Byzantine Agreement with Less Communication: Recent Advances

Shir Cohen[1], Idit Keidar[1], and Oded Naor[1]

[1]Technion – Israel Institute of Technology

1 Introduction

The development of reliable distributed systems often relies on Byzantine Agreement (BA) [26]. In this problem, a set of correct processes aim to reach a common decision, despite the presence of malicious ones. BA has been around for four decades, yet practical use-cases for it in large-scale systems have emerged only in the last decade. One major application for BA is cryptocurrencies. For example, Bitcoin [30], the first cryptocurrency, requires a large set of users to agree on the state of the blockchain. Since Bitcoin is a real currency with real value, the need to protect it against Byzantine users is crucial. Following Bitcoin, many other blockchains and FinTech platforms have emerged, e.g., [6, 21, 30, 36]. Consequently, an efficient implementation, in terms of communication, has become one of the main foci of BA solutions.

It has been shown by Dolev and Reischuk [17] that in deterministic algorithms with n processes, $\Omega(n^2)$ word complexity is needed in the worst-case, assuming the number of faulty processes in the system is $f = O(n)$. The word complexity of a deterministic BA protocol is defined as the number of words all correct processes send until a decision is reached, where a word is a constant number of bits (e.g., the size of a PKI signature). Nevertheless, almost all deterministic works incur a word complexity of at least $O(n^3)$ in a synchronous model with n processes, f of which can fail, and the optimal resilience of $n \geq 2f + 1$ [3, 18]. In fact, this gap remained open for 35 years until recently Momose and Ren [28] solved synchronous BA with optimal resilience with $O(n^2)$ words, although less resilient solutions with $O(n^2)$ complexity have been known prior to this result.

Yet synchronous solutions are not robust in large-scale systems, where messages can be delayed for extensive periods. A more practical approach is to consider the *eventual synchrony* (ES) model, where communication is initially asynchronous but eventually becomes synchronous. Eventually, synchronous algorithms always ensure safety, but their liveness is conditioned on communication becoming timely. In this model, performance is measured during the synchronous period, and the optimal resilience is $n = 3f + 1$. Recent works have used threshold cryptography in order to achieve quadratic complexity in certain optimistic scenarios [2, 6, 22, 28, 37].

Because attackers may cause communication delays, an even more robust approach is to consider a fully asynchronous model. But in this model, BA cannot be solved deterministically [20]. Although the complexity lower bound does not apply to randomized algorithms, until fairly recently, randomized solutions also required (expected) $O(n^2)$ word complexity [5, 13, 29, 33].

A few recent studies have used randomness to circumvent Dolev and Reischuk's lower bound and provide BA solutions with sub-quadratic word complexity [2, 6, 16, 21, 23, 25, 30, 31, 35] in all three models. There are two approaches to using probability. The first is weakening the problem guarantees to probabilistic ones. Works in this vein usually utilize committee sampling, assume an adaptive adversary, and provide probabilistic safety and liveness [16, 21, 23, 30]. The second considers models in which deterministic BA solutions are possible, and designs protocols where the *expected* complexity is sub-quadratic [2, 6, 25, 31, 35]. The latter works are resilient only to a static adversary whereas the former tolerate a dynamic one.

Each work in the field examines BA through a slightly different lens: BA can be solved in models varying in their timing assumptions, adversarial behaviors, and properties of their outcomes. This work overviews the latest results on the subject. We point out common patterns in BA algorithms and highlight novel techniques. While doing so, we also identify gaps in existing works and raise open questions for future work. We start by defining the BA problem and describing possible models in Section 2. In Section 3, we cover works for deterministic BA, and in Section 4 for the randomized problem. We end with some open questions in Section 5.

2 Models and Definitions

We discuss different variations of the *Byzantine Agreement (BA)* problem also known as *consensus*. In this paper, we use the terms consensus and BA interchangeably. In this problem, a set Π of n processes attempt to reach a common decision upon one of the values initially suggested by the processes in Π. Different models affect the problem's difficulty. The connecting thread of our discussions is the presence of up to f Byzantine processes in Π. These processes may deviate arbitrarily from the protocol, possibly being corrupted and controlled by an *adversary*. Processes in Π that are not corrupted are called *correct* processes. Let us define the problem formally.

Definition 1 (Byzantine Agreement). *In Byzantine Agreement, each correct process $p_i \in \Pi$ proposes an input value v_i and decides on an output value $decision_i$ s.t. the following properties hold:*

Validity *If all correct processes propose the same value v, then all correct processes decide v.*

Agreement *No two correct processes decide differently.*

Termination *Every correct process eventually decides.*

Slight changes to the definition above give rise to several variations of the problem, applicable to different settings and areas of interest. For example, the problem as defined above does not make any assumption on processes' input values, and is known as *multi-value* BA. If we were to require that v_i is a binary value, the problem would be *binary BA*. The validity condition specified above is known as *strong unanimity*, and it might be more suitable for binary BA (since there are only two possible values).

For multi-value BA, a validity condition named *external validity* was proposed by Cachin et al. [12]. In this condition, there is some global predicate $validate(v) \in \{true, false\}$ that can be computed locally. If a correct process decides a value v, then $validate(v) = true$. The external validity condition is more suitable for multi-value BA, and is used in recent multi-value BA algorithms such as [4, 12, 37]. An example for external validity applicability can be found in various blockchains, where the predicate is verifying that the decided blocks meet the criteria of an eligible block to be

included in the blockchain (valid signatures on the transactions, no double-spend, etc.). We note that the Dolev-Reischuk lower bound [17] does not apply to the external validity BA variant, and to the best of our knowledge, no similar lower bound was proven for this case.

Definition 1 introduces *deterministic BA*, which requires that the properties hold in all runs of the protocol. In contrast, in *randomized BA*, one or more of the properties is ensured only with high probability (WHP). In some cases, randomized BA requires deterministic validity and agreement while allowing termination with probability 1. Other works [16, 21, 23] solve *BA with high probability*, where all properties are assured with high probability.

Timing assumptions   The BA problem(s) can be examined under different timing assumptions. In a *synchronous* network, there is a bound δ on the time it takes a message to reach its destination. In the *asynchronous* model, we assume nothing about communication times, so message delays are unbounded. *Eventual synchrony* is a middle ground where the system first behaves asynchronously, and eventually becomes synchronous. The asynchronous period is unbounded but it is guaranteed that at some point in time, called the *Global Stabilization Time (GST)*, the system becomes stable.

Adversary models   The adversary may corrupt up to f processes in the system. It is known that to solve BA in a synchronous model with strong unanimity, n must be at least $2f + 1$. Under eventual synchrony, the condition is stronger, and $n \geq 3f + 1$ is required [19].

The number of Byzantine processes corrupted by the adversary, known as the *resilience* of the protocol, is not the only concern for the protocol designer. Another important aspect is when and how the adversary controls them. We distinguish between a *static* adversary and an *adaptive* one. The static adversary decides which processes to corrupt before the execution begins, without prior knowledge of processes' initial values or coin flips. An adaptive adversary, on the other hand, can corrupt processes at any time, as long as it does not corrupt more than the threshold f.

Getting into a higher resolution, we can also examine the information available to the adversary while it makes its decisions and its power to prevent information flow in the system. In a synchronous model, one considers *"after-the-fact" removals* [1]. An adversary that is capable of after-the-fact removals can first observe messages sent by processes and then remove these messages from the network by corrupting their senders. It was shown that no after-the-fact removals are required for achieving sub-quadratic word complexity [1]. The idea of preventing the adversary from "front running" messages sent by processes before corrupting their senders can also be formulated in the asynchronous model. It can be assumed either explicitly [16] or implicitly by using a separate key to encrypt each message and atomically deleting the key upon sending a message signed by it [9, 21]. Additionally, one can use the notion of causality to define a *delayed-adaptive* adversary [16]. This adversary can use the contents of a message m sent by a correct process for scheduling a message m' only if m causally precedes m'.

Resilience   The resilience is the upper bound on the number of Byzantine processes in which the problem is still solvable. In eventual synchrony and asynchronous networks, the optimal resilience is $f < n/3$, independent of the validity conditions or the adversarial model described above. In the synchrony model, optimal resilience with strong unanimity is $f < n/2$ [28]. With external validity, the optimal resilience in synchronous networks is $f < n$.
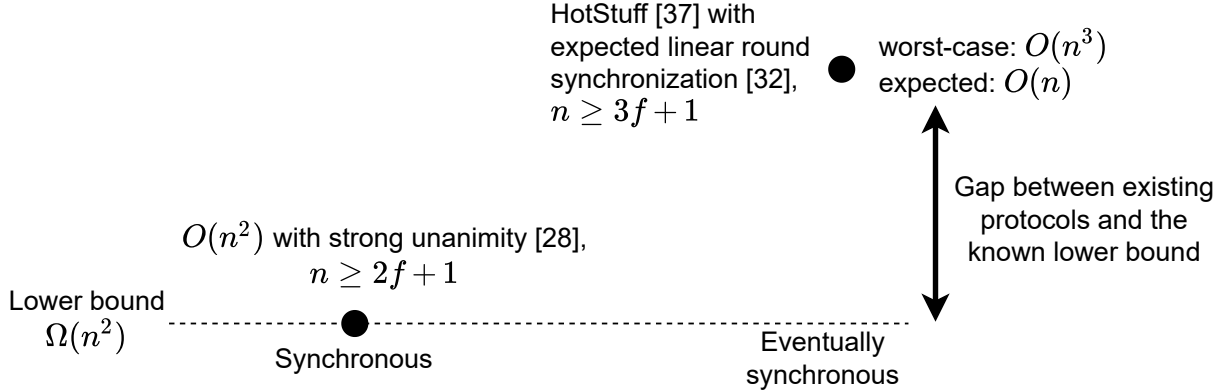
Figure 1: Illustration of the gap between existing BA protocols in the eventual synchrony and synchrony models and the known lower bound.

Complexity of Byzantine agreement   The way we measure complexity depends on the timing assumptions.

Synchrony Under synchrony assumptions, the *worst-case* word complexity is defined as the maximum number of words all the correct processes send under any possible adversarial behavior allowed by the model. This is counted from the beginning of the run and until the last correct process reaches a decision.

Eventual synchrony In the eventual synchrony model, the definition is usually identical to the above, except that the number of words is measured from GST and until a decision is reached. Sometimes, the complexity is defined by observing synchronous runs.

Asynchrony Since asynchronous deterministic consensus is not achievable even in the presence of at least one failure, there is no way to bound the worst-case complexity. Instead, we consider the *expected* complexity. The expectation is over the distribution of randomness used to solve BA with probability 1, assuming that the adversary takes the actions that lead to the worst complexity under this distribution.

3   Deterministic Consensus

In deterministic consensus protocols, the liveness property, i.e., the property that all processes eventually decide, is guaranteed. The seminal result of FLP [20] showed that deterministic consensus under asynchrony assumptions in the presence of at least one faulty process cannot be achieved. Thus, deterministic consensus protocols assume either a completely synchronous network during the entire run, or they assume sufficiently long times in which the network is synchronous, that are used to make progress in the consensus protocol.

In recent years, an increasing number of works such as [2, 6, 22, 28, 37] have used threshold cryptography to reduce the word complexity of BA protocols. This approach works as follows: A K-of-n threshold signature is used to create a signature on some message m. Each of the n processes

has a secret key, which it uses to create a unique share of the signature. Any K shares can be combined into a valid cryptographic signature on m. Each individual share and the combined signature consist of a single word. In BA protocols, usually all parties send a share of a message m to a process p, and once p has K shares (usually $K = f + 1$ or $K = 2f + 1$), it combines them into a signature and sends it to all processes. Thus, p can prove that a certain threshold of processes signed m with only $O(n)$ complexity.

Dolev and Reischuk proved in the early 1980s [17] that the lower bound for BA in the worst-case is at least $\Omega(n^2)$ messages per decision. To the best of our knowledge, until recently no deterministic BA protocol in the synchronous model or the eventual synchronous model met this lower bound with optimal resilience.

Recently, Momose and Ren [28] showed a protocol for BA with strong unanimity with $O(n^2)$ worst-case word complexity in the synchronous model with optimal resilience. Their protocol is based on threshold cryptography and ideas from [8], which also solves the problem with $O(n^2)$ communication complexity but for $f < n/3$, i.e., sub-optimal resilience. While there are synchronous BA protocols with external validity such as [3], to the best of our knowledge, there is no known protocol with the same resilience or better and $O(n^2)$ message complexity for solving BA with external validity.

In the eventually synchronous model, recent works such as Tendermint [25], Casper [11], and HotStuff [37] solve BA in $O(n)$ word complexity in optimistic scenarios, sometimes called the "happy path". In the worst-case, HotStuff requires $O(n^3)$ communication (after GST). Specifically, HotStuff's complexity is $O(n)$ whenever all correct processes are synchronized to the same round and follow a correct leader in this round. Achieving this is known as the round synchronization problem [32]. In HotStuff, the details of the round synchronization are abstracted away into a module named a Pacemaker, whose implementation is left unspecified. LibraBFT [6], which is based on HotStuff, implements the Pacemaker with cubic complexity in the worst-case.

A recent work by Naor and Keidar [32], which is based on [31], uses randomization to solve round synchronization with expected $O(n)$ word complexity in the ES setting under a static adversary. By combining this protocol with HotStuff, BA can be solved with expected $O(n)$ in the average case, but with $O(n^3)$ word complexity in the worst-case. It is still unknown if in the ES model under an adaptive adversary, BA can be solved with $O(n^2)$ word complexity in the worst-case.

The gap between known results and the lower bound for deterministic BA is illustrated in Figure 1.

4 Randomized Consensus

In randomized BA algorithms, the execution depends on local coin tosses at processes. While a simple local coin is a basic building block used in traditional works [7, 10], simply tossing coins may result in an exponential expected time to reach agreement. A more efficient approach is to utilize some sort of local randomness in order to create a *shared coin*. Processes communicate with each other to try to agree on the coin toss, and the coin has some well-defined *success rate* [13, 14, 21, 24, 33], namely a positive probability for all to agree on a random bit.

In previous sections, we mentioned Dolev and Reischuk's renowned lower bound, which indicates the necessity of quadratic communication [17] in deterministic settings. Can we do better by using randomized algorithms? The answer depends on the adversary. First, if we assume an adaptive adversary that is capable of after-the-fact removal, the answer is no. Abraham et al. [1] recently

extended the classic bound and proved that any randomized BA protocol under such an adversary must incur quadratic communication in expectation. Furthermore, a number of works [1,9,34] have proved that for subquadratic BA some form of setup assumption is required.

On the other hand, some algorithms that assume a restricted adversary and trusted setup do break the barrier of quadratic communication, solving binary BA. A common way of reducing algorithms' communication complexity is using committee sampling. With this technique, each step of the protocol is executed by a subset of the processes. That is, a process that is sampled to a committee related to a specific message sends it, whereas a process that is not sampled does nothing. In the presence of an adaptive adversary, we must ensure that the sampling for each committee cannot be anticipated by the adversary. To this end, committees are independent of each other, and messages are sent to all processes.

The first sub-quadratic algorithm was introduced by King and Saia [23] in the full information synchronous model, where there are no restrictions on the adversary's computational power. While they manage to achieve $\widetilde{O}(n\sqrt{n})$ word complexity, their work achieves very low resilience ($n > 400f$).

By using cryptographic abstractions and assuming a computationally bounded adversary, it is possible to achieve both high resilience and low communication complexity. This has been done by using *verifiable random functions* (VRFs) [27] to elect committees via a procedure known as *cryptographic sortition* [21]. A VRF is a pseudorandom function that provides a proof of its correct computation. Based on a local VRF computation, processes can choose committee members without communicating with each other. Each process runs a local procedure, independent of other processes, which checks whether it is elected. If selected, it is provided with a proof that can be used as a certificate to others. The unpredictability of the selection is a protective measure against an adaptive adversary that seeks to corrupt the elected members.

An adversary can identify correct processes that participate in a committee once they send messages. An unrestricted adaptive adversary can then corrupt a committee member and schedule a new message to arrive before the message which was sent before the corruption. We recall that this scenario can be prevented by an explicit assumption [16], or implicitly in the so-called *erasure model* [9, 21], where correct processes are able to securely erase secret keys from memory. This matches the no after-the-fact lower bound in [1], as the adversary cannot conceal messages sent by processes that were not corrupted at the time of the sending.

Algorand [21] first defined and used cryptographic sortition. It solves BA WHP with sub-quadratic communication and a near-optimal resilience of $f = (\frac{1}{3} - \epsilon)n$, assuming a trusted *public key infrastructure* (PKI) in the erasure model and using VRFs to sample committees. Their initial work was presented in a synchronous model, and was followed by a solution assuming eventual synchrony [15]. In both cases, they rely on the timing assumption to make progress. Specifically, their shared coin implementation is based on the idea that every value sent by a correct process reaches all processes in the system before a timeout signals the end of a "round". This is clearly not the case in asynchronous settings.

Algorand's shared coin implementation is quite simple and based on VRF values. A simplified pseudo-code of it appears in Algorithm 1. In their coin, every process broadcasts a pseudo-random value originated at a VRF output. At the end of the communication round, the minimum value is evaluated. The chosen random bit is the minimum value's least significant bit. Whenever the minimum value is sent by a correct process, all processes receive it and result in the same coin toss. Since the VRFs are pseudo-random and verified, there is a probability of at least $\frac{2}{3}$ that a correct

process holds the minimum value and all correct processes obtain the same bit.

Algorithm 1 Algorand's Shared Coin Framework - coin toss in round r

---
1: $v_{min} \leftarrow \infty$
2: $v_i \leftarrow VRF_i(r)$
3: send v_i to all processes
4: **wait** until timeout
5: **for** all v_j received in round r **do**
6: $\quad$ $v_{min} \leftarrow \min\{v_{min}, v_j\}$
7: return $LSB(v_{min})$

Recently, the quadratic barrier was broken also for the asynchronous case. Cohen et al. [16] have solved BA WHP in a model that assumes a trusted PKI and a delayed-adaptive adversary that cannot "front run" messages sent by processes before corrupting their sender. A simplified pseudo-code of their shared coin appears in Algorithm 2. With no timing assumptions, the task of handling dynamically selected committees is more difficult and the notion of "timeout" loses its meaning. Instead, they condition the protocol's progress on a predefined parameter W. They present sufficient conditions on sampling, which ensure safety and liveness with high probability. Specifically, the conditions ensure that waiting for W messages allows progress WHP. Because the adversary can create disagreement by hiding messages of correct processes from each other, achieving a positive success rate is more challenging. It requires an additional communication round for "boosting" the presence of the values. In addition, their shared coin algorithm restricts the protocol's resilience to $n \approx 4.5f$. We note that the delayed adaptiveness of the adversary is necessary to ensure that it cannot discover who holds the global minimum before scheduling the messages. Without it, the adversary would be able to hide the minimum from some correct processes, creating disagreement.

Algorithm 2 Cohen et al.'s Shared Coin Framework - coin toss in round r

---
1: $v_{min} \leftarrow \infty$
2: $v_i \leftarrow VRF_i(r)$
3: send v_i to all processes
4: **wait** until receiving W messages
5: **for** all v_j received in the W messages **do**
6: $\quad$ $v_{min} \leftarrow \min\{v_{min}, v_j\}$
7: send v_{min} to all processes
8: **wait** until receiving W messages
9: **for** all v_j received in the W messages **do**
10: $\quad$ $v_{min} \leftarrow \min\{v_{min}, v_j\}$
11: return $LSB(v_{min})$

Blum et al. [9] have presented an algorithm that solves asynchronous BA with near-optimal resilience. Compared to Cohen et al., they strengthen the adversary, removing the delayed adaptiveness requirement, but also the trusted setup phase. Their trusted dealer fixes committees' members and coin tosses in advance. Furthermore, it distributes the tosses to the members before the execution starts using secret sharing. As in Algorand, they also work in the erasure model.

Algorithm 3 presents a simplified pseudo-code of their shared coin.

Algorithm 3 Blum et al.'s Shared Coin Framework - coin toss in round r

---
1: send predefined round r share to all processes
2: return constructed predefined round r secret

---

5 Open Questions

While in recent years there are significant improvements in the word complexity of Byzantine agreement, we identify some still unanswered questions:

- Is there a lower bound on the number of messages or signatures for BA with external validity, similar to Dolev-Reischuk's [17] $\Omega(n^2)$ lower bound?

- In the synchronous model, is there a multi-value BA protocol with external validity that has $O(n^2)$ word complexity in the worst-case?

- In the eventual synchrony setting, is there any BA protocol (binary or multi-value) that has $O(n^2)$ word complexity in the worst-case?

- Can sub-quadratic BA be solved with deterministic guarantees for some of the properties, specifically safety ones, while others hold WHP?

- In the asynchronous settings, is there a sub-quadratic multi-value BA protocol?

- Whilst existing asynchronous sub-quadratic BA protocols rely on guarantees that hold whp provided that the protocol is run a constant number of times, is it possible to design asynchronous sub-quadratic SMR?

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 *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 317–326, 2019.

[2] Ittai Abraham, Guy Golan-Gueta, and Dahlia Malkhi. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil. *CoRR*, abs/1803.05069, 2018.

[3] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync HotStuff: Simple and practical synchronous state machine replication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 106–118. IEEE, 2020.

[4] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Validated asynchronous byzantine agreement with optimal resilience and asymptotically optimal time and word communication. *CoRR*, abs/1811.01332, 2018.

[5] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

[6] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. State machine replication in the Libra Blockchain. *The Libra Assn., Tech. Rep*, 2019.

[7] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30. ACM, 1983.

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

[9] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. Cryptology ePrint Archive, Report 2020/851, 2020. https://eprint.iacr.org/2020/851.

[10] Gabriel Bracha and Sam Toueg. Resilient consensus protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 12–26. ACM, 1983.

[11] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[12] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology —CRYPTO 2001*, pages 524–541. Springer Berlin Heidelberg, 2001.

[13] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in Constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, 2005.

[14] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *STOC*, volume 93, pages 42–51. Citeseer, 1993.

[15] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. Algorand agreement: Super fast and partition resilient byzantine agreement. *IACR Cryptology ePrint Archive*, 2018:377, 2018.

[16] Shir Cohen, Idit Keidar, and Alexander Spiegelman. Not a coincidence: Sub-quadratic asynchronous byzantine agreement whp. *arXiv preprint arXiv:2002.06545*, 2020.

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

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

[19] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[20] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[21] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.

[22] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.

[23] Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. *Journal of the ACM (JACM)*, 58(4):1–24, 2011.

[24] Valerie King and Jared Saia. Byzantine agreement in polynomial expected time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 401–410. ACM, 2013.

[25] Jae Kwon. Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1(11), 2014.

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

[27] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 120–130. IEEE, 1999.

[28] Atsuki Momose and Ling Ren. Optimal communication complexity of byzantine consensus under honest majority. *arXiv preprint arXiv:2007.13175*, 2020.

[29] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous binary byzantine consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time. *Journal of the ACM (JACM)*, 62(4):31, 2015.

[30] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[31] Oded Naor, Mathieu Baudet, Dahlia Malkhi, and Alexander Spiegelman. Cogsworth: Byzantine view synchronization. In *Proceedings of the Cryptoeconomic Systems Conference (CES'20)*, 2020.

[32] Oded Naor and Idit Keidar. Expected Linear Round Synchronization: The missing link for Linear Byzantine SMR. In *34th International Symposium on Distributed Computing (DISC)*, 2020.

[33] Michael O Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 403–409. IEEE, 1983.

[34] Matthieu Rambaud. Lower bounds for authenticated randomized byzantine consensus under (partial) synchrony: The limits of standalone digital signatures.

[35] Alexander Spiegelman. In search for a linear byzantine agreement. *arXiv preprint arXiv:2002.06993*, 2020.

[36] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[37] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.