such that every honest processor outputs the same value as the final consensus.

1) Update symbols with majority rule: P_i, $i \in \mathcal{S}_0$, updates the value of $y_i^{(i)}$ as $y_i^{(i)} \leftarrow$ Majority$(\{y_i^{(j)} : j \in \mathcal{S}_1\})$, where the coded symbols $\{y_i^{(j)}\}_{j\in\mathcal{S}_1}$ were received in Phase 3. Majority$(\bullet)$ is a function defined to return the most frequent value in the list, based on the majority rule. Note that P_i, $i \in \mathcal{S}_1$ keeps its previous value of $y_i^{(i)}$.

2) Broadcast updated symbol: $P_i, i \in [1:n]$, sends the value of $y_i^{(i)}$ to P_j, $\forall j \in [1:n] \setminus i$.

3) Decode: Upon receiving $n-t$ coded symbols $\{y_j^{(j)}\}_j$, P_i decodes message and updates it into $\boldsymbol{w}^{(i)}$ using the $n-t$ received symbols, for $i \in \mathcal{S}_0$. For P_i, $i \in \mathcal{S}_1$, it skips this step.

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

Remark 7. In Phase 4, since $y_i^{(i)}$ has c bits, the communication complexity of exchanging the coded symbols (see Line 52 in Algorithm 1), denoted by b_6, is $b_6 = cn(n-1)$ bits.

Remark 8. Note that up to $\lfloor \frac{n-k}{2} \rfloor$ Byzantine errors can be corrected in the design of an (n,k) error correction code, using some efficient decoding algorithms [47]–[49]. It can be verified that in Step 3 of this phase, every honest processor decodes and outputs the same message, with the parameter design of k and c in this protocol.

E. Performance analysis of A-COOL

This section provides the performance analysis of the proposed A-COOL protocol.

Lemma 1. *A-COOL reaches an agreement on an ℓ-bit value with the communication complexity of $O(\max\{n\ell, nt\log t\})$ bits and the round complexity of $O(1)$ rounds, given $n \geq 5t+1$.*

Proof. The proposed A-COOL reaches an agreement on an ℓ-bit value given $n \geq 5t+1$ (see Lemma 2).

By adding the communication complexity $b_1, b_2, \cdots, b_6$ expressed in Remarks 1, 3, 4, 5 and 7, respectively, the total communication complexity of the proposed A-COOL, denoted by b, is computed as

$$\begin{aligned} b &= \sum_{i=1}^{6} b_i \\ &= O(cn(n-1) + n^2) \\ &= O(\max\{\ell/t, \log n\} \cdot n(n-1) + n^2) \\ &= O(\max\{\ell n^2/t, n^2 \log n\}) \quad \text{bits} \end{aligned} \tag{10}$$

where c is designed as $c = \lceil \frac{\max\{\ell,\ (t/5+1)\cdot\log(n+1)\}}{k} \rceil$ for $k = \lfloor \frac{t}{5} \rfloor + 1$. In the description of the proposed A-COOL, we just considered the case of $t \leq \frac{n-1}{5}$ and $t = \Omega(n)$. For this case, the total communication complexity shown in (10) can be rewritten as $b = O(\max\{n\ell, nt\log t\})$ bits. It can also be shown that the same communication complexity performance can be achieved, for the case with relatively small t compared to n. Due to the lack of space, the detail is provided in the long version [50]. By combining the above two cases, we conclude that the total communication complexity of A-COOL is $b = O(\max\{n\ell, nt\log t\})$ bits.

Given that the round complexity of the one-bit consensus in Phase 3 is constant, it implies that the round complexity of the proposed protocol is constant, i.e., $O(1)$. □

Lemma 2. *The proposed A-COOL is a signature-free ABA protocol, which satisfies the termination, consistency and validity conditions in all executions, given $n \geq 5t+1$.*

Proof. The proof of Lemma 2 will use the tools from coding theory and graph theory. Due to the lack of space, the proof is provided in the long version [50]. □

V. Conclusion

In this work, we studied the ABA problem and proposed a new signature-free protocol, i.e., A-COOL, which is communication efficient. Specifically, the proposed A-COOL protocol achieves the optimal communication complexity $O(n\ell)$ when $\ell \geq t\log t$, given $n \geq 5t+1$. The proposed A-COOL protocol can also be extended to the asynchronous Byzantine broadcast (ABB) problem, by adding one step of broadcasting ℓ-bit message from the leader to the distributed processors. The ℓ-bit message sent from the leader will serve as the initial value at every distributed node in the ABA problem. Then, the proposed A-COOL can be applied here to achieve the same performance as in the ABA setting, in terms of resilience, communication complexity and round complexity.

References

[1] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[2] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[3] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the third symposium on Operating systems design and implementation*, Feb. 1999, pp. 173–186.

[4] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, Oct. 2017, pp. 51–68.

[5] D. Dolev and H. Strong, "Authenticated algorithms for Byzantine agreement," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 656–666, Nov. 1983.

[6] M. Rabin, "Randomized byzantine generals," in *24th Annual Symposium on Foundations of Computer Science*, Nov. 1983.

[7] P. Feldman and S. Micali, "Byzantine agreement in constant expected time," in *26th Annual Symposium on Foundations of Computer Science*, Oct. 1985.

[8] D. Dolev and R. Reischuk, "Bounds on information exchange for Byzantine agreement," *Journal of the ACM*, vol. 32, no. 1, pp. 191–204, Jan. 1985.

[9] P. Feldman and S. Micali, "Optimal algorithms for Byzantine agreement," in *20th annual ACM symposium on Theory of computing*, Jan. 1988.

[10] C. Cachin and S. Tessaro, "Asynchronous verifiable information dispersal," in *IEEE Symposium on Reliable Distributed Systems (SRDS)*, Oct. 2005.

[11] J. Katz and C. Koo, "On expected constant-round protocols for Byzantine agreement," *Journal of Computer and System Sciences*, vol. 75, no. 2, pp. 91–112, Feb. 2009.

[12] Y. Lu, Z. Lu, Q. Tang, and G. Wang, "Dumbo-MVBA: Optimal multi-valued validated asynchronous Byzantine agreement, revisited," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2020, pp. 129–138.

[13] B. Coan and J. Welch, "Modular construction of a Byzantine agreement protocol with optimal message bit complexity," *Information and Computation*, vol. 97, no. 1, pp. 61–85, Mar. 1992.

[14] P. Berman, J. Garay, and K. Perry, "Bit optimal distributed consensus," *Computer Science*, pp. 313–321, 1992.

[15] J. Garay and Y. Moses, "Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 247–290, 1998.

[16] J. Chen, "Fundamental limits of Byzantine agreement," 2020, available on: arXiv:2009.10965v2.

[17] B. Chor and B. Coan, "A simple and efficient randomized Byzantine agreement algorithm," *IEEE Transactions on Software Engineering*, vol. 11, no. 6, pp. 531–539, Jun. 1985.

[18] B. Pfitzmann and M. Waidner, "Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$," IBM Research Report, 1996.

[19] P. Feldman and S. Micali, "An optimal probabilistic protocol for synchronous Byzantine agreement," *SIAM Journal on Computing*, vol. 26, no. 4, pp. 873–933, Aug. 1997.

[20] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, "Efficient synchronous Byzantine consensus," 2017, available on ArXiv: https://arxiv.org/abs/1704.02397.

[21] M. Ben-Or, "Another advantage of free choice: Completely asynchronous agreement protocols," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Aug. 1983, pp. 27–30.

[22] G. Bracha, "An asynchronous $[(n-1)/3]$-resilient consensus protocol," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Aug. 1984.

[23] C. Attiya, D. Dolev, and J. Gil, "Asynchronous Byzantine consensus," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Aug. 1984, pp. 119–133.

[24] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[25] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *Journal of the ACM*, vol. 32, no. 4, pp. 824–840, Oct. 1985.

[26] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus," *Journal of the ACM*, vol. 34, no. 1, pp. 77 – 97, Jan. 1987.

[27] R. Canetti and T. Rabin, "Fast asynchronous Byzantine agreement with optimal resilience," in *25th Annual ACM Symposium on the Theory of Computing*, Jun. 1993, pp. 42–51.

[28] J. Aspnes and O. Waarts, "Randomized consensus in expected $O(N \log^2 N)$ operations per processor," *SIAM Journal on Computing*, vol. 25, no. 5, pp. 1024–1044, Oct. 1996.

[29] 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)*, Aug. 2008, pp. 405–414.

[30] 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*, vol. 62, no. 4, Sep. 2015.

[31] 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*, Oct. 2016.

[32] S. Duan, M. Reiter, and H. Zhang, "BEAT: Asynchronous BFT made practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Jan. 2018, pp. 2028–2041.

[33] I. Abraham, D. Malkhi, and A. Spiegelman, "Asymptotically optimal validated asynchronous Byzantine agreement," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2019, pp. 337–346.

[34] T. Crain, "A simple and efficient asynchronous randomized binary Byzantine consensus algorithm," 2020, available on: arXiv:2002.04393.

[35] R. Turpin and B. Coan, "Extending binary Byzantine agreement to multivalued Byzantine agreement," *Information Processing Letters*, vol. 18, no. 2, pp. 73–76, Feb. 1984.

[36] M. Fitzi and M. Hirt, "Optimally efficient multi-valued byzantine agreement," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2006, pp. 163–168.

[37] A. Patra and P. Rangan, "Communication optimal multi-valued asynchronous Byzantine agreement with optimal resilience," in *International Conference on Information Theoretic Security*, 2011, pp. 206–226.

[38] 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)*, Jun. 2011, pp. 11–20.

[39] A. Patra, "Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity," in *Proceedings of the 15th international conference on Principles of Distributed Systems, OPODIS*, 2011, pp. 34–49.

[40] M. Hirt and P. Raykov, "Multi-valued Byzantine broadcast: The $t < n$ case," *Advances in Cryptology – ASIACRYPT 2014, Lecture Notes in Computer Science*, vol. 8874, pp. 448–465, Nov. 2014.

[41] C. Ganesh and A. Patra, "Broadcast extensions with optimal communication and round complexity," in *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2016, pp. 371–380.

[42] W. Chongchitmate and R. Ostrovsky, "Information-theoretic broadcast with dishonest majority for long messages," in *Theory of Cryptography. TCC 2018. Lecture Notes in Computer Science*, vol. 11239, Nov. 2018, pp. 370–388.

[43] C. Ganesh and A. Patra, "Optimal extension protocols for byzantine broadcast and agreement," in *Distributed Computing*, Jul. 2020.

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

[45] E. Nayak, L. Ren, E. Shi, N. Vaidya, and Z. Xiang, "Improved extension protocols for Byzantine broadcast and agreement," in *34th International Symposium on Distributed Computing (DISC)*, Oct. 2020.

[46] G. Bracha, "Asynchronous Byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, Nov. 1987.

[47] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[48] R. Roth, *Introduction to coding theory*. Cambridge University Press, 2006.

[49] E. Berlekamp, "Nonbinary BCH decoding (abstr.)," *IEEE Trans. Inf. Theory*, vol. 14, no. 2, pp. 242–242, Mar. 1968.

[50] F. Li and J. Chen, "Communication-efficient signature-free asynchronous Byzantine agreement," 2021, available on: http://www2.latech.edu/%7Ejinyuan/aba2021.pdf.