Sublinear Message Bounds of Authenticated Implicit Byzantine Agreement[*]

Manish Kumar [†]   Anisur Rahaman Molla [‡]

Abstract

This paper studies the message complexity of authenticated Byzantine agreement (BA) in synchronous, fully-connected distributed networks under an honest majority. We focus on the so-called *implicit* Byzantine agreement problem where each node starts with an input value and at the end a non-empty subset of the honest nodes should agree on a common input value by satisfying the BA properties (i.e., there can be undecided nodes)[*]. We show that a sublinear (in n, number of nodes) message complexity BA protocol under honest majority is possible in the standard PKI model when the nodes have access to an unbiased global coin and hash function. In particular, we present a randomized Byzantine agreement algorithm which, with high probability achieves implicit agreement, uses $\tilde{O}(\sqrt{n})$ messages, and runs in $\tilde{O}(1)$ rounds while tolerating $(1/2 - \epsilon)n$ Byzantine nodes for any fixed $\epsilon > 0$, the notation $\tilde{O}$ hides a $O(\text{polylog } n)$ factor[†]. The algorithm requires standard cryptographic setup PKI and hash function with a static Byzantine adversary. The algorithm works in the CONGEST model and each node does not need to know the identity of its neighbors, i.e., works in the KT_0 model. The message complexity (and also the time complexity) of our algorithm is optimal up to a $\text{polylog } n$ factor, as we show a $\Omega(\sqrt{n})$ lower bound on the message complexity.

To the best of our knowledge, this is the first sublinear message complexity result of Byzantine agreement. A quadratic message lower bound is known for any deterministic BA protocol (due to Dolev-Reischuk [JACM 1985]). The existing randomized BA protocols have at least quadratic message complexity in the honest majority setting. Our result shows the power of a global coin in achieving significant improvement over the existing results. It can be viewed as a step towards understanding the message complexity of randomized Byzantine agreement in distributed networks with PKI.

Keywords: Distributed Algorithm, Randomized Algorithm, Byzantine Agreement, Message Complexity, Global Coin, Cryptographic Assumptions.

1 Introduction

Byzantine agreement is a fundamental and long studied problem in distributed networks [33, 7, 35]. In this problem, all the nodes are initiated with an input value. The Byzantine agreement problem is required to satisfy: (i) the honest nodes must decide on the same input value; and (ii) if all the honest nodes receive the same input value, then they must decide on that value[‡]. This should be done in the presence of a constant fraction of Byzantine nodes that can arbitrarily deviate from the protocol executed by the honest nodes.

[†]Indian Statistical Institute, Kolkata 700108, India.E-mail: manishsky27@gmail.com.

[‡]Indian Statistical Institute, Kolkata 700108, India. E-mail: anisurpm@gmail.com.

[*]Implicit BA is a generalization of the classical BA problem. Throughout, we write Byzantine agreement or BA to mean implicit Byzantine agreement.

[†]We use the abbreviated phrase "w.h.p." or "with high probability," signifying a probability of at least $1 - 1/n^\epsilon$ for a constant ϵ, where n corresponds to the number of nodes present in the network.

[‡]Throughout, we interchangeably use the term 'non-Byzantine' and 'honest', and similarly, 'Byzantine' and 'faulty'.

Byzantine agreement provides a critical building block for creating attack-resistant distributed systems. Its importance can be seen from widespread and continued application in many domains such as wireless networks [29, 34, 48, 47], sensor networks [46], grid computing [6], peer-to-peer networks [45] and cloud computing [49], cryptocurrencies [11, 15, 3, 28, 37], secure multi-party computation [9] etc. However, despite huge research, we lack efficient practical solutions to Byzantine agreement for large networks. A main drawback for this is the *large message complexity* of currently known protocols, as mentioned by many systems papers [4, 5, 13, 36, 50]. The best known Byzantine protocols have (at least) quadratic message complexity [23, 10, 16, 30], even in the authenticated settings [14, 41]. In a distributed network, nodes communicate with their neighbors by passing messages. Therefore, communication cost plays an important role to analyze the performance of the algorithms, as also mentioned in many papers [1, 25, 24, 41].

King and Saia [27] presented the first Byzantine agreement algorithm that *breaks* the quadratic message barrier in synchronous, complete networks. The message complexity of their algorithm is $\tilde{O}(n^{1.5})$. Later, Braud-Santoni et al. [12] improved this to $\tilde{O}(n)$ message complexity. Both works require the nodes to know the IDs of the other nodes a priori. This model is known as KT_1 model (known till 1) [43]. Another challenging model is KT_0, where nodes do not know their neighbors a priori [43]. Note that in KT_0 model, nodes can know their neighbors easily by communicating to all the neighbors, perhaps in a single round, but that will incur $\Omega(n^2)$ messages. The KT_0 model is more appropriate to the modern distributed networks which are permissionless, i.e., nodes can enter and leave the network at will.

In this paper, our main focus is to study the message complexity of the Byzantine agreement problem in the KT_0 model under the assumption of cryptographic setup and a global coin (as defined in [44]). In fact, we study the implicit version of the Byzantine agreement, where not all the honest nodes need to be decided; only a non-empty subset of the honest node must decide on an input value. Our main result is a randomized algorithm to solve implicit Byzantine agreement using sublinear messages (only $\tilde{O}\sqrt{n}$) while tolerating $f \leq (1/2 - \epsilon)n$ Byzantine nodes, where n is the number of nodes in the network, f is the number of Byzantine nodes and $\epsilon > 0$ is a fixed constant. The implicit algorithm can be easily extended to the explicit Byzantine agreement (where all the honest nodes must decide) using $O(n \log n)$ messages only. The algorithm is simple and easily implementable, which is highly desired for practical purposes. While the assumptions on the Public Key Infrastructure (PKI) set up with keyed hash function and the global coin together make the model a little weaker, they are realistic and implementable [§]. Similar assumptions were made earlier in the literature, e.g., in Algorand, Gilad et al. [21] uses "seed" and PKI setup, where "seed" is essentially the shared random bits. In their approach, they formed a set of candidate nodes and reached an agreement with the help of the sortition algorithm (see Section 5 of [21]) using proof-of-stake (PoS). They further used verifiable random functions (VRFs) [38] for the verification of the candidate nodes which return hash and proof. In our work, we do not require the assumption of VRFs. The message complexity of Algorand is $\tilde{O}(n)$, albeit for the explicit agreement.

Without PKI setup, hash function and global coin assumptions, we do not know if a sublinear (or even a linear) message complexity Byzantine agreement algorithm is possible or not. So far, the best results have quadratic message bound in the KT_0 model and sub-quadratic in the KT_1 model.

Our result introduces the first *sublinear message complexity* Byzantine agreement algorithm and at the same time tolerates optimal resilience, i.e., $f \leq (1/2 - \epsilon)n$. We also argue a lower bound on the message complexity of the problem. The lower bound shows that the message complexity of our algorithm is optimal up to a $\mathrm{polylog}\, n$ factor. Our results can be viewed as a step towards understanding the message complexity of randomized BA in distributed networks under the assumptions of PKI, hash function and global coin.

Paper Organization: The rest of the paper is organized as follows. In the rest of this section, we state our result and introduce the model and definition. Section 2 is a related work, which introduces the seminal works done in the same direction. Section 3 presents the main implicit Byzantine agreement algorithm. In

---

[§]Throughout, we interchangeably use the term 'hash function' and 'keyed hash function'.

Section 4, we present the lower bound on the message complexity to support the optimality of our algorithm. Finally, we conclude with some open problems in Section 5.

1.1 Our Results

We show the following main results.

Theorem 1.1 (Implicit Agreement). Consider a synchronous, fully-connected, anonymous network of n nodes and CONGEST communication model. Assuming a public-key infrastructure with the keyed hash function, there exists a randomized algorithm which, with the help of global coin, solves implicit Byzantine agreement with high probability in $O(\log^2 n)$ rounds and uses $O(n^{0.5} \log^{3.5} n)$ messages while tolerating $f \leq (1/2 - \epsilon)n$ Byzantine nodes under non-adaptive adversary, where ϵ is any fixed positive constant.

Theorem 1.2 (Explicit Agreement). Consider a synchronous, fully-connected network of n nodes and CONGEST communication model. Assuming a public-key infrastructure with the keyed hash function, there exists a randomized algorithm which, with the help of global coin, solves Byzantine agreement with high probability in $O(\log^2 n)$ rounds and uses $O(n \log n)$ messages while tolerating $f \leq (1/2 - \epsilon)n$ Byzantine nodes under non-adaptive adversary, where ϵ is any fixed positive constant.

Theorem 1.3 (Lower Bound). Consider any algorithm A that has access to an unbiased global coin and sends at most $f(n)$ messages (of arbitrary size) with high probability on a complete network of n nodes. If A solves the authenticated Byzantine agreement under honest majority with constant probability, then $f(n) \in \Omega(\sqrt{n})$.

1.2 Model and Definitions

The network is synchronous and fully-connected graph of n nodes. Initially, nodes do not know their neighbors, also known as KT_0 model [43]. Nodes have access to an unbiased global coin through which they can generate shared random bits. The network is $f \leq (1/2 - \epsilon)n$ resilient, i.e., at most $(1/2 - \epsilon)n$ nodes (among n nodes) could be faulty, for any constant $\epsilon > 0$. We consider Byzantine fault [33]. A Byzantine faulty node can behave maliciously such that it sends any arbitrary message or no message in any round to mislead the protocol, e.g., it may send different input values to different nodes, or it may not send any message to some of the nodes in a particular round. We assume that a *static* adversary controls the Byzantine nodes, which selects the faulty nodes before the execution starts. However, the adversary can adaptively choose when and how a node behaves maliciously. Further, the adversary is rushing and has full information– the adversary knows the states of all the nodes and can view all the messages in a round before sending out its own messages for that round. We assume that each node possesses multi-valued input of size $O(\log n)$ provided by the adversary.

We assume the existence of digital signatures, Public Key Infrastructure (PKI) and hash function. Each node possesses a public-secret key pair (p_k, s_k). Secret keys are generated randomly, and correspondingly public keys are generated with the help of a prime order group's generator. Therefore, the public keys are not skewed but random. Trusted authority also provides other cryptographic primitives for each node, and certifies each node's public keys. Nodes use digital signatures for the authentication of any information. We abstract away the details of the cryptographic setup; assuming it is a standard framework. Public key (p_k) of all the nodes, hash function, shared random bits (generated through global coin) and n are the common knowledge for all the nodes.

We consider the *CONGEST* communication model [43], where a node is allowed to send a message of size (typically) $O(\log n)$ or $O(\text{polylog}(n))$ bits through an edge per round. The message complexity of an algorithm is the total number of messages sent by all the non-faulty nodes throughout the execution of the algorithm.

3

Definition 1 (Implicit Byzantine Agreement). Suppose initially all the nodes have an input value (say, provided by an adversary). An implicit Byzantine agreement holds when the following properties hold: (i) the final state of all the non-Byzantine nodes is either "decided" or "undecided"; (ii) all the "decided" non-Byzantine nodes must agree on the same value (consistency property); (iii) if all the non-Byzantine nodes have the same input value then they must decide on that value (validity property); (iv) all the non-Byzantine nodes eventually reach to the final state, where at least one non-Byzantine node must be in the "decided" state (termination).

Definition 2 (Keyed Hash Function). Let $\mathcal{K}_h, X$ be two non-empty finite sets and H be an b-bit function such that $H : \mathcal{K}_h \times X \rightarrow \{0, 1\}^b$, where H follows three properties: (i) Preimage Resistant - Given a hash value h, it should be difficult to find any message m such that $h = H(k, m)$. (ii) Second Preimage Resistant - Given an input m_1, it should be difficult to find a different input m_2 such that $H(k, m_1) = H(k, m_2)$. (iii) Collision Resistant - It should be difficult to find two different messages m_1 and m_2 such that $H(k, m_1) = H(k, m_2)$.

In the *Byzantine Broadcast*, there is a designated sender (could be Byzantine or honest) who broadcasts the input values to the nodes. Termination and consistency are the same as in the case of Byzantine agreement. In the case of validity, all the non-faulty nodes output the same value if the sender is honest. Also, that value should be the input value of the sender.

2 Related Work

Byzantine agreement and Byzantine broadcast have been studied extensively in various models and settings in the last four decades, starting from its classic introduction by Lamport, Shostak and Pease [33, 42]. They presented protocols and fault tolerance bounds for two settings (both synchronous). Without cryptographic assumptions (the unauthenticated setting), Byzantine broadcast and agreement can be solved if $f < n/3$. Assuming digital signatures (the authenticated setting), Byzantine broadcast can be solved if $f < n$ and Byzantine agreement can be solved if $f < n/2$. The initial protocols had exponential message complexities [33, 42]. Fully polynomial protocols were later shown for both the authenticated ($f < n/2$) [14] and the unauthenticated ($f < n/3$) [20] settings. Both protocols require $f + 1$ rounds of communication, which matches the lower bound on round complexity for deterministic protocols [18].

Our Byzantine agreement algorithm makes use of a few past results. First, we make use of the concept of a set of candidate nodes, which is a subset of nodes. The notion of a committee (set of candidate nodes) is used in, e.g., [8, 22, 31, 32]. Finally, we adapt the Byzantine Agreement algorithm designed by Dolev et al. [14].

The previous results in the same direction are summarized in the Table 1. In comparison with our work, we are using shared random value and non-adaptive adversary with rushing adversary. Shared random value help us to break the linear communication (message) complexity. In implicit agreement, we have sublinear communication complexity while in explicit agreement it is linear with cryptographic assumptions and tolerate $f \leq (1/2 - \epsilon)n$ Byzantine nodes.

3 Authenticated Implicit Byzantine Agreement

In this section, we present a randomized Byzantine agreement algorithm in a complete n-node network that tolerates $f \leq (1/2 - \epsilon)n$ Byzantine nodes under a public-key infrastructure, keyed hash function and access to a global coin. The algorithm incurs $\tilde{O}(\sqrt{n})$ messages, has latency $\tilde{O}(1)$, and has high success probability.

In the algorithm, we run a subroutine BA protocol, which can tolerate $f \leq (1/2 - \epsilon)n$ Byzantine nodes and may have a polynomial message (and time) complexity. In fact, we adapt the classical algorithm

Comparison of the Results					
Protocol	Agreement Type	Communication (in bits)	Adversary	Cryptographic Assumptions	Resilience
Santoni et al. [12]	Implicit	$\tilde{O}(n)$	Non-adaptive	No	$f < n/(3+\epsilon)$
King-Saia [27]	Explicit	$\tilde{O}(n^{1.5})$	Adaptive	No	$f \leq (1/3 - \epsilon)n$
Dolev-Strong [14]	Explicit	$\tilde{O}(n^3)$	Adaptive	Yes	$f < n/2$
Momose-Ren [41]	Explicit	$\tilde{O}(n^2)$	Adaptive	Yes	$f < n/2$
Abraham et al. [2]	Explicit	$O(n^2)^*$	Adaptive	Yes	$f < n/2$
This paper	Implicit	$\tilde{O}(n^{0.5})$	Non-adaptive	Yes	$f \leq (1/2 - \epsilon)n$
This paper	Explicit	$\tilde{O}(n)$	Non-adaptive	Yes	$f \leq (1/2 - \epsilon)n$

Table 1: Comparison of various model with our result. ϵ is any positive constant. Our results assume a global coin and hash function while others are not. * indicates the bound holds in expectation.

presented by Dolev-Strong [14].[¶] Dolev-Strong designed an algorithm for the Byzantine broadcast (BB) problem, which can be converted into a Byzantine agreement algorithm with an initial round to broadcast the input Byzantine nodes under a public-key infrastructure and access to a global coin. The communication/bit complexity is $O(\kappa n^3)$ [19]. However, using multi-signature it can be improved to $O(\kappa n^2 + n^3)$, where κ is a security parameter which is essentially the maximum size of the messages [41]. While the original Dolev-Strong BB protocol tolerates $f \leq n - 1$ faults, the converted BA protocol works for the honest majority nodes, i.e., tolerates $f < n/2$ Byzantine faults which is optimal for an authenticated BA [2, 17, 26, 33, 39, 41].

Dolev-Strong algorithm is deterministic and has a latency of $f + 1$ rounds. The general idea of the BB protocol is to form a signature chain consisting of signatures from distinct nodes. A signature chain of $f + 1$ signatures must contain a non-faulty signature from a non-faulty node which can send the value to all the other nodes. The protocol is designed in such a way that in $f + 1$ rounds it forms a signature chain of size $f + 1$. We adapted the Dolev-Strong BB protocol for the Byzantine agreement problem and used it in our implicit BA algorithm.

Let us now describe the implicit BA algorithm. The public key (p_k) of the nodes is known to all the nodes (as distributed by the trusted third party), but a node does not know which port or edge is connecting to which node (having a particular public key). To minimize the message complexity, a generic idea is to select a set of small-size candidate nodes, which will be responsible for solving the (implicit) agreement among themselves. Thus, it is important to have honest majority in the set of candidate nodes. For this, a random set of nodes, called as *candidate nodes* or *committee nodes*, of size $O(\log n)$ is selected. Let us

---

[¶]One can use other suitable BA protocols, e.g., the protocol in [41].

denote the candidate nodes set by $\mathcal{C}$. A Byzantine node may try to claim that it is in $\mathcal{C}$, which needs to be stopped to guarantee the honest majority in $\mathcal{C}$. To overcome this problem, we take the help of a global coin and a keyed hash function, which together determine the candidate nodes.

A common random number, say r, is generated with the help of the global coin. The random number should be large enough to use as the key to the hash function. Note that the random number is generated after the selection of the Byzantine nodes by the adversary. Every node uses its respective public key (p_k) as the message and r as the key of the hash function, say, H. That is they compute $H_r(p_k)$. For each p_{k_i}, we represent its hash value $H_r(p_{k_i})$ as H_i. Since the hash values are random (with high probability), the smallest $c \log n$ values among the n hash values are chosen to be the candidate nodes, where c is a suitable constant (to be fixed later). More precisely, a node i with the hash value H_i is in $\mathcal{C}$ if it is in the smallest $c \log n$ values of the set $\{H_i : i = 1, 2, \ldots, n\}$. Therefore, a node can easily figure out the candidate nodes since it knows the random number r, hash function and the public keys of all the nodes. However, the node does not know the ports connecting to them (as KT_0 model).

Although a node knows all the candidate nodes (in fact, their public keys), it does not know the edges connecting to the candidate nodes, since the network is anonymous, i.e., KT_0 model. It can be known by the candidate nodes by sending a message to all the nodes, but that will cost $n \log n$ messages. Since knowing each other is message expensive in this model, the candidate nodes communicate among themselves via some other nodes. For this, each candidate node randomly samples $\Theta(\sqrt{n \log n})$ nodes among all the n nodes; call them as *referee nodes*. The reason behind sampling so many referee nodes is to make sure there is at least one common "non-faulty" referee node between any pair of candidate nodes. The candidate nodes communicate with each other via the referee nodes. Notice that a node may be sampled as a referee node by multiple candidate nodes. It might happen that a Byzantine referee node may change the value of an honest candidate node before forwarding it to the candidate nodes. To avoid this, we take advantage of digital signature. Each referee node signs the input value before transmitting it to its referee nodes. As a digital signature helps to detect forging messages, each candidate node considers only the genuine messages received from the referee nodes.

Thus, we have a small committee of nodes (i.e., $\mathcal{C}$) with the highly probable honest majority and the committee nodes can communicate via the referee nodes. Then we apply the Dolev-Strong BA protocol [14] in the committee to achieve agreement. Let us now present the adapted Dolev-Strong algorithm to work for the Byzantine agreement.

Step 0: Each candidate node u does the following in parallel. u signs and sends its input value to its corresponding referee nodes, say, $\mathcal{R}_u$. Each referee node w sends all the received values to its respective candidate nodes, say, $\mathcal{C}_w$. Therefore, the candidate nodes have the input values of all the candidate nodes. Now each candidate node proposes a value for the agreement based on the priority, along with all the signatures received corresponding to that input value. If there is a value that is sent by the majority of the nodes (i.e., more than $(c \log n)/2$ nodes), then that value gets the *highest priority*. In case of more than one majority, the value proposed by the maximum number of nodes gets the highest priority. There might be the case, two values are proposed by the same number of nodes; in that case, the larger input value gets the highest priority. If a candidate node has the highest priority input value, then it sends the value (after signing) to all the candidate nodes along with the received signatures for the highest priority value. Otherwise, the candidate node does not send anything. The reason of getting more than one majority value is that a Byzantine node may propose different values to different nodes. If no such majority value is received, then the candidate nodes decide on a default value, say, the minimum in the input value set. By sending the input value, we mean sending the input value along with all the signatures corresponding to that input value.

Then, the following two steps are performed iteratively[‖].

[‖]An iteration is the number of rounds required to send messages from one candidate node to all the candidate nodes via the referee nodes. Since we consider the CONGEST model, an iteration may take up to $O(\log n)$ rounds in our algorithm.

Step 1: If a candidate node u receives a set of at least i legitimate signatures in i^{th} iteration with a higher priority value than its earlier sent value, then u proposes this new highest priority value along with all the signatures to its referees nodes $\mathcal{R}_u$.

Step 2: If a referee node w receives a set of at least i legitimate signatures in i^{th} iteration (with the highest priority value) then w forwards this highest priority value to its corresponding candidate nodes $\mathcal{C}_w$.

The above two steps (i.e., Step 1 and 2) are performed for $O(c \log n)$ iterations and then the algorithm terminates. In the end, all the (non-faulty) candidate nodes have the same value, either the default value or the value possessed by the majority of nodes (the highest priority value). Intuitively, there are $O(c \log n)$ candidate nodes and a single node may propose the highest priority value in each iteration (say, the single node with the highest priority value is faulty and send to only faulty nodes) and thus the highest priority value may propagate slowly. But eventually, the highest priority value is received by all the candidate nodes as the set of candidate nodes contains the majority of the non-faulty nodes (see Lemma 1) and there is a common non-faulty referee node between any pair of candidate nodes (see Lemma 2). On the other hand, if there is no highest priority value, then they decide on the default value. Thus, the (honest) candidate nodes agree on a unique value. A pseudocode is given in Algorithm 1.

Let us now show the above claims formally. We first show that the majority of the nodes in the candidate set are honest.

Lemma 1. The number of Byzantine nodes in the candidate set $\mathcal{C}$ is strictly less than $\frac{1}{2}|\mathcal{C}|$ with high probability.

Proof. Let α fraction of the nodes in the network are Byzantine where $\alpha = 1/2 - \epsilon$ is a fixed constant. So $\alpha + \epsilon = 1/2$. The nodes in $\mathcal{C}$ are chosen uniformly at random (i.e., with probability $1/n$) and the number of Byzantine nodes is at most αn, the probability that a particular node in $\mathcal{C}$ is Byzantine is at most α. Let $\mathcal{C}$ contain k nodes, $\{u_i \,|\, i = 1, 2, \ldots, k\}$. Let us define random variables X_is such that $X_i = 1$ if u_i is Byzantine, and 0 otherwise. Further, $X = \sum_{i=1}^{k} X_i$ is the total number of Byzantine nodes in $\mathcal{C}$. Then, by linearity of expectation, $E[X] \leq \alpha k$. Then, by Chernoff bounds [40],

$$\Pr(X \geq (\alpha + \epsilon)k) = \Pr(X \geq (1 + \epsilon/\alpha)E[X]) \leq \exp\left(-E[X](\epsilon/\alpha)^2/3\right)$$
$$\leq \exp\left(-(k\epsilon^2)/(3\alpha)\right) \text{for } k = |\mathcal{C}| = (3\alpha/\epsilon^2) \log n$$

Thus, $\Pr(X < (\alpha + \epsilon)|\mathcal{C}|) = \Pr(X < \frac{1}{2}|\mathcal{C}|) > 1 - 1/n$. In other words, $\mathcal{C}$ contains at most $(1/2 - \delta)|\mathcal{C}|$ Byzantine nodes with high probability, for any fixed $\delta > 0$. ∎

The candidate nodes communicate with each other via the referee nodes, which are sampled randomly by the candidate nodes. The number of referee nodes is sampled in such a way that there must be a common referee node between every pair of candidate nodes (so that the candidate nodes can communicate) and at the same time keep the message complexity lower. In fact, we need to guarantee a stronger result. Namely, there must be a *non-faulty* common referee node for reliable communication.

Lemma 2. Any pair of candidate nodes have at least one common non-faulty referee node with high probability.

Proof. Let us consider two candidate nodes v and w, and let x_i be the ith node selected by v. Let the random variable X_i be 1 if x_i is also chosen by w and 0 otherwise. Since the x_is are chosen independently at random, the X_is are independent. So, $Pr[X_i = 1] = \frac{2\sqrt{n \log n}}{n}$. Hence, the expected number of (choice of) referee nodes that v and w haven in common are:

$$E[X] = E[X_1] + E[X_2] + \cdots + E[X_{(2\sqrt{n \log n})}] (by\ linearity)$$
$$= 2\sqrt{n \log n} \cdot \frac{2\sqrt{n \log n}}{n} = 4 \log n \tag{1}$$

Algorithm 1 AUTHENTICATED-IMPLICIT-BA

Require: A complete n node anonymous network with $f \le (1/2 - \epsilon)n$ Byzantine nodes. Each node receives an input value provided by an (static) adversary, a pair of public-private keys (p_k, s_k), keyed hash function and a global coin. $\epsilon > 0$ is a fixed constant.
Ensure: Implicit Agreement.

1: Select $c \log n$ nodes as the candidate nodes set (say, $\mathcal{C}$), which have the smallest $c \log n$ hash values generated with the help of public key and random number. The value of the constant c is $3\alpha/\epsilon^2$, follows from Lemma 1.
2: Each candidate node u randomly samples $2\sqrt{n \log n}$ nodes as referee nodes (say, $\mathcal{R}_u$).
3: Each candidate node u signs and sends its input value (with signature) to $\mathcal{R}_u$.
4: Each referee node w sends all the received values to their respective candidate nodes $\mathcal{C}_w$ along with the legitimate signatures one by one. It takes $O(\log n)$ rounds.
5: Each candidate node u sends the input value along with all the received signatures to $\mathcal{R}_u$ based on the (highest) priority.   ▷ Highest priority is defined in the description, Step 0.
6: **for** the next $(c \log n)$ iterations, the candidate and referee nodes in parallel **do**
7:      Each referee node w checks:
8:      **if** w receives a set of at least i legitimate signatures in the ith iteration **then**
9:          w sends highest priority value to $\mathcal{C}_w$.
10:      **else**
11:          w does not send any messages.
12:      **end if**
13:      Each candidate node u checks:
14:      **if** u receives a set of at least i legitimate signatures in the ith iteration with a highest priority value than it sent earlier **then**
15:          u sends highest priority value to $\mathcal{R}_u$.
16:      **else**
17:          u does not send any messages.
18:      **end if**
19: **end for**
20: All the (non-faulty) candidate nodes have the same highest priority value on which they agree. Otherwise, if they do not receive any highest priority value, they agree on a default value, say, the minimum in the input value set.

Thus, by using the Chernoff bound [40], $\Pr[X < (1 - \delta)E[X]] < e^{-\delta^2 E[x]/2}$ for $\delta = 0.5$, we get

$$\Pr[X < (1 - 0.5)4 \log n] < e^{-(0.5)^2 (4 \log n)/2} < \frac{1}{\sqrt{n}} \tag{2}$$

For that reason at least $2 \log n$ choice of nodes by v are jointly chosen by v and w. As a consequence, the probability that none of these choices is non-faulty:

$$\left(\frac{1}{2} - \epsilon\right)^{2 \log n} < \frac{1}{n} \tag{3}$$

Therefore, the probability of selecting at least one non-faulty referee node from the sampled nodes of u is at least $1 - 1/n$. $\square$

Lemma 1 ensures that a committee (i.e., the candidate set) of size $O(\log n)$ with honest majority can be selected. Lemma 2 ensures that the committee nodes can communicate with each other reliably via the

referee nodes. Thus, the BA problem on n nodes reduces to a $O(\log n)$-size committee nodes. Then the Dolev-Strong BA protocol ensures that the committee nodes achieve Byzantine agreement among themselves deterministically. Therefore, the algorithm (Algorithm 1) correctly solves the implicit Byzantine agreement with high probability (i.e., among the committee nodes only). The non-faulty nodes which are not selected in the committee may set their state as "undecided" immediately after the selection of the candidate nodes.

Below, we analyze the message and time complexity of the algorithm.

Lemma 3. The message complexity of the authenticated implicit BA algorithm is $O(n^{0.5} \log^{3.5} n)$.

Proof. In Step 3 and 4 of the algorithm, $O(\log n)$ candidate nodes send their input value with signature to the other candidate nodes via the $2\sqrt{n \log n}$ referee nodes. Here the size of each message is $O(\kappa + \log n)$ bits, where κ is the security parameter, the size of the signature. So these two steps uses $O(\sqrt{n \log n}) \cdot O(\log n) \cdot O(\kappa + \log n) = O((\kappa + \log n)\sqrt{n \log^3 n})$ bits.

Inside the $O(\log n)$ iteration (Step 6): $O(\log n)$ candidate nodes may have at most $O(\log n)$ messages to be sent to the referee nodes for $O(\log n)$ rounds. The size of each message is $O(\kappa + \log n)$ bits. So it uses $O((\kappa + \log n)\sqrt{n \log^7 n})$ bits. Further, the same number of message bits are used when the referee nodes forward the messages to the candidate nodes. Thus, a total $O((\kappa + \log n)\sqrt{n \log^7 n})$ bits are used inside the iteration.

Hence, the total communication complexity of the algorithm is $O((\kappa + \log n)\sqrt{n \log^7 n})$ bits. Since the length of κ is typically to be the maximum size of the messages [41], it is safe to assume κ is of order $O(\log n)$ for large n. Therefore, the total number of bits used is: $O(\sqrt{n \log^9 n})$. Thus, the message complexity of the algorithm is $O(\sqrt{n \log^7 n})$, since the size of each message is $O(\log n)$[**].   □

Lemma 4. The time complexity of the algorithm is $O(\log^2 n)$ rounds.

Proof. There are $O(\log n)$ candidate nodes, and each may have $O(\log n)$ messages to be sent to its referee nodes in parallel. Thus, it takes $O(\log^2 n)$ rounds, since it takes one round to send a constant number of messages of size $O(\log n)$ bits. The same time bound holds when the referee nodes forward the messages to the candidate nodes. Therefore, the overall round complexity is $O(\log^2 n)$.   □

Thus, we get the following result of the implicit Byzantine agreement with authentication.

Theorem 3.1 (Implicit Agreement). Consider a synchronous, fully-connected network of n nodes and CONGEST communication model. Assuming a public-key infrastructure with the keyed hash function, there exists a randomized algorithm which, with the help of a global coin, solves implicit Byzantine agreement with high probability in $O(\log^2 n)$ rounds and uses $O(n^{0.5} \log^{3.5} n)$ messages while tolerating $f \leq (1/2 - \epsilon)n$ Byzantine nodes under non-adaptive adversary, where ϵ is any fixed positive constant.

Our implicit agreement algorithm can be easily extended to solve explicit agreement (where all the honest nodes must decide on the same value satisfying the validity condition) in one more round. After the implicit agreement, the committee nodes send the agreed value (along with the signature) to all the nodes in the network in the next round. This incurs $O(n \log n)$ messages. Then all the nodes decide on the majority value since the majority of the nodes in the committee are honest. Thus, the following result of explicit agreement follows immediately.

Theorem 3.2 (Explicit Agreement). Consider a synchronous, fully-connected network of n nodes and CONGEST communication model. Assuming a public-key infrastructure with the keyed hash function, there

---

[**]Alternatively, the communication complexity is $O\left(\sqrt{n \log^9 n}\right)$ bits.

exists a randomized algorithm which, with the help of global coin, solves Byzantine agreement with high probability in $O(\log^2 n)$ rounds and uses $O(n \log n)$ messages while tolerating $f \leq (1/2 - \epsilon)n$ Byzantine nodes under non-adaptive adversary, where ϵ is any fixed positive constant.

4   Lower Bound on Message Complexity

We argue a lower bound of $\Omega(\sqrt{n})$ on the number of messages required by any algorithm that solves the authenticated Byzantine agreement under honest majority with high probability. Recall that all the nodes have access to an unbiased global coin. The nodes know the IDs of the other nodes, but are unaware of the port connecting to the IDs. Also, the communication is authenticated.

Our AUTHENTICATED-IMPLICIT-BA algorithm solves multi-valued agreement in polylogarithmic rounds and uses $\tilde{O}(\sqrt{n})$ messages (see, Theorem 3.1). In a non-Byzantine setting, the multi-valued agreement can be used to elect a leader by using the IDs of the nodes as the input values. Thus, any lower bound on the message complexity (and also on the time complexity) of the leader election problem also applies to the multi-valued agreement. Therefore, the $\Omega(\sqrt{n})$ lower bound shown by [8] for the leader election problem using global coin (in the non-Byzantine setting) also applies to our multi-valued Byzantine agreement with global coin. Note that the lower bound holds because of the high success probability requirement of the agreement; otherwise, the leader election solves the agreement with zero message cost, but with only constant success probability. However, the above argument does not hold for the binary agreement, where the input values are either 0 or 1. Below, we argue for the binary case.

Let A be an algorithm that solves the authenticated Byzantine (binary) agreement with constant probability (say, more than $1/2$) under an honest majority with the access of an unbiased global coin and uses only $o(\sqrt{n})$ messages. We show a contradiction. Recall that it is a KT_0 model; so nodes do not know which edge connects to which node-ID. To achieve agreement with constant probability, nodes must communicate with the other nodes; otherwise, if the nodes try to agree locally without any communication, it is likely that there exist two nodes that agree on two different values (this can be easily shown probabilistically). On the other hand, if all the nodes try to communicate, then the message complexity of A would be $\Omega(n)$. So, only a few nodes need to initiate the process. Thus, A must pick these few initiator nodes randomly; otherwise, if picked deterministically, the Byzantine nodes can take over the initiator nodes. The initiator nodes must communicate with the nodes in the network to achieve agreement.

The initiator nodes do not know each other. In the KT_0 setting, for any two nodes to find each other with more than $1/2$ probability requires $\Omega(\sqrt{n})$ messages– inferred from the following lemmas.

Lemma 5. The KT_0 model takes $\Omega(\sqrt{n})$ messages for any two nodes to find each other with more than constant probability.

Proof. Suppose x and y be the two nodes. Both x and y samples $f(n)$ nodes uniformly at random from all the nodes to find out each other. Let E be the event of having a collision in the random samples. Then, for $f(n) < \sqrt{n}$,

$$\Pr[E] = 1 - \left(1 - \frac{f(n)}{n}\right)^{f(n)} = 1 - e^{\frac{-(f(n))^2}{n}} < 1 - 1/e = 0.63 \tag{4}$$

Therefore, x and y cannot find out each other with more than constant probability with $o(\sqrt{n})$ messages. $\qquad \square$

Each initiator node samples some nodes. The initiator and the sampled nodes may exchange messages with the other nodes in the network throughout the execution of A– all these nodes form a connected sub-graph. Let us call this sub-graph as *communication graph* of the initiator node. Thus, for every initiator node, there is a communication graph. Some of them may merge and form a single communication graph.

10

However, it is shown in [8] that if the algorithm A sends only $o(\sqrt{n})$ messages, then there exist two disjoint communication graphs w.h.p. (see, Section 2 of [8]). Since the nodes do not know each other (in KT_0), the communication graphs have similar information. The global coin also gives the same information to all the nodes. Since the communication graphs are disjoint, no information is exchanged between them. A formal proof of this argument can be found in [32, 8].

Let H_u and H_v be the two disjoint communication graphs corresponding to the two initiators u and v respectively. We show that H_u and H_v agree with opposite decisions, i.e., if H_u decides on 0 then H_v decides on 1 and vice versa.

Lemma 6. The nodes of H_u and H_v agree with opposing decisions.

Proof. The nodes in H_u and H_v are random since the network is anonymous and no information is known before contacting a node. Thus, the same $(1/2 - \epsilon)$ fraction of Byzantine nodes (as in the original graph G) are present in both H_u and H_v in expectation. Now consider an input distribution $\mathcal{I}$, in which each node in the graph G is given an input value 0 and 1 with probability $1/2$. Then in expectation, half of the honest nodes in H_u (and also in H_v) gets 0 and the other half gets 1. We argue that the two disjoint sets of nodes in H_u and H_v decide on two different input values under the input distribution $\mathcal{I}$. Recall that the nodes in H_u have the same information as the nodes in H_v. Further, they run the same algorithm A. The Byzantine adversary (which controls the Byzantine nodes) knows the algorithm and also the input values of the honest nodes in H_u and H_v. Since the number of Byzantine nodes in each of the H_u and H_v is almost half of their size, and $0 - 1$ distribution among the honest nodes is almost 50-50, the Byzantine nodes can control the output of the agreement in the two groups— H_u and H_v. Suppose the Byzantine nodes in H_u exchange some input bits with honest nodes to agree on 0 in H_u, then the Byzantine nodes in H_v must exchange opposite bits to agree on 1 in H_v. Thus, H_u and H_v agree with opposing decisions. □

The above lemma contradicts the assumption that A solves the Byzantine (binary) agreement with only $o(\sqrt{n})$ messages. The global coin does not help, as it gives the same information to all the nodes. Thus, we get the following result on the lower bound of the message complexity.

Theorem 4.1. Consider any algorithm A that has access to an unbiased global coin and sends at most $f(n)$ messages (of arbitrary size) with high probability on a complete network of n nodes. If A solves the authenticated Byzantine agreement under honest majority with more than $1/2$ probability, then $f(n) \in \Omega(\sqrt{n})$.

Note that the lower bound holds for the algorithms in the *LOCAL* model, where there is no restriction on the size of a message that can be sent through edges per round [43].

5   Conclusion and Future Work

We studied one of the fundamental problem in distributed networks, namely Byzantine agreement. We showed that implicit Byzantine agreement can be solved with sublinear message complexity in the honest majority setting with the help of cryptographic set up of PKI and hash function, and access to a global coin. The bound is also optimal up to a $\text{polylog } n$ factor. To the best of our knowledge, this is the first sublinear message bound result on Byzantine agreement.

A couple of interesting open problems are: (i) is it possible to achieve a sublinear message complexity Byzantine agreement algorithm without the global coin or hash function in this setting? (ii) whether a sublinear message bound is possible under adaptive adversary, which can take over the Byzantine nodes at any time during the execution of the algorithm?

References

[1] Ittai Abraham, TH Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *PODC*, 2019.

[2] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience. In *FC 2019*.

[3] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. In *OPODIS 2017*.

[4] A. Agbaria and R. Friedman. Overcoming byzantine failures using checkpointing. *UILU-ENG- 03-2228 (CRHC-03-14)*, 2003.

[5] Yair Amir, Claudiu Danilov, Jonathan Kirsch, John Lane, Danny Dolev, Cristina Nita-Rotaru, Josh Olsen, and David John Zage. Scaling byzantine fault-tolerant replication towide area networks. In *DSN*, 2006.

[6] D. P. Anderson and J. Kubiatowicz. The worldwide computer. *Scientific American*, 286(3):28–35, 2002.

[7] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd edition)*. John Wiley Interscience, 2004.

[8] John Augustine, Anisur Rahaman Molla, and Gopal Pandurangan. Sublinear message bounds for randomized agreement. In *PODC*, pages 315–324, 2018.

[9] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC, 1988*.

[10] Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. Byzantine agreement in the full-information model in o(log n) rounds. In *STOC*, pages 179–186. ACM, 2006.

[11] Bitcoin. Bitcoin website https://bitcoin.org/.

[12] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *PODC,2013*.

[13] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *TOCS*, 20(4):398–461, 2002.

[14] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[15] Ethereum. Ethereum website https://ethereum.org/.

[16] Paul Feldman and Silvio Micali. Byzantine agreement in constant expected time (and trusting no one). In *FOCS*, pages 267–276, 1985.

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

[18] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[19] Matthias Fitzi. Generalized communication and security models in byzantine agreement. *PhD Dissertation*, 2002.

[20] Juan A. Garay and Yoram Moses. Fully polynomial byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998.

[21] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, pages 51–68, 2017.

[22] Seth Gilbert and Dariusz R. Kowalski. Distributed agreement with optimal communication complexity. In *SODA*, pages 965–977, 2010.

[23] Shafi Goldwasser, Elan Pavlov, and Vinod Vaikuntanathan. Fault-tolerant distributed computing in full-information networks. In *FOCS*, pages 15–26, 2006.

[24] Jim Gray. The cost of messages. In *PODC*, pages 1–7. ACM, 1988.

[25] Vassos Hadzilacos and Joseph Y. Halpern. Message-optimal protocols for byzantine agreement. *Mathematical Systems Theory*, 26(1):41–102, 1993.

[26] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, 2009.

[27] Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. *J. ACM*, 58(4):18:1–18:24, 2011.

[28] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium, USENIX Security 16, 2016*.

[29] Jiejun Kong. *Anonymous and untraceable communications in mobile wireless networks*. Citeseer, 2004.

[30] Dariusz R. Kowalski and Achour Mostéfaoui. Synchronous byzantine agreement with nearly a cubic number of communication bits: synchronous byzantine agreement with nearly a cubic number of communication bits. In *PODC*, pages 84–91. ACM, 2013.

[31] Manish Kumar and Anisur Rahaman Molla. Brief announcement: On the message complexity of fault-tolerant computation: Leader election and agreement. In *PODC '21*.

[32] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. Sublinear bounds for randomized leader election. *Theor. Comput. Sci.*, 561:134–143, 2015.

[33] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[34] Na Li, Nan Zhang, Sajal K Das, and Bhavani Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 2009.

[35] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[36] Dahlia Malkhi and Michael K. Reiter. Unreliable intrusion detection in distributed computations. In *CSFW*, pages 116–125, 1997.

[37] Silvio Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016. URL: http://arxiv.org/abs/1607.01341.

[38] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130. IEEE Computer Society, 1999.

[39] Silvio Micali and Vinod Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. 2017.

[40] Michael Mitzenmacher and Eli Upfal. Probability and computing: Randomized algorithms and probabilistic analysis. 2004.

[41] Atsuki Momose and Ling Ren. Optimal communication complexity of byzantine consensus under honest majority. *Journal of Environmental Sciences (China) English Ed*, 2020.

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

[43] David Peleg. Distributed computing: A locality-sensitive approach. 2000.

[44] Michael O. Rabin. Randomized byzantine generals. In *FOCS*, pages 403–409, 1983.

[45] Sean C. Rhea, Patrick R. Eaton, Dennis Geels, Hakim Weatherspoon, Ben Y. Zhao, and John Kubiatowicz. Pond: The oceanstore prototype. In *FAST*, pages 1–14, 2003.

[46] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Commun.*, 11(6):38–43, 2004.

[47] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer networks*, 76:146–164, 2015.

[48] Rolf H Weber. Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1):23–30, 2010.

[49] Alex Wright. Contemporary approaches to fault tolerance. *Commun. ACM*, 52(7):13–15, 2009.

[50] Hiroyuki Yoshino, Naohiro Hayashibara, Tomoya Enokido, and Makoto Takizawa. Byzantine agreement protocol using hierarchical groups. In *ICPADS*, 2005.