



Composable Byzantine Agreements with Reorder Attacks

Jing Chen* 



Department of Computer Science and Technology, Tsinghua University, Beijing, China

Jin Dong 

Beijing Academy of Blockchain and Edge Computing (BABEC), China

Jichen Li   

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Xuanzhi Xia  

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Wentao Zhou  

Department of Computer Science and Technology, Tsinghua University, Beijing, China

Abstract

Byzantine agreement (BA) is a foundational building block in distributed systems that has been extensively studied for decades. With the growing demand for protocol composition in practice, the security analysis of BA protocols under multi-instance executions has attracted increasing attention. However, most existing adversary models focus solely on party corruption and neglect important threats posed by adversarial manipulations of communication channels in the network. Through channel attacks, messages can be reordered across multiple executions and lead to violations of the protocol's security guarantees, without the participating parties being corrupted.

In this work, we present the first adversary model that combines party corruption and channel attacks. Based on this model, we establish new security thresholds for Byzantine agreement under parallel and concurrent compositions, supported by complementary impossibility and possibility results that match each other to form a tight bound. For the impossibility result, we show that even authenticated Byzantine agreement protocols cannot be secure under parallel composition when $n \leq 3t$ or $n \leq 2c + 2t + 1$, where t and c denote the number of corrupted parties and communication channels, respectively. For the possibility result, we prove the existence of secure protocols for unauthenticated Byzantine agreement under parallel and concurrent composition, when $n > 3t$ and $n > 2c + 2t + 1$. More specifically, we provide a general black-box compiler that transforms any single-instance secure BA protocol into one that is secure under parallel executions, and we provide a non-black-box construction for concurrent compositions.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Security and privacy → Security protocols

Keywords and phrases Byzantine agreement, protocol composition, channel reorder attack, security threshold

Digital Object Identifier 10.4230/LIPIcs.AFT.2025.13

Funding This work is partially supported by Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing.

Acknowledgements The authors would like to thank several anonymous reviewers for their valuable comments.

*Corresponding author.



© Jing Chen, Jin Dong, Jichen Li, Xuanzhi Xia, and Wentao Zhou;
licensed under Creative Commons License CC-BY 4.0
7th Conference on Advances in Financial Technologies (AFT 2025).

Editors: Zeta Avarikioti and Nicolas Christin; Article No. 13; pp. 13:1–13:23



Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The Byzantine agreement (BA) problem [46] is a foundational challenge in distributed systems, concerned with achieving consensus among parties even in the presence of failures or malicious behavior. This problem has been extensively studied under various models, leading to significant theoretical breakthroughs, including both impossibility results [43, 26] and the development of innovative protocols [24, 9, 47, 39]. As a core mechanism for achieving distributed consistency and fault tolerance, Byzantine agreement underpins a wide range of applications, including blockchain technologies [12, 1, 48, 55, 41, 56], secure multiparty computation [54, 35, 4, 11, 17, 28, 33] and diverse distributed services [30, 45, 2].

Much of the early research on the Byzantine agreement problem was conducted under a classical model where a network of n parties, each holding an initial input value, communicates over reliable synchronous channels. In this model, faults are restricted to the parties themselves: up to t parties may be corrupted by an adversary, but the communication network is assumed to be secure – messages cannot be forged, altered, or dropped by the adversary once sent by an honest party. The standard goals of a BA protocol in this setting are: (1) *agreement* – all honest parties eventually terminate and output the same value; and (2) *validity* – if all honest parties start with the same input, that value is the output. Studies show that in unauthenticated settings, BA is achievable if and only if $t < \frac{n}{3}$ [43, 46], while in authenticated settings, where digital signatures prevent forgery [50, 36], BA can be achieved with any number of corruption $t < \frac{n}{2}$ [27]. The expected constant-round protocols with optimal resiliency in the two settings were then constructed in [23] and [39], respectively.

However, the aforementioned studies primarily focus on single-instance executions of Byzantine protocols, leaving the critical issue of protocol composition unaddressed. In modern distributed environments, such as sharded blockchains, multiparty computation frameworks, or cross-chain settings, multiple consensus protocols may be composed sequentially, in parallel, or concurrently. In sequential composition, each protocol instance begins only after the previous one has been completed. In parallel composition, all instances are initiated at the same time and proceed with their steps aligned with each other. The most general model, concurrent composition, grants the adversary full control over the start times and execution rates of different instances.

The shift from single-instance to multi-instance execution introduces new problems, demanding a re-examination of existing assumptions about fault models and protocol security. In particular, [44] showed that, without a unique and common session identifier for every execution of the protocol, no authenticated BA protocol can remain secure even under just two parallel executions when corruption exceeds $n/3$. In other words, authenticated BA performs as poorly as unauthenticated BA under parallel composition. Common session IDs can sometimes be achieved via a bootstrap phase, but in this work we would like to pursue the power of stateless composition without relying on such a bootstrap, so that the resulting BA protocols can be directly applied whenever compositional executions are needed.

The challenge with protocol composition is that the adversary’s power is amplified, and in our study we will amplify it even further. Indeed, a key limitation of the classical model is its implicit assumption that adversarial power is confined to corrupting parties, while communication channels remain trustworthy in the sense that messages sent between honest parties will be received correctly. This abstraction overlooks more realistic adversarial behaviors, such as man-in-the-middle or message-reordering attacks, which target the network rather than the parties themselves, especially when protocol composition is concerned. By carefully disrupting or manipulating some communication channels between multiple protocol instances, an adversary may induce cross-protocol interference, leading to violations of agreement or validity that would not occur in isolated executions.

1.1 Adversary Model

To address the limitations of classical adversary models, we introduce an extended adversarial framework tailored to protocol composition environments. In this model, the adversary retains the classical ability to corrupt up to t parties, allowing them to deviate arbitrarily from the protocol. In addition, channels can also be corrupted, without sending and receiving parties being corrupted or even aware of the channel attack. This reflects realistic attack approaches in networks where multiple protocols share the underlying network infrastructure.

In the real world, the adversary may exploit man-in-the-middle attacks, such as ALPACA [7], to redirect the messages and bypass the security guarantees of TLS and application layer protocols, resulting in cross-protocol attacks that rearrange the messages from different protocols without breaking the signature scheme.

Such a setting captures the inherent interdependence of parallel or concurrently running consensus protocols, where messages from different instances may traverse overlapping physical or logical channels. Through reorder attacks, the adversary can introduce inter-instance inconsistencies that undermine global agreement guarantees, even if each protocol remains individually secure under classical assumptions.

This hybrid adversary model – combining party corruption with channel attacks – introduces new theoretical challenges. In particular, these two kinds of attacks are coupled together, thus determining the exact resilience thresholds under simultaneous control of t parties and c channels requires a fresh analytical approach that goes beyond traditional BA models.

1.2 Our Results

Our main results establish a new tight security threshold under the new adversary model. We first present the impossibility result showing that even in the authenticated setting, BA under parallel (and concurrent) composition is not achievable if $n \leq 3t$ or $n \leq 2c + 2t + 1$, where t is the number of corrupted parties and c is the number of channels that the adversary can manipulate.

Intuitively, even if a majority of parties are honest, agreement may still fail if the adversary can confuse one party, making it unable to tell which specific protocol instance it is participating in. For example, when two protocols are running in parallel, an attacker can target a particular party and swap the messages it is supposed to receive in the two protocols, causing its result to differ from that of the other honest parties. Therefore, it is not enough to simply bound the number of corrupted parties – one must also ensure that the honest parties maintain a sufficient number of communication with parties in the same protocol. This leads to a trade-off between party corruption and channel interference, and motivates the combined threshold $n > 2c + 2t + 1$ as the condition necessary to preserve the communication advantage of the honest majority.

We further prove possibility results under compositional executions, showing that the condition $n > 2c + 2t + 1$ along with the condition $n > 3t$ in classical setting (which is the tight bound when only party corruption is considered) is sufficient even in the unauthenticated setting. That is, when the combined adversarial power satisfies both conditions, it is possible to design a BA protocol that remains secure under arbitrary parallel and concurrent executions. These results complement our impossibility result, forming a tight security threshold: $n > \max\{2c + 2t + 1, 3t\}$ is both necessary and sufficient for protocol composition in our adversary model.

Noticeably, for parallel composition, we provide a general black-box compiler that transforms any single-instance secure BA protocol into one that is secure under parallel executions. A non-black-box construction for concurrent composition is provided, which is secure under asynchronous networks. By carefully structuring message sending and leveraging the honest majority's residual communication connectivity, our protocol prevents cross-instance interference and ensures that no single execution is compromised due to shared network vulnerabilities.

In conclusion, our work provides the first formal characterization of security for BA in the presence of combined party corruption and channel attacks under protocol composition, filling a critical gap in the theoretical understanding of consensus under adversarial network interference. By introducing a unified adversary model and establishing the tight security threshold, we extend the classical BA framework to more accurately reflect the complication of modern distributed systems.

2 Related Work

Composition of Byzantine Agreements. In a stateless model where protocols do not have distinguishing IDs, which is also considered by our work, [44] studied the sequential, parallel, and concurrent composition of authenticated BA protocols and presented the impossibility result as mentioned in the introduction. It also constructed secure randomized protocols under sequential composition. Without cryptographic primitives, [3] considered concurrently secure BAs with expected-constant round in both synchronous and asynchronous networks with optimal resiliency in the classical model (i.e., $n > 3t$). However, as pointed out by [16], its security proof had some subtle issues regarding, e.g., the use of oblivious leader election.

Both [3] and [16] are not stateless: they assume a unique and common session ID for every execution of the protocol. As pointed out by [44], concurrent composition of any number of executions in this case is possible. Indeed, the focus of [3] and [16] was to construct expected-constant round protocols that are concurrently composable. Similarly, [37] assumed unique session IDs but considered an adversary who can corrupt a player in some but not all parallel executions.

Moreover, [16] followed the Universal Composability (UC) framework [8], which means the protocol is secure under arbitrary composition with itself and other cryptographic protocols in an adversarially controlled manner. Focusing on the asynchronous setting, [16] considered concurrent BA and presented the first information-theoretic multi-valued oblivious common coin (OCC) protocol with optimal resiliency. It further provided a modularized protocol for round-preserving parallel composition of BA that was simpler than the construction in [3].

Computation and Communication Complexity. An important line of work considers the complexity of a single execution of a Byzantine agreement. A body of foundational studies, in particular [19, 20, 9], proposed Byzantine agreement protocols that required a quadratic number of messages. More specifically, the $O(n^2)$ communication complexity necessitates nearly all-to-all communication between parties. Subsequently, for *randomized Byzantine protocols*, a seminal work [40] proposed a protocol where each party speaks to only $\tilde{O}(1)$ other parties, though their protocol achieved almost-everywhere agreement [21] rather than agreement, where $1 - O(\log^{-1} n)$ fraction of the parties reach agreement. Subsequent studies have sought to bridge this gap by achieving agreement from almost-everywhere agreement. To the best of our knowledge, so far the best communication complexity achievable for agreement in authenticated setting is $\tilde{O}(1)$ rounds with $\tilde{O}(\sqrt{n})$ communication and computation [32]. With enhanced cryptographic primitives such as LWE, this can be improved to $\tilde{O}(1)$ rounds and $\text{poly}(\lambda, \log n)$ communication [5, 25], where λ is the security parameter.

However, for *deterministic Byzantine protocols*, [18] established that achieving agreement inherently requires at least $\Omega(t^2)$ communication. Furthermore, [14] extended this lower bound to other generalized validity definitions (e.g., weak validity), demonstrating that any well-defined Byzantine agreement variant necessitates $\Omega(t^2)$ communication. Thus, deterministic Byzantine agreements incur fundamentally more overhead than randomized ones.

Different from this line of works that focused on single-instance executions of BA protocols, our work investigates compositional executions of BA protocols. Although our protocols are not necessarily optimal in terms of their complexity, our black-box compiler protocol achieves polynomial communication and computation. Further improving the complexity of our protocols and proving complexity lower-bounds in our model are interesting open problems.

Network Attacks. Many prior works on network security have shown that channel attacks (e.g., man-in-the-middle attacks) without corrupting participating parties exist in a wide range of scenarios and pose a significant threat in real applications. By delaying, forging, dropping, or redirecting, attackers can break SSL/TLS certificate validation in many critical software applications [34, 51], cause security issues for HTTPS and its certificate trust model [15], inflict devastating damage on private and consortium blockchains [22], and so on.

Finally, it's worth noticing that for transaction messages in DeFi, attackers (miners) can gain Miner Extracted Value (MEV) through transaction reordering attacks such as those studied in [53], while [38] indicated that currently no mitigation schemes can fully solve this problem. For a fully connected large-scale network, [52] analyzed Simple Majority Protocol (SMP) under probabilistic message loss and proved that it can reach consensus in three rounds of communication with probability approaching 1. However, those are within the same consensus protocol and the role of channel attacks in the composition of Byzantine agreements has not been explored yet.

3 Our Model

3.1 Byzantine Agreement

We consider the problem of *Byzantine Agreement* where $P = \{P_1, \dots, P_n\}$ is a set of n parties, and P is common knowledge among all parties. The protocol Π is executed m times in parallel (or concurrently), where we denote the k -th instance by Π_k . Each party P_i in each instance Π_k is formally modeled by an interactive (randomized) Turing machine ITM_k^i , with an input tape containing its initial value, an output tape for its final output, a random tape, and $n - 1$ pairs of communication tapes (input/output) corresponding to the other $n - 1$ parties in the same Π_k , denoted by $Input_{i,j}^k$ and $Output_{i,j}^k$ for each $P_j \neq P_i$. More specifically, $Output_{i,j}^k$ contains messages that i sends to j , and $Input_{i,j}^k$ contains messages that i receives from j . Under parallel or concurrent composition, a party locally runs m copies of such interactive Turing machines simultaneously, each corresponding to a particular protocol instance.

Network and Communication. In most parts of the paper except Section 6, the network is synchronous and point-to-point: communication proceeds in rounds, each consisting of a *send* phase followed by a *receive* phase. In the send phase, a party writes the sending message onto the corresponding communication output tape. Then, in the receive phase, the message is written onto the target party's input tape under the same protocol instance.

More specifically, the communication channel between parties P_i and P_j , denoted by C_{ij} , is the set of all input and output tapes between the two parties, for all m protocol instances. By symmetry, C_{ji} means the same thing as C_{ij} and an attack on one is also an attack on the other. In an uncompromised channel C_{ij} , messages originating from $Output_{i,j}^k$ are delivered exclusively to $Input_{j,i}^k$ for all $k \in \{1, \dots, m\}$, and vice versa.

Stateless Setup. In stateless compositions of BA protocols, there is no common session identifier for the protocol instances. The indices of different protocol instances are for discussion purposes and are not accessible by the parties. Indeed, each Turing machine ITM_k^i has access to its tapes but doesn't know the global index k . Internally, each party may have individual numbering for its Turing machines, but those numbers may not be consistent among different parties.

For authenticated Byzantine Agreement (ABA), a *common setup* is shared across all protocol instances. In particular, all parties undergo a one-time trusted preprocessing phase that generates cryptographic primitives, which are then used consistently across all protocol instances. Formally, each interactive Turing machine has a read-only *setup tape*, and all Turing machines run by the same party share this setup tape. For example, this tape may contain the secret key of a signature scheme used by the party, as well as the public verification keys of other parties. For unauthenticated Byzantine Agreement, no cryptographic primitives are used and there is no setup tape or a setup phase.

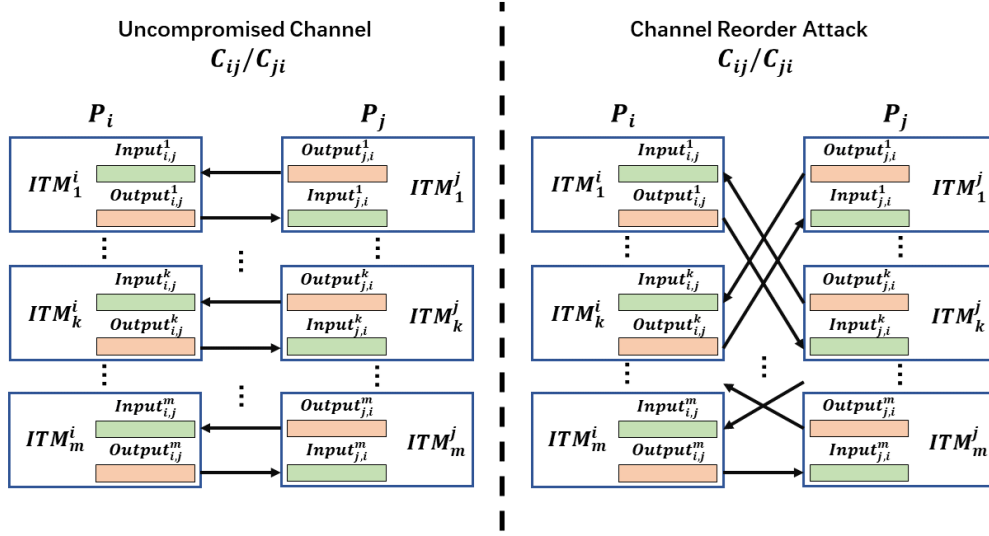
Composition of BA Protocols. We consider two types of protocol composition: parallel composition and concurrent composition. In parallel composition, all protocols start simultaneously, and each round has one time step across all protocol instances. In concurrent composition, the adversary can independently control the start time and the time steps of each round of each protocol instance.

Adversarial Model. We consider an adversary $\mathcal{A}(t, c)$ operating in the *point-to-point full information* setting in a parallel or concurrent execution of protocol instances $\Pi_1, \Pi_2, \dots, \Pi_m$. Its capabilities include:

- *Corruption:* The adversary can corrupt up to t parties and chooses which parties to corrupt before any protocol instance begins. This is called static corruption.
- *Channel Reorder Attack:* The adversary can attack up to c channels between honest parties. When C_{ij} is under attack, the adversary is allowed to redirect messages on the corresponding input tapes for different protocol instances and in both directions. See below for a formal definition, which is also illustrated in Figure 1.

► **Definition 1 (Channel Reorder Attack).** *For any two honest parties P_i and P_j , a channel reorder attack on C_{ij} allows the adversary to arbitrarily redirect messages between different protocol instances and in both directions. Specifically, the adversary can deliver any message $msg \in Output_{x,y}^{k_1}$ to any $Input_{y,x}^{k_2}$, where $k_1, k_2 \in \{1, \dots, m\}$ and $\{x, y\} = \{i, j\}$.*

- *Rushing:* The adversary is rushing, meaning it can observe all messages sent by honest parties in a round before deciding the messages to send from corrupted parties and message reordering in the same round.
- *Computation Power:* For our impossibility results, which apply to authenticated BA protocols, we assume that all parties (honest and adversarial) are probabilistic polynomial-time (PPT). In contrast, for our positive results, which apply to unauthenticated BA protocols, we allow computationally unbounded adversaries.



■ **Figure 1** Communication diagrams for channel C_{ij}/C_{ji} , with and without channel reorder attack.

For this extended adversary model, a protocol Π is considered a solution to the BA problem if it satisfies the following properties:

► **Definition 2** (Byzantine Agreement). Let $\mathcal{A}(t, c)$ be an adversary who can corrupt up to t parties and c channels. A protocol Π solves the Byzantine Agreement problem if for any adversary $\mathcal{A}(t, c)$, the following two properties hold:

- Agreement: All honest parties eventually terminate and output the same value.
- Validity: If all honest parties begin with the same initial value v , then their output values must all be v .

3.2 Composition Security

Composition security plays a pivotal role in analyzing the robustness of cryptographic protocols executed within larger systems or concurrently with other protocol instances, rather than in isolation. Our work adopts the definition of composition security as presented in [44], with the notable distinction that we explicitly consider attacks on communication channels.

► **Definition 3** (Composition Security of ABA). Let P_1, \dots, P_n be parties for an ABA protocol Π . We say that Π remains secure under m instances of parallel (or concurrent) executions, if for every PPT adversary $\mathcal{A}(t, c)$, the requirement of BA holds for every individual instance of Π within the following procedure:

1. Setup Phase: A single, global trusted setup phase is performed once. This phase generates setup strings s_1, \dots, s_n for parties P_1, \dots, P_n , which are then consistently used across all m protocol instances.
2. Static Corruption: The adversary $\mathcal{A}(t, c)$ statically corrupts up to t parties, gaining full control over their actions. Additionally, the adversary can attack up to c communication channels C_{ij} between honest parties, enabling channel reorder attacks described in Definition 1.
3. Parallel (Concurrent) Executions: The following procedure is repeated in parallel (or concurrently) for each protocol instance, until the adversary halts:
 - The adversary specifies the initial input values for all parties.
 - Each party uniformly generates the content of its random tape for the execution in this protocol instance.

- All parties are invoked for the execution of Π .
- The adversary determines the messages sent by the corrupted parties, while honest parties strictly adhere to the protocol Π .
- Messages transmitted over attacked channels are subject to the adversary's reordering on corresponding input tapes, whereas messages on unattacked channels between honest parties are delivered reliably.

For unauthenticated Byzantine Agreement protocols, the definition of composition security is analogous, with the difference that no setup phase or cryptographic primitives are needed.

4 Impossibility Result

Authentication allows protocols to achieve higher fault tolerance in the stateless execution model, tolerating an arbitrary number of corrupted parties. However, this intuition breaks down in composed scenarios. When multiple executions occur, a signature from a Party P on message x does not prove that P signed x in the “current” specific execution. An adversary can “borrow” signed messages from one execution and use them in another, rendering the public-key infrastructure “useless” in distinguishing execution contexts.

► **Theorem 4.** *There does not exist a protocol for stateless authenticated Byzantine Agreement in a synchronous network with n parties that remains secure under parallel composition (even for just two executions) against an adversary $\mathcal{A}(t, c)$, provided that (1) $n \leq 3t$, or (2) $n \leq 2c + 2t + 1$.*

Proof. For condition (1), the result follows directly from Theorem 1 in [44].

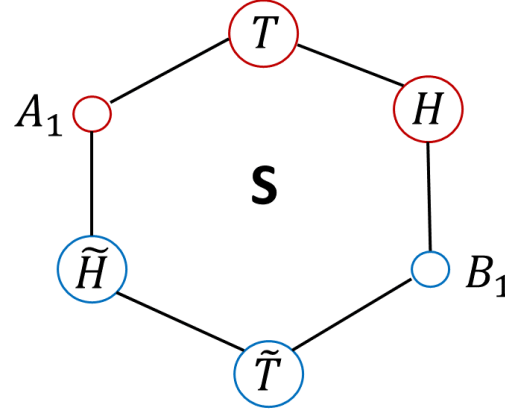
We prove the result for condition (2) by contradiction. Assume that there exists an authenticated Byzantine Agreement protocol Π that remains secure under two parallel executions against an adversary $\mathcal{A}(t, c)$ with $n \leq 2c + 2t + 1$. We assume $c > 0$, otherwise we have $2t + 1 \geq n$ and it can be reduced to condition (1).

We consider two independent executions of Π : Let A_1, A_2, \dots, A_n and B_1, \dots, B_n be independent copies of n parties participating in protocol Π . The independent copies mean that for each i , A_i and B_i are the same party that runs in two different parallel executions of Π . For the purpose of the proof below, we denote the set $T = \{A_2, \dots, A_{\lfloor \frac{n+1}{2} \rfloor}\}$, $\tilde{T} = \{B_2, \dots, B_{\lfloor \frac{n+1}{2} \rfloor}\}$, $H = \{A_{\lfloor \frac{n+3}{2} \rfloor}, \dots, A_n\}$, and $\tilde{H} = \{B_{\lfloor \frac{n+3}{2} \rfloor}, \dots, B_n\}$. Note that we have $n \leq 2c + 2t + 1$, so we have $c + t \geq |T| = |\tilde{T}|$ and $c + t \geq |H| = |\tilde{H}|$.

Let $\text{rounds}(\Pi)$ denote the maximum number of communication rounds required for protocol Π to achieve termination in any execution. By the termination guarantees of the agreement property of Byzantine Agreement protocols, $\text{rounds}(\Pi)$ is finite and well-defined. Furthermore, given the security assumption that Π remains secure under parallel composition, it follows that Π must terminate within $\text{rounds}(\Pi)$ rounds even when executed concurrently across multiple protocol instances.

We now introduce an important abstract concept and will instantiate it in different ways in the proof. A system X is a tuple:

- A set $P = \{P_1, \dots, P_n\}$ of n parties, each party P_j runs two copies of ITMs, A_j and B_j , for protocol Π ;
- An adversary $\mathcal{A}(t, c)$ controlling t corrupted parties and reordering c channels;
- Initial input values for all ITMs on their input tapes;
- A network topology governing inter-party connectivity.



■ **Figure 2** Network topology of System S .

Intuitively, if the system contains a non-trivial adversary who corrupts some parties and reorders some channels, then the network topology is consistent with the adversary's reordering of the channels. In some mental games, we may instantiate the system without an adversary, in which case the network topology dictates how the parties' communication channels are inter-connected crossing the two executions of Π .

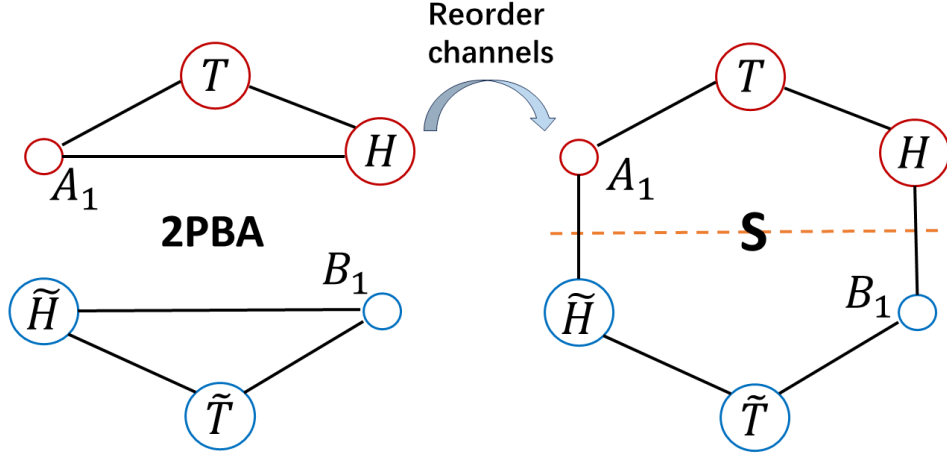
For any ITM M run by a party in P , its *view* in system X , denoted by $\text{view}_X(M)$, consists of the content of M 's input tapes and random tape, where the former in particular includes the initial input value and the messages received by M from other parties during the execution of X .

When all parties (including $\mathcal{A}(t, c)$) behave deterministically, $\text{view}_X(M)$ is uniquely determined by the initial configurations of the tapes and the adversarial behavior. Conversely, under randomized executions (where the content of the random tapes are sampled uniformly at random), $\text{view}_X(M)$ constitutes a random variable over the corresponding probability space.

We now instantiate the system S through the following configuration:

- **Network Topology:** ITM A_1 establishes connectivity with the set \tilde{H} (replacing its original link to H), while ITM B_1 interfaces with H instead of \tilde{H} . This structural reconfiguration is illustrated in Figure 2.
- **Initial Input Value:**
 - ITMs in $\{A_1\} \cup T \cup H$ receive input 0;
 - ITMs in $\{B_1\} \cup \tilde{T} \cup \tilde{H}$ receive input 1.
- **Adversary:** In this system, there is no adversary; formally, we have $t = 0$ and $c = 0$.
- **Protocol Execution:** All ITMs in S strictly adhere to the instruction of protocol Π , simulating an *authenticated Byzantine Agreement* environment. Specifically, each party:
 - Generates messages as prescribed by Π 's honest execution semantics;
 - Sends these messages via the network topology defined above.

Now we first state the following lemmas and we will prove them afterwards.



■ **Figure 3** Party's view is identical between System 2PBA and System S.

► **Lemma 5.** *In system S, all ITMs in $\{A_1\} \cup T$ output 0, and there exists at least one ITM in H outputs 0. Symmetrically, all ITMs in $\{B_1\} \cup \tilde{T}$ output 1, and there exists at least one ITM in \tilde{H} outputs 1.*

► **Lemma 6.** *In system S, all ITMs in $\{A_1\} \cup \tilde{H}$ and at least one ITM in \tilde{T} output the same value; Symmetrically, all ITMs in $\{B_1\} \cup H$ and at least one ITM in T outputs the same value.*

By combining Lemma 5 with Lemma 6 we have $Output_S(\tilde{H}) = Output_S(A_1) = 0$ and at least one ITM $\tilde{h} \in \tilde{H}$ satisfies $Output_S(\tilde{h}) = Output_S(B_1) = 1$. Hence, we derive a contradiction on the output of \tilde{h} , thereby completing the proof of the theorem. ◀

Next, we complete the proofs of Lemma 5 and Lemma 6.

Proof of Lemma 5. To prove the lemma, we define a new system 2PBA as follows and as shown in Figure 3:

- **Network Topology:** The system consists of two parallel executions of protocol Π , denoted Π_0 (consisted of A_1 , T and H) and Π_1 (consisted of B_1 , \tilde{T} and \tilde{H}).
- **Initial Input Value:**
 - ITMs in $\{A_1\} \cup T \cup H$ receive input 0;
 - ITMs in $\{B_1\} \cup \tilde{T} \cup \tilde{H}$ receive input 1.
- **Adversary:** The adversary $\mathcal{A}(t, c)$ chooses an arbitrary subset of parties $\{P_{\lfloor \frac{n+3}{2} \rfloor}, \dots, P_n\}$, denoted by C , which satisfies $|C| = \min\{c, n - \lfloor \frac{n+3}{2} \rfloor + 1\}$. Then the adversary reorders all the channels between C and P_1 , and corrupts the parties in $\{P_{\lfloor \frac{n+3}{2} \rfloor}, \dots, P_n\} \setminus C$. As we have $2c + 2t + 1 \geq n$, the adversary can indeed corrupt all these parties. In the following proof, we denote $ITM_0^C = \{ITM_0^j : j \in C\}$ as the set of ITMs run by C in the protocol Π_0 , and as the same, we denote $ITM_1^C = \{ITM_1^j : j \in C\}$.
- **Protocol Execution:**
 - All honest parties in S strictly adhere to the instruction of protocol Π ;
 - The corrupted parties reorder their channels between P_1 , and besides this action, they adhere to the instruction of protocol Π .

We then prove $\text{view}_{2PBA}(A_1) = \text{view}_S(A_1)$, $\text{view}_{2PBA}(T) = \text{view}_S(T)$, and $\text{view}_{2PBA}(H) = \text{view}_S(\tilde{H})$.

We proceed by induction on the round counter $j \geq 1$.

Base Case ($j = 1$).

- *Message Generation:* All parties in both 2PBA and S generate identical message sets due to protocol Π strictly.
- *Message Routing:*
 - In S: Messages propagate through the hexagonal graph.
 - In 2PBA: The adversary $\mathcal{A}(t, c)$ enforces the following redirections:

$$\begin{aligned} A_1 &\rightarrow H \xrightarrow{\text{reorder}} \tilde{H}, & \tilde{H} &\rightarrow B_1 \xrightarrow{\text{reorder}} A_1, \\ A_1 &\rightarrow T, & T &\rightarrow H \\ H &\rightarrow T, & T &\rightarrow A_1 \\ H &\rightarrow A_1 \xrightarrow{\text{reorder}} B_1, & B_1 &\rightarrow \tilde{H} \xrightarrow{\text{reorder}} H. \end{aligned}$$

After reordering, B_1 communicates with H rather than \tilde{H} , A_1 communicates with \tilde{H} rather than H . Other than that, all the ITMs strictly adhere to the instruction of Π . It is easy to see that the actual message routing is the hexagonal graph the same as in S.

- *View Equivalence:* The above routing rules induce identical message distributions at all parties' interfaces. Thus, $\text{view}_{2PBA}(A_1) = \text{view}_S(A_1)$, $\text{view}_{2PBA}(T) = \text{view}_S(T)$, and $\text{view}_{2PBA}(H) = \text{view}_S(\tilde{H})$.

Inductive Step. Assume equivalence holds through step j . For step $j + 1$:

- *Message Generation:* Identical message distributions emerge from equivalent historical views (by inductive hypothesis).
- *Message Routing:* The hexagonal routing in S and adversarial redirection in 2PBA preserve the same correspondence as $j = 1$.
- *View Update:* Therefore, $\text{view}_{2PBA}^{(j+1)}(A_1) = \text{view}_S^{(j+1)}(A_1)$ and $\text{view}_{2PBA}^{(j+1)}(T) = \text{view}_S^{(j+1)}(T)$ maintain for the $j + 1$ rounds.

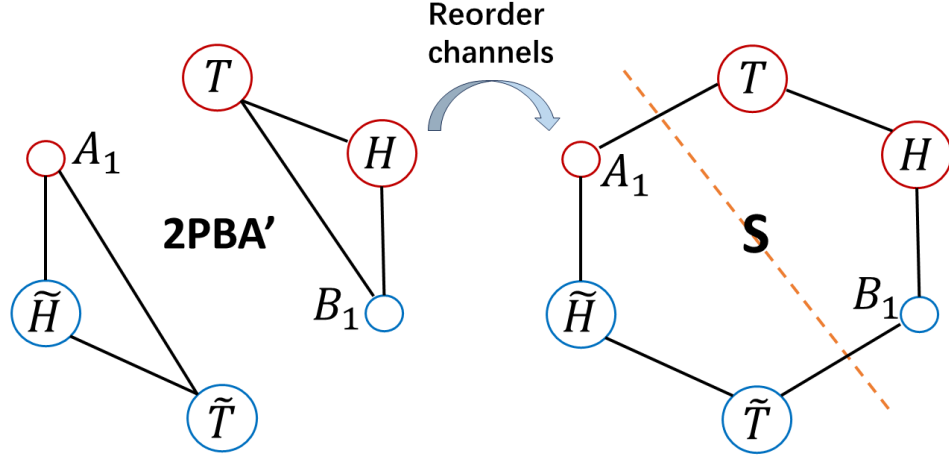
Owing to the termination property of Byzantine Agreement, which mandates that the protocol Π halts within $\text{rounds}(\Pi)$, the aforementioned procedure completes in at most $\text{rounds}(\Pi)$ rounds. Consequently, we derive that $\text{view}_{2PBA}(A_1) = \text{view}_S(A_1)$, $\text{view}_{2PBA}(T) = \text{view}_S(T)$, and $\text{view}_{2PBA}(H) = \text{view}_S(\tilde{H})$.

Given the equivalence of views, the parties A_1 , T and H in both systems 2PBA and S must yield identical final outputs. By the validity property of Byzantine Agreement, since A_1 , T , and H in Π_0 are initialized with input 0 and A_1 , T , $ITM_0^C \subset H$ are honest, then $\text{Output}_{2PBA}(A_1) = \text{Output}_{2PBA}(T) = \text{Output}_{2PBA}(ITM_0^C) = 0$. Consequently, system S inherits this outcome, we have $\text{Output}_S(A_1) = \text{Output}_S(T) = \text{Output}_S(ITM_0^C) = 0$.

Analogously, through the system 2PBA, we can rigorously demonstrate $\text{view}_{2PBA}(B_1) = \text{view}_S(B_1)$, $\text{view}_{2PBA}(\tilde{T}) = \text{view}_S(\tilde{T})$ and $\text{view}_{2PBA}(\tilde{H}) = \text{view}_S(\tilde{H})$, thereby establishing $\text{Output}_S(B_1) = \text{Output}_S(\tilde{T}) = \text{Output}_S(ITM_1^C) = 1$. ◀

Proof of Lemma 6. Similarly, we construct system 2PBA' as instantiated below, shown in Figure 4:

- **Network Topology:** The system consists of two parallel executions of protocol Π , denoted Π'_0 (consisted of A_1 , \tilde{H} and \tilde{T}) and Π'_1 (consisted of B_1 , H and T).
- **Initial Input Value:**
 - ITM $\{A_1\}$ receives input 0 and ITMs in $\tilde{H} \cup \tilde{T}$ receive input 1;
 - ITM $\{B_1\}$ receives input 0 and ITMs in $H \cup T$ receive input 1.



■ **Figure 4** Party's view is identical between System 2PBA' and System S.

- **Adversary:** The adversary $\mathcal{A}(t, c)$ chooses an arbitrarily subset of parties $\{P_2, \dots, P_{\lfloor \frac{n+1}{2} \rfloor}\}$, denoted by C' , which satisfies $|C'| = \min\{c, \lfloor \frac{n+1}{2} \rfloor - 1\}$. Then the adversary reorders all the channels between C' and P_1 , and corrupts the parties in $\{P_2, \dots, P_{\lfloor \frac{n+1}{2} \rfloor}\} \setminus C'$. As we have $2c + 2t + 1 \geq n$, the adversary can indeed corrupt all these parties. In the following proof, we denote $ITM_0^{C'} = \{ITM_0^j : j \in C'\}$ as the set of ITMs run by C' in the protocol Π_0 , and as the same, we denote $ITM_1^{C'} = \{ITM_1^j : j \in C'\}$.
- **Protocol Execution:**
 - All honest parties in S strictly adhere to the instruction of protocol Π ;
 - The corrupted parties reorder their channels between P_1 , and besides this action, they adhere to the instruction of protocol Π .

The subsequent proof by mathematical induction is analogous to the reasoning in system 2PBA of the preceding proof; therefore it is omitted here. It is noteworthy that after we get $\text{view}_{2PBA'}(A_1) = \text{view}_S(A_1)$, $\text{view}_{2PBA'}(\tilde{H}) = \text{view}_S(\tilde{H})$, and $\text{view}_{2PBA'}(\tilde{T}) = \text{view}_S(\tilde{T})$, we use the Agreement property instead of the Validity property of Byzantine agreement to deduce that $\text{Output}_S(A_1) = \text{Output}_S(\tilde{H}) = \text{Output}_S(ITM_1^{C'})$. Similarly, we have $\text{Output}_S(B_1) = \text{Output}_S(H) = \text{Output}_S(ITM_0^{C'})$. ◀

5 Black-box Compiler for Parallel Composable BA

This section presents a novel black-box compiler that transforms any BA protocol into one that remains secure under parallel composition. The proposed approach addresses the challenges introduced by message reordering attacks through the integration of a new Reliable Message Transmission (RMT) protocol.

5.1 Reliable Message Transmission for Parallel Composition

To achieve security under parallel composition, our compiler replaces the standard point-to-point communication primitives in a given BA protocol with our newly designed Reliable Message Transmission protocol. This RMT primitive guarantees correct and unforgeable message delivery within the same protocol instance, even in the presence of adversarial message reordering. The formal specification of the RMT is presented in Protocol 1.¹

■ **Algorithm 1** Reliable Message Transmission.

Given that party P_i wishes to send message msg to P_j , the reliable message transmission protocol proceeds as follows:

- Round 0: Party P_i initiates this message transmission by sending the message $(deliver, msg, P_i, P_j)$ to all parties except P_j .
 - Round 1: Party P_i sends the message $(deliver, msg, P_i, P_j)$ to P_j ; and each other party $P_k \neq P_j$ that received message $(deliver, msg, P_i, P_j)$ from P_i in round 0 forwards the same message $(deliver, msg, P_i, P_j)$ to party P_j .
 - Decision: Upon party P_j receiving strictly more than $\frac{n-1}{2}$ copies of the message $(deliver, msg, P_i, P_j)$ by the end of round 1, it accepts message msg from P_i .
-

When $n > 2t + 2c + 1$, this RMT protocol ensures message delivery through a redundancy-based approach. The principle is to use a majority vote to secure message transmission against potential attacks, a technique also found in prior works but in different formats; see, e.g., [49, 29, 13]. When executed by all participating parties, the RMT primitive is characterized by two essential properties for its security within the black-box compiler.

- *Correctness*: If both parties P_i and P_j are honest, and P_i used the RMT protocol to send a message msg to P_j in round r , then P_j will accept the message msg from P_i by the end of round $r + 1$.
- *Unforgeability*: If both parties P_i and P_j are honest, and P_i did not use the RMT protocol to send message msg to P_j in round r , then P_j will not accept msg from P_i by the end of round $r + 1$.

The following lemma establishes the security guarantees of the RMT protocol:

► **Lemma 7.** *Protocol 1 satisfies both correctness and unforgeability properties against any adversary $\mathcal{A}(t, c)$, provided that $n > 2c + 2t + 1$.*

Proof. *Correctness*: Since P_i is an honest party, it will send the message $(deliver, msg, P_i, P_j)$ to all other parties except P_j in round 0. In round 1, each party $P_k \neq P_j$, including P_i , forwards/sends the message to P_j , unless:

- P_k is corrupted by the adversary, or
- The communication channel C_{ik} or C_{kj} is attacked by the adversary.

Since each corruption or channel attack can prevent at most one forwarding, at most $t + c$ of these forwarded messages can be suppressed. Thus, P_j receives at least $n - 1 - t - c$ forwarded messages. Since $n > 2c + 2t + 1$, we have $\frac{n-1}{2} > t + c$, which is equivalent to $n - 1 - t - c > \frac{n-1}{2}$. Therefore, P_j receives strictly more than $\frac{n-1}{2}$ copies of the message $(deliver, msg, P_i, P_j)$, and accepts msg from P_i as valid by the end of round 1.

¹If the values of t and c are publicly known, then the threshold $\frac{n-1}{2}$ in Protocol 1 can be replaced by $t + c$ and the protocol continues to work.

Unforgeability: If P_i did not initiate the RMT protocol for message msg to P_j , then the only way for P_j to receive $(deliver, msg, P_i, P_j)$ is via adversarial intervention. The adversary can inject at most $t + c$ such messages through corrupted parties or attacked channels. Since $t + c < \frac{n-1}{2}$, P_j receives fewer than the majority threshold of messages, and thus does not accept msg from P_i . ◀

5.2 Black-box Compiler

Using the RMT protocol, our black-box compiler applies a simple yet effective transformation: given an existing BA protocol Π_* , it constructs a new protocol Π by replacing all point-to-point message delivery operations with the RMT protocol. The internal logic, state transitions, and computation procedures of Π_* are completely preserved, ensuring that the transformation is non-intrusive and protocol-agnostic.

Formally, a party P_i 's ITM^i in protocol Π uses its corresponding ITM_*^i in Π_* as a sub-routine. The outer one, ITM^i , handles message sending and receiving according to the RMT protocol, and passes accepted messages to the inner one, ITM_*^i . The inner one then computes its state transition and out-going messages according to Π_* , and passes the out-going messages to ITM^i to send out.

Notice that in Π , a message that originally takes one round to deliver in Π_* now takes two rounds. As such, honest parties will only initiate a message sending in odd rounds (wlog assuming the protocol starts with round 1) and will only accept a message in even rounds. More specifically, if in the execution of Π_* an honest party P_i sends a message to P_j in round r , then in the execution of Π , P_i initiates the corresponding message sending in round $2r - 1$ and P_j accepts the message by the end of round $2r$.

► **Theorem 8.** *Let Π_* be a BA protocol for n parties that tolerates up to t Byzantine corruptions. Let Π be the protocol obtained by replacing all point-to-point communication in Π_* with the RMT protocol. Then, Π is secure under parallel composition of any number of executions against any adversary $\mathcal{A}(t, c)$, provided that $n > 2t + 2c + 1$.*

Proof. Arbitrarily fixing an instance Π_j , we prove its security under parallel composition by induction. Consider a mental game with a single execution of the protocol Π_* as follows.

For each honest party P_i and the corresponding ITM_j^i in Π_j , let its inner Turing machine be ITM_{*j}^i . We construct an ITM_*^i for Π_* , such that ITM_*^i has the same initial input tape, random tape (and setup tape) as ITM_j^i , and thus also as ITM_{*j}^i .

Note that, for any adversary $\mathcal{A}(t, c)$ in Π , the only effects it can have on some party P_i in instance Π_j are the following:

- It can fully control ITM_j^i if the party P_i is corrupted.
- It can cause an honest party P_i 's ITM_j^i to accept a message msg from a corrupted party P_j at some round $2r$. This is because, in Π , no honest ITM accepts a message in odd rounds, and due to the unforgeability property of RMT, the adversary cannot forge a message from other honest parties to make an honest P_i accept it.

Therefore, for any adversary \mathcal{A} in Π_j , we simulate a corresponding adversary \mathcal{A}_* in Π_* as follows:

- Whenever \mathcal{A} corrupts a party P_i in Π_j , \mathcal{A}_* corrupts the corresponding ITM_*^i in Π_* .
- Whenever \mathcal{A} causes an honest ITM_j^i to accept a message msg from a corrupted party P_j at round $2r$ in Π_j , \mathcal{A}_* sends message msg from ITM_j^j to ITM_*^i at round r in Π_* .

Since for every honest party P_i , both ITM_{*j}^i and ITM_*^i have the same input, random, and setup tapes at the beginning of round 1, they have the same state transition and out-going messages. Because ITM_j^i handles message sending and receiving according to the RMT

protocol, and by the construction of the adversary \mathcal{A}_* in Π_* , at the end of round 2 in Π_j , ITM_j^i has accepted exactly the same messages as received by ITM_*^i at the end of round 1 in Π_* . Thus ITM_{*j}^i after round 2 of Π_j has the same view as ITM_*^i after round 1 of Π_* . That is, $\text{view}_{\Pi_j}^2(ITM_{*j}^i) = \text{view}_{\Pi_*}^1(ITM_*^i)$.

Let $\text{round}(\Pi_*)$ be the maximum finite number of rounds in protocol Π_* . For each $r = 1, \dots, \text{round}(\Pi_*)$, we compare the view of ITM_{*j}^i after $2r$ rounds of Π_j with the view of ITM_*^i after r rounds of Π_* . By the inductive hypothesis, we have

$$\text{view}_{\Pi_j}^{2r-2}(ITM_{*j}^i) = \text{view}_{\Pi_*}^{r-1}(ITM_*^i).$$

Thus they again have the same state transition and out-going messages in the corresponding round $2r - 1$ and round r .

Due to the correctness and the unforgeability properties of the RMT protocol, and by the construction of the adversary \mathcal{A}_* , again by the end of round $2r$ of Π_j , ITM_j^i accepts exactly the same messages as received by ITM_*^i at the end of round r in Π_* . Hence, $\text{view}_{\Pi_j}^{2r}(ITM_{*j}^i) = \text{view}_{\Pi_*}^r(ITM_*^i)$ for all r .

Accordingly, we conclude that the final output value of ITM_j^i in Π_j (which is that of ITM_{*j}^i) and the final output value of ITM_*^i in Π_* are identical. As this holds for all honest parties in Π_j , we have that Π_j inherits the same agreement and validity properties as Π_* . Thus Π is secure under parallel composition. \blacktriangleleft

By applying our compiler to any existing BA protocol, we can construct a protocol that remains secure under parallel composition. In particular, by transforming the protocol of Garay and Moses [31], which is secure in the unauthenticated, stateless setting whenever $n > 3t$, we obtain the following corollary:

► Corollary 9. *There exists a protocol for stateless, unauthenticated BA that is secure under parallel composition of any number of executions against any adversary $\mathcal{A}(t, c)$, provided that the total number of parties n satisfies $n > 3t$ and $n > 2c + 2t + 1$.*

While the black-box compiler significantly enhances BA protocol's security, it introduces some performance overheads. The round complexity of the compiled protocol Π is doubled compared with the original protocol Π_* . The communication overhead per message in the original Π_* is asymptotically $O(n)$. In particular, for a message from P_i to P_j , P_i sends to $n - 2$ parties in Round 0 of RMT, and up to $n - 1$ parties may send the message to P_j in round 1. These overheads are the trade-off for achieving security against adversarial message reordering in parallel compositions. Designing more efficient compilers that maintain similar security guarantees while reducing communication or round complexity remains an important direction for future research.

6 Concurrent BA Protocol under Asynchronous Network

The preceding sections focused on parallel composition under a synchronous network model, where message delivery is guaranteed within a fixed time bound (i.e., a round). In this section, we shift our attention to a more challenging setting, the asynchronous network model. Here, messages can be delayed arbitrarily, and there is no global clock to coordinate the actions of different parties. Moreover, we consider concurrent composition, where protocol instances may not start simultaneously, making it significantly harder for a party to determine whether a received message indeed belongs to a particular instance or has been reordered.

To address these challenges, we first construct a reliable broadcast protocol tailored for the asynchronous setting. Building upon this foundation, we then leverage the seminal work of [6] and construct an unauthenticated BA protocol that is secure under concurrent composition in an asynchronous network. Notably, security in the asynchronous model implies security in the synchronous setting. Therefore, our BA protocol is also secure under concurrent composition in synchronous networks.

6.1 Reliable Broadcast Under Concurrent Composition

Reliable Broadcast (RB) [10] is a fundamental communication primitive that ensures a message sent by an honest sender is reliably delivered to all parties. It guarantees the following properties:

- *Agreement*: If any honest party accepts a message m , then all honest parties eventually accept m .
- *Validity*: If the sender is honest and broadcasts a message m , then all honest parties eventually accept m .

Based on the work of [6] and utilizing the idea behind our RMT protocol in Section 5, we present Protocol 2 that achieves reliable broadcast under concurrent executions against any adversary $\mathcal{A}(t, c)$, under the conditions $n > 3t$ and $n > 2c + 2t + 1$. To our best knowledge, this is the first RB protocol that is concurrently secure in the stateless model, and may be of independent interest.

The core idea is to avoid accepting messages based solely on direct receipt. Instead, each party maintains a local view of all other parties' state. This view is only updated after a majority (i.e., more than $\frac{n-1}{2}$) of confirming messages have been received. Specifically, let the original sender in the RB protocol be $P_g = P_1$. The local state of a party P_j with $j = 1, \dots, n$ at some party P_i , denoted by S_j^i , can be in one of the following three forms:

- $\{initial, \perp\}$: no value has been received by P_j yet;
- $\{prepare, v\}$: value v tentatively accepted by P_j ;
- $\{commit, v\}$: value v confirmed and will not change at P_j .

Respectively, the messages sent by a party P_i during the protocol reflect its own state and take one of the following forms:

- $(receive, v, P_g)$: P_i has received value v sent by P_g ;
- $(echo, v, P_i)$: P_i has tentatively accepted v ;
- $(ready, v, P_i)$: value v has been confirmed by P_i .

By adapting the BA protocol Π^* from Figure 4 of [6] and replacing all broadcast procedures with our RB protocol, we have the following:

► **Theorem 10.** *There exists a protocol for stateless, unauthenticated BA that is secure under concurrent composition of any number of executions in an asynchronous network against any adversary $\mathcal{A}(t, c)$, where the number of parties n satisfies $n > 3t$ and $n > 2c + 2t + 1$.*

To prove Theorem 10, it suffices to prove the security of our RB protocol under concurrent composition in an asynchronous network; see Theorem 17 in Section 6.2. Theorem 10 then holds by the security of protocol Π^* and an inductive analysis similar to the proof of Theorem 8. We include the protocol Π^* in the appendix of this paper for reference purpose.

■ **Algorithm 2** Reliable Broadcast.

Say party P_g wishes to broadcast value v . Each party P_i initializes its local view of each party P_j 's state as $S_j^i = \{initial, \perp\}$. The protocol proceeds as follows:

- Step 0 (performed by P_g): send $(initial, v)$ to all parties including itself.
- For each party P_i :
- Step 1: Upon receiving some message $(initial, v)$ from P_g , send $(receive, v, P_g)$ to all parties.
 - Step 2: Wait until the receipt of strictly more than $\frac{n-1}{2}$ $(receive, v, P_g)$ messages for the same value v from different parties. If local state $S_i^i = \{initial, \perp\}$, set $S_i^i = \{prepare, v\}$ and send $(echo, v, P_i)$ to all parties.
 - Step 3: Upon receiving some message $(echo, v, P_j)$ from party P_j , forward it to all parties.
 - Step 4: Wait until the receipt of strictly more than $\frac{n-1}{2}$ $(echo, v, P_j)$ messages for the same v and P_j from different parties. If $S_j^i = \{initial, \perp\}$, set $S_j^i = \{prepare, v\}$.
 - Step 5: Wait until strictly more than $\frac{n+t}{2}$ parties P_j have local state $S_j^i = \{prepare, v\}$ for the same v . If $S_i^i = \{initial, \perp\}$ or $S_i^i = \{prepare, * \}$, where $*$ can be any value, set $S_i^i = \{commit, v\}$ and send $(ready, v, P_i)$ to all parties.
 - Step 6: Upon receiving some message $(ready, v, P_j)$ from party P_j , forward it to all parties.
 - Step 7: Wait until the receipt of strictly more than $\frac{n-1}{2}$ $(ready, v, P_j)$ messages for the same v and P_j from different parties. If $S_j^i = \{initial, \perp\}$ or $S_j^i = \{prepare, * \}$, where $*$ can be any value, set $S_j^i = \{commit, v\}$.
 - Step 8: Wait until $t+1$ parties P_j have state $S_j^i = \{commit, v\}$ for the same v . If $S_i^i = \{initial, \perp\}$ or $S_i^i = \{prepare, * \}$, where $*$ can be any value, set $S_i^i = \{commit, v\}$ and send $(ready, v, P_i)$ to all parties.
 - Step 9: Wait until $2t+1$ parties P_j have state $S_j^i = \{commit, v\}$ for the same v . Accept v .

6.2 Composition Security of Reliable Broadcast

We now prove that Protocol 2 satisfies the properties of reliable broadcast under concurrent composition. In particular, we consider m instances of Protocol 2 concurrently executed against adversary $\mathcal{A}(t, c)$, under conditions $n > 3t$ and $n > 2c + 2t + 1$.

► **Lemma 11.** *In any instance, for any two honest parties P_i and P_j , P_i eventually sets its local state $S_j^i = \{commit, v\}$ if and only if P_j broadcasts the message $(ready, v, P_j)$ during the execution of this instance.*

Proof. If an honest party P_j broadcasts $(ready, v, P_j)$, then in the execution of any other honest party P_i , this message will be forwarded by every party P_k unless:

- P_k is corrupted by the adversary, or
- the communication channel C_{jk} or C_{ki} is under attack.

Since each adversary action can block at most one message, the number of suppressed forwards is at most $t + c$. Therefore, P_i eventually receives more than $\frac{n-1}{2}$ such messages and sets its local state for P_j to $S_j^i = \{commit, v\}$.

Conversely, if P_j did not send the message in that instance, then the adversary can inject at most $t + c$ copies $(ready, v, P_j)$. Since $t + c < \frac{n-1}{2}$, an honest party will not receive enough messages to satisfy the threshold, and thus will not set P_j 's local state to $S_j^i = \{commit, v\}$. ◀

► **Lemma 12.** *In any instance, for any two honest parties P_i and P_j , P_i eventually sets its local state $S_j^i = \{\text{prepare}, v\}$ only if P_j broadcasts the message (echo, v, P_j) during the execution of this instance.*

Proof. If P_j did not send the message (echo, v, P_j) in that instance, then the adversary can inject at most $t + c$ copies (echo, v, P_j) . Since $t + c < \frac{n-1}{2}$, an honest party will not receive enough messages to satisfy the threshold, and thus will not set P_j 's local state to $S_j^i = \{\text{prepare}, v\}$. ◀

► **Lemma 13.** *In any instance, for any two honest parties P_i and P_j who respectively send messages (ready, v, P_i) and (ready, u, P_j) during the execution of this instance, we have $v = u$.*

Proof. Assume for contradiction. Let P_x and P_y be the first honest parties to send (ready, v, P_x) and (ready, u, P_y) , respectively. Party P_x must have observed strictly more than $\frac{n+t}{2}$ party P_j with local state $S_j^x = \{\text{prepare}, v\}$. Party P_y must have observed strictly more than $\frac{n+t}{2}$ party P_j with local state $S_j^y = \{\text{prepare}, u\}$. This implies an overlap where an honest party P_k has $\{\text{prepare}, v\}$ in the view of P_x and $\{\text{prepare}, u\}$ in the view of P_y .

Therefore, by Lemma 12, it implies that P_k sent both (echo, v) and (echo, u) messages. However, an honest party sends only one type of *echo* message during a reliable broadcast protocol. Therefore, $v = u$. ◀

► **Lemma 14.** *In any instance, for any two honest parties P_i and P_j who accept values v and u respectively, we have $v = u$.*

Proof. If P_i accepts v , then it must have observed at least $2t + 1$ parties P_k with the state $S_k^i = \{\text{commit}, v\}$ in its local view, which includes at least $t + 1$ honest parties. Similarly, P_j must have observed at least $2t + 1$ parties P_k with state $S_k^j = \{\text{commit}, u\}$ in its local view, which includes at least $t + 1$ honest parties. By Lemma 11, there exists at least one honest party P_x that have send (ready, v, P_x) , and one honest party P_y that have send (ready, u, P_y) . Applying Lemma 13, it follows that $v = u$. ◀

► **Lemma 15.** *In any instance, if an honest party P_i accepts a value v , then every other honest party eventually accepts v in that instance.*

Proof. Suppose P_i accepts value v . Then its local view must include at least $2t + 1$ parties P_x with the state $S_x^i = \{\text{commit}, v\}$, including at least $t + 1$ honest parties. By Lemma 11, every honest party P_k in these $t + 1$ parties must have send (ready, v, P_k) in that instance. Also by Lemma 11, eventually, all honest parties P_j will receive these messages and update their local views, setting P_k 's state to $S_k^j = \{\text{commit}, v\}$. According to step 8 of the protocol, upon observing $t + 1$ parties in state $\{\text{commit}, v\}$, every honest party P_j updates its own state to $S_j^j = \{\text{commit}, v\}$, and send (ready, v, P_j) to all parties. Again, by Lemma 11, all honest parties P_y will receive these messages and update their local views, setting P_j 's state $S_j^y = \{\text{commit}, v\}$ eventually. Since $n > 3t$, the number of honest parties exceeds $2t + 1$. Thus, every honest party will eventually have at least $2t + 1$ $\{\text{commit}, v\}$ states in its local view and will accept v . ◀

► **Lemma 16.** *In any instance, if an honest party P_g broadcasts a value v , then all honest parties eventually accept v in that instance.*

Proof. Since P_g is honest, it sends $(initial, v, P_g)$ to all parties. At least $(n - t - c)$ honest parties receive this message and forward $(receive, v, P_g)$ to all other parties. Each honest party P_i eventually receives more than $\frac{n-1}{2}$ such $(receive, v, P_g)$ messages. It then sets its local state $S_i^i = \{prepare, v\}$ and sends $(echo, v, P_i)$ to all parties. As a result, each honest party P_i receives more than $\frac{n-1}{2}$ *echo* messages for value v and each honest party P_j , and thus sets the state $S_j^i = \{prepare, v\}$. This ensures that every honest party P_i observes enough *prepare* state in its local view to satisfy the condition for committing. Accordingly, each honest party updates its own state $S_i^i = \{commit, v\}$ and sends $(ready, v, P_i)$ to all parties.

In Step 7 of the protocol, each honest party P_i receives more than $\frac{n-1}{2}$ *ready* messages for value v and each honest party P_j , causing each party to update the state of P_j $S_j^i = \{commit, v\}$. Therefore, since $n > 3t$, every honest party P_i eventually has at least $2t + 1$ $\{commit, v\}$ states in local view and accepts v . ◀

▶ **Theorem 17.** *Protocol 2 realizes reliable broadcast against any adversary $\mathcal{A}(t, c)$ under concurrent composition in an asynchronous network, provided that $n > 3t$ and $n > 2t + 2c + 1$.*

Proof. Validity: In any instance, if the sender P_g is honest and broadcast value v , then by Lemma 16, all honest parties eventually accept v in that instance.

Agreement: In any instance, if the sender P_g is corrupted, and some honest party P_i accepts a value v in that instance, then by Lemma 15, all other honest parties eventually accept the same value v in that instance. ◀

7 Conclusion and Future Directions

The primary contributions of this work include the introduction of a novel adversarial model that simultaneously enables the corruption of parties and communication channels. The reorder attack is both theoretically interesting and practically viable, enhancing the security analysis for Byzantine Agreement protocols under both parallel and concurrent execution settings.

Building upon this model, our work establishes surprising positive and negative results. On the one hand, authenticated Byzantine Agreement becomes insecure when either $n \leq 3t$ or $n \leq 2c + 2t + 1$. While on the other hand, for unauthenticated Byzantine Agreement, secure protocols exist under both parallel and concurrent compositions when $n > 3t$ and $n > 2c + 2t + 1$. Notably, these findings provide tight conditions for the security of Byzantine Agreement under compositional executions – a fundamental advance in understanding protocol resilience against corruption and reorder attacks.

The framework and results established in this paper motivate several promising directions for future research:

1. In terms of positive results, we provide a black-box reduction for parallel-executed BA protocols. Specifically, any unauthenticated Byzantine Agreement that is secure under single-instance execution when $t < n/3$ can be efficiently transformed into a parallel-secure protocol when in addition one has $2t + 2c + 1 < n$. For concurrently-executed Byzantine Agreement, we demonstrate our positive result via a concrete protocol using reliable broadcast. We believe that our reliable broadcast protocol can also be used to help other BA protocols achieve concurrent security, but a key open question remains: does there exist a black-box reduction approach that generically transforms a single-instance secure BA protocol into one that is concurrently secure? Also, the composition of BA protocols against dynamic adversaries would be very interesting to explore, since so far all studies on compositions consider static adversaries.

2. The communication complexity of BA protocols has been a central focus in the literature. Under our proposed adversarial model and compositional execution framework, this work focused on polynomial-time computation and communication protocols and didn't investigate whether the complexity bounds of compositionally-secure BA protocols diverge from those of classical single-execution authenticated Byzantine Agreement. Specifically, we are curious about how reorder attacks may impact existing communication lower- and upper-bounds. Intuitively, such attacks necessitate additional communication to preserve security, but the exact complexity trade-offs require a rigorous study.
3. In this work we focus on the standard notion of validity commonly considered in the literature. Since other validity notions such as *weak validity* have also been studied [42], we are interested in whether our conclusions still hold with respect to those notions. Intuitively, weaker validity conditions would relax the safety constraints that BA protocols must satisfy, hence may allow the impossibility results be circumvented. Thus extending our current model to weaker validity notions constitutes a promising direction for future research.

References

- 1 Michael Abd-El-Malek, Gregory R. Ganger, Garth R. Goodson, Michael K. Reiter, and Jay J. Wylie. Fault-scalable Byzantine fault-tolerant services. *SIGOPS Oper. Syst. Rev.*, 39(5):59–74, October 2005. doi:10.1145/1095809.1095817.
- 2 Shreya Agrawal and Khuzaima Daudjee. A performance comparison of algorithms for Byzantine agreement in distributed systems. In *2016 12th European Dependable Computing Conference (EDCC)*, pages 249–260. IEEE, 2016. doi:10.1109/EDCC.2016.17.
- 3 Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distrib. Comput.*, 16(4):249–262, 2003. doi:10.1007/s00446-002-0083-3.
- 4 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10. ACM, 1988. doi:10.1145/62212.62213.
- 5 Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $O(\sqrt{n})$ -bit barrier: Byzantine agreement with polylog bits per party. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC '21, pages 319–330. ACM, 2021. doi:10.1145/3465084.3467897.
- 6 Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987. doi:10.1016/0890-5401(87)90054-X.
- 7 Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, and Sebastian Schinzel. ALPACA: Application layer protocol confusion - analyzing and mitigating cracks in TLS authentication. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann>.
- 8 R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, page 136. IEEE, 2001.
- 9 Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186. USENIX Association, 1999. URL: <https://dl.acm.org/citation.cfm?id=296824>.
- 10 Jo-Mei Chang and Nicholas F. Maxemchuk. Reliable broadcast protocols. *ACM Trans. Comput. Syst.*, 2(3):251–273, 1984. doi:10.1145/989.357400.
- 11 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 11–19. ACM, 1988. doi:10.1145/62212.62214.

- 12 Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019. doi:10.1016/J.TCS.2019.02.001.
- 13 Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, and Jovan Komatovic. As easy as ABC: optimal (a)ccountable (b)yzantine (c)onsensus is easy! *J. Parallel Distributed Comput.*, 181:104743, 2023. doi:10.1016/J.JPDC.2023.104743.
- 14 Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, Anton Paramonov, and Manuel Vidigueira. All Byzantine agreement problems are expensive. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, PODC '24, pages 157–169. ACM, 2024. doi:10.1145/3662158.3662780.
- 15 Jeremy Clark and Paul C. van Oorschot. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *2013 IEEE Symposium on Security and Privacy*, pages 511–525, 2013. doi:10.1109/SP.2013.41.
- 16 Ran Cohen, Pouyan Forghani, Juan Garay, Rutvik Patel, and Vassilis Zikas. Concurrent asynchronous Byzantine agreement in expected-constant rounds, revisited. In *Theory of Cryptography Conference*, pages 422–451. Springer, 2023. doi:10.1007/978-3-031-48624-1_16.
- 17 Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. Round-efficient Byzantine agreement and multi-party computation with asynchronous fallback. In *Theory of Cryptography Conference*, pages 623–653. Springer, 2021. doi:10.1007/978-3-030-90459-3_21.
- 18 Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for Byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985. doi:10.1145/2455.214112.
- 19 Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983. doi:10.1137/0212045.
- 20 Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988. doi:10.1145/42282.42283.
- 21 Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 370–379. ACM, 1986. doi:10.1145/12130.12169.
- 22 Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. Impact of man-in-the-middle attacks on Ethereum. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 11–20, 2018. doi:10.1109/SRDS.2018.00012.
- 23 Paul Feldman and Silvio Micali. Optimal algorithms for Byzantine agreement. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 148–161. ACM, 1988. doi:10.1145/62212.62225.
- 24 Pease Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997. doi:10.1137/S0097539790187084.
- 25 Rex Fernando, Yuval Gelles, and Ilan Komargodski. Scalable distributed agreement from LWE: Byzantine agreement, broadcast, and leader election. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, pages 1–23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ITCS.2024.46.
- 26 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985. doi:10.1145/3149.214121.
- 27 Matthias Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, 2003.
- 28 Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional Byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology – EUROCRYPT '02*, pages 482–501. Springer, 2002. doi:10.1007/3-540-46035-7_32.
- 29 Matthias Fitzi and Ueli M. Maurer. From partial consistency to global broadcast. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 494–503. ACM, 2000. doi:10.1145/335305.335363.

- 30 Zvi Galil, Stuart Haber, and Moti Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Advances in Cryptology – CRYPTO ’87*, pages 135–155. Springer, 1987. doi:10.1007/3-540-48184-2_10.
- 31 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. doi:10.1137/S0097539794265232.
- 32 Yuval Gelles and Ilan Komargodski. Optimal load-balanced scalable distributed agreement. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC ’24, pages 411–422. ACM, 2024. doi:10.1145/3618260.3649736.
- 33 Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology – CRYPTO ’02*, pages 178–193. Springer, 2002. doi:10.1007/3-540-45708-9_12.
- 34 Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: Validating SSL certificates in non-browser software. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS ’12, pages 38–49. ACM, 2012. doi:10.1145/2382196.2382204.
- 35 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, STOC ’87, pages 218–229. ACM, 1987. doi:10.1145/28395.28420.
- 36 Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988. doi:10.1137/0217017.
- 37 Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. A new look at composition of authenticated Byzantine generals. *arxiv*, 2012. arXiv:1203.1463.
- 38 Lioba Heimbach and Roger Wattenhofer. SoK: Preventing transaction reordering manipulations in decentralized finance. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, AFT ’22, pages 47–60. ACM, 2023. doi:10.1145/3558535.3559784.
- 39 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for Byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, 2009. doi:10.1016/J.JCSS.2008.08.001.
- 40 Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA ’06, pages 990–999. SIAM, 2006. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109667>.
- 41 Po-Chun Kuo, Hao Chung, Tzu-Wei Chao, and Chen-Mou Cheng. Fair Byzantine agreements for blockchains. *IEEE Access*, 8:70746–70761, 2020. doi:10.1109/ACCESS.2020.2986824.
- 42 Leslie Lamport. The weak byzantine generals problem. *Journal of the ACM (JACM)*, 30(3):668–676, 1983. doi:10.1145/2402.322398.
- 43 Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. doi:10.1145/357172.357176.
- 44 Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated Byzantine agreement. *J. ACM*, 53(6):881–917, 2006. doi:10.1145/1217856.1217857.
- 45 Thomas Locher. Fast Byzantine agreement for permissioned distributed ledgers. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA ’20, pages 371–382. ACM, 2020. doi:10.1145/3350755.3400219.
- 46 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980. doi:10.1145/322186.322188.
- 47 Birgit Pfitzmann and Michael Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, STACS ’92, pages 339–350. Springer, 1992. doi:10.1007/3-540-55210-3_195.
- 48 Zhiguo Qu, Zhexi Zhang, Bo Liu, Prayag Tiwari, Xin Ning, and Khan Muhammad. Quantum detectable Byzantine agreement for distributed data trust management in blockchain. *Information Sciences*, 637:118909, 2023. doi:10.1016/J.INS.2023.03.134.

- 49 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 73–85. ACM, 1989. doi:10.1145/73007.73014.
- 50 R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.
- 51 David Sounthiraraj, Justin Sahs, Garrett Greenwood, Zhiqiang Lin, and Latifur Khan. Smv-hunter: Large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps. In *Network and Distributed System Security Symposium (NDSS)*, pages 1–14. The Internet Society, 2014. doi:10.14722/ndss.2014.23205.
- 52 Ran Tamir, Ariel Livshits, and Yonatan Shadmi. Simple majority consensus in networks with unreliable communication. *Entropy*, 24(3), 2022. doi:10.3390/e24030333.
- 53 Jianhuan Wang, Jichen Li, Zecheng Li, Xiaotie Deng, and Bin Xiao. n-mvlt attack: Optimal transaction reordering attack on DeFi. In *Computer Security – ESORICS 2023*, pages 367–386. Springer, 2023. doi:10.1007/978-3-031-51479-1_19.
- 54 Andrew Chi-Chih Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, SFCS '82*, pages 160–164. IEEE, 1982. doi:10.1109/SFCS.1982.38.
- 55 Junghun Yoo, Youlim Jung, Donghwan Shin, Minhyo Bae, and Eunkyong Jee. Formal modeling and verification of a federated Byzantine agreement algorithm for blockchain platforms. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 11–21. IEEE, 2019.
- 56 Zhenwei Zhao, Xiaoming Li, Bing Luan, Weining Jiang, Weidong Gao, and Subramani Neelakandan. Secure internet of things (IoT) using a novel Brooks Iyengar quantum Byzantine agreement-centered blockchain networking (BIQBA-BCN) model in smart healthcare. *Information Sciences*, 629:440–455, 2023. doi:10.1016/J.INS.2023.01.020.

A Consensus Protocol in Figure 4 of [6]

■ **Algorithm 3** The Consensus Protocol in Figure 4 of [6].

Phase(i): (by process p)

Step 1: $Broadcast(p, 3i + 1, value_p)$. Wait until validate $n - t$ ($3i + 1$)-messages.

■ $value_p :=$ majority value of the $n - t$ validated messages.

Step 2: $Broadcast(p, 3i + 2, value_p)$. Wait until validate $n - t$ ($3i + 2$)-messages.

■ (i) If more than $\frac{n}{2}$ of the messages have the same value v , then $value_p = (d, v)$.

■ (ii) Otherwise, $value_p := value_p$.

Step 3: $Broadcast(p, 3i + 3, value_p)$. Wait until validate $n - t$ ($3i + 3$)-messages.

■ (i) If validated more than $2t$ messages with value (d, v) then $decision_p := value_p := v$.

■ (ii) If validated more than t messages with value (d, v) then $value_p := v$.

■ (iii) Otherwise, $value_p := coin_toss$ (0 or 1 with probability $\frac{1}{2}$).

Go to round 1 of phase $i + 1$
