Byzantine Agreement with Optimal Resilience via Statistical Fraud Detection

SHANG-EN HUANG, SETH PETTIE, and LEQI ZHU, University of Michigan, USA

Since the mid-1980s it has been known that Byzantine Agreement can be solved with probability 1 asynchronously, even against an omniscient, computationally unbounded adversary that can adaptively *corrupt* up to $f < n/3$ parties. Moreover, the problem is insoluble with $f \geq n/3$ corruptions. However, Bracha's [13] 1984 protocol (see also Ben-Or [8]) achieved $f < n/3$ resilience at the cost of *exponential* expected latency $2^{\Theta(n)}$, a bound that has *never* been improved in this model with $f = \lfloor (n-1)/3 \rfloor$ corruptions.

In this article, we prove that Byzantine Agreement in the asynchronous, full information model can be solved with probability 1 against an adaptive adversary that can corrupt $f < n/3$ parties, while incurring only *polynomial latency with high probability*. Our protocol follows an earlier polynomial latency protocol of King and Saia [33, 34], which had *suboptimal* resilience, namely $f \approx n/10^9$ [33, 34].

Resilience $f = (n-1)/3$ is uniquely difficult, as this is the point at which the influence of the Byzantine and honest players are of roughly equal strength. The core technical problem we solve is to design a collective coin-flipping protocol that *eventually* lets us flip a coin with an unambiguous outcome. In the beginning, the influence of the Byzantine players is too powerful to overcome, and they can essentially fix the coin's behavior at will. We guarantee that after just a polynomial number of executions of the coin-flipping protocol, either (a) the Byzantine players fail to fix the behavior of the coin (thereby ending the game) or (b) we can "blacklist" players such that the blacklisting rate for Byzantine players is at least as large as the blacklisting rate for good players. The blacklisting criterion is based on a simple statistical test of *fraud detection*.

CCS Concepts: • **Theory of computation** → **Distributed algorithms**; • **Mathematics of computing** → **Probabilistic inference problems;**

Additional Key Words and Phrases: Byzantine agreement, asynchronous network, full information model, strong adversary, fraud detection.

ACM Reference Format:
Shang-En Huang, Seth Pettie, and Leqi Zhu. 2024. Byzantine Agreement with Optimal Resilience via Statistical Fraud Detection. *J. ACM* 71, 2, Article 12 (April 2024), 37 pages. https://doi.org/10.1145/3639454

1 INTRODUCTION

In the Byzantine Agreement problem [35, 38], n players begin with input values in $\{-1, 1\}$ and each must *decide* an output value in $\{-1, 1\}$ subject to:

Agreement. All uncorrupted players must **decide** the same value (and only that value).
Validity. If all uncorrupted players **decide** v, then at least one such player had v as its input.

This article is derived from extended abstracts presented at STOC 2022 [28] and SODA 2023 [29]. This work was supported by NSF Grants CCF-1815316 and CCF-2221980.
Authors' address: S.-E. Huang, S. Pettie, and L. Zhu, University of Michigan, Computer Science and Engineering, 2260 Hayward St., Ann Arbor, Michigan 48109, USA; e-mails: huangaul@bc.edu, pettie@umich.edu, zhu.jimmy@gmail.com.

ACM 0004-5411/2024/04-ART12
https://doi.org/10.1145/3639454

Termination. Each uncorrupted player terminates the protocol with probability 1.

The difficulty of this problem depends on the *strength* of the adversary and assumptions on the communication medium. We consider a standard asynchronous model of communication against a *strong* adversary. Each player can send point-to-point messages to other players, which can be *delayed* arbitrarily by the adversary, but not dropped or forged. In addition, the adversary is aware of the internal state of every player, is computationally unbounded, and may adaptively *corrupt* up to f players; these are also known as *Byzantine* players. Once corrupted, the behavior of a player is arbitrary, and assumed to be controlled by the adversary. Following Ben-Or [8], Bracha [13] proved that Byzantine Agreement can be solved in this model when $f < n/3$. The protocols of Ben-Or and Bracha are not entirely satisfactory because they have latency *exponential* in n. (See Section 2.1 for the definition of *latency* in the asynchronous model.)

The Byzantine Agreement problem has been solved satisfactorily in a stronger communication model or against a weaker adversary than the one we assume.

Synchronized Communication. Lamport et al. [35] proved that if communication occurs in *synchronized* rounds, Byzantine Agreement can be obtained deterministically in $f + 1$ rounds, where $f < n/3$. Fischer and Lynch [22] proved that round complexity $f + 1$ is optimal. The communication complexity of [35] is exponential, and was later reduced to polynomial by Garay and Moses [25]. Dwork et al. [19] developed agreement protocols under weakly synchronous models.

Impossibility Results. Fischer et al. [24] proved that the problem cannot be solved deterministically, in an *asynchronous* system in which just *one* player is subject to a crash failure. This result is commonly known as *FLP Impossibility*. Thus, to solve Byzantine Agreement we must assume some level of synchronization *or* randomization. Even with randomization, the problem is insoluble when $f \geq n/3$. The proof of this result is straightforward in the asynchronous model [14, Theorem 3] and more complicated in the synchronized model [23, 38].

Cryptographic Solutions. Cryptography becomes useful against a *computationally bounded adversary*. Byzantine Agreement can be solved against such an adversary controlling $f < n/3$ players, with $O(1)$ latency.[1] [15, 16]; see also [1, 7, 11, 21].

Non-Adaptive Adversaries. The ability to *adaptively* corrupt players is surprisingly powerful. Goldwasser, Pavlov, and Vaikuntanathan [26], improving [10], considered a *synchronized*, full information model in which the adversary corrupts up to f players up front, that is, it is *non-adaptive*. They proved that Byzantine Agreement can be solved with resiliency $f < n/(3+\epsilon)$ in $O(\log n/\epsilon^2)$ rounds. See Chor and Coan [17] for prior results in similar adversarial models [17].

Kapron et al. [31] developed a Byzantine Agreement protocol in the *asynchronous*, full information model, in which the adversary corrupts $f < n/(3+\epsilon)$ players non-adaptively. Their protocol has different parameterizations, and can achieve agreement in quasipolynomial latency with probability $1 - 1/\text{poly}(n)$, or polylogarithmic latency with probability $1 - 1/\text{poly}(\log n)$. When these protocols err, they do not satisfy the **Agreement** and **Termination** criteria, and may deadlock or terminate without agreement.

Limits of Fully Symmetric Protocols. Lewko [36] proved that in the asynchronous, full information model, a certain class of "fully symmetric" Byzantine Agreement protocols has latency $2^{\Omega(n)}$ when $f = \Theta(n)$. This class was designed to capture Ben-Or [8] and Bracha [13] but is broader. Protocols in this class make state transitions that depend on the *set* of validated messages received, but may not take into account the transaction history of the sender. In retrospect, Lewko's result

[1]This assumes that RSA encryption cannot be broken by a polynomially bounded adversary.

can be seen as justifying two strikingly different approaches for improving [8, 13] in the asynchronous, full information model. The first is to break symmetry by having the players take on different roles: this is necessary to implement Feige's *lightest bin* rule and other routines in Kapron et al.'s [31] protocol. The second is to stay broadly within the Ben-Or–Bracha framework, but periodically *blacklist* players after accumulating statistical evidence of fraudulent coin flips in their transaction history. This is the approach taken by King and Saia [33, 34] and the present work.

Fraud Detection. King and Saia [33, 34] presented two Byzantine Agreement protocols with polynomial latency based on fraud detection and blacklisting. The first uses *exponential* local computation and is resilient to $f < n/400$ adaptive corruptions. The second uses polynomial local computation and is resilient to $n/(0.87 \times 10^9)$ adaptive corruptions. The extended abstracts of this work [28, 29] presented polynomial time and latency protocols resilient to $f < n/4$ [28] and the optimal $f < n/3$ [29] number of Byzantine failures. The statistical tests used by these protocols focus on positive correlations among Byzantine players [28] and negative correlations between Byzantine and honest players [29]; see Section 3.2.[2]

1.1 New Results

One feature of the asynchronous model is that every player must perpetually entertain the possibility that f players have *crashed* and will never be heard from again. Thus, when $n = 3f + 1$, the number of fully participating players at any stage is $n - f = 2f + 1$ and up to f of these players may be corrupt! Among the set of *participating* players, the good players hold the thinnest possible majority: $f + 1$ vs. f.

We develop a special coin-flipping protocol to be used in Bracha's framework [8, 13] when the corrupt and non-corrupt players have roughly equal influence. Initially, all players have *weight* 1. The coin-flipping protocol has the property that if the corrupt players repeatedly foil its attempts to flip a global coin, then we can fractionally *blacklist* players (reduce their weights) in such a way that the blacklisting rate for good players is only infinitesimally larger than the blacklisting rate for corrupt players. Specifically, we guarantee that among any $n - f = 2f + 1$ participating players, the total weight of the good players minus the total weight of the corrupt players is bounded away from zero. Eventually, all corrupt players have their weights reduced to zero, meaning they have no influence over the global coin protocol. At this point, the *scheduling* power of the adversary is still strong, but insufficient to fix the behavior of the global coin for much longer. Agreement is reached in a polynomial number of iterations of Bracha's algorithm, with high probability.

The final result is a randomized f-resilient Byzantine Agreement protocol with latency $\tilde{O}(n^4\epsilon^{-8})$, where $n = (3+\epsilon)f$ and $\epsilon \geq 1/f$. In other words, the latency ranges between n^4 and n^{12}, depending on ϵ. This latency bound holds in expectation and *with high probability*, meaning probability $1-n^{-c}$ for any desired constant c. See Table 1.

1.2 Organization of the Article

In Section 2 we review the formal asynchronous distributed model (Section 2.1), Bracha's Reliable-Broadcast primitive (Section 2.2), and his Byzantine Agreement protocol [13] (Section 2.3), which reduces the problem to collective coin-flipping.

In Section 3, we review related work on collective coin-flipping, and describe King and Saia's coin-flipping method, which is based on a shared *blackboard* protocol. In Section 3.1, we give specifications for an improved Iterated-Blackboard protocol. We outline the fraud detection

[2]The Byzantine agreement protocols of [1, 7] also employ a form of blacklisting ("shunning") whenever a party detects that another has not followed a certain cryptographic protocol.

Table 1. Randomized Byzantine Agreement Protocols in the *Asynchronous*, *Full Information* Model against an *Adaptive* Adversary

Citation	Resilience	Latency	Local Computation per Message
Ben-Or [8]	$f < n/5$	$2^{\Theta(n)}$	$\text{poly}(n)$
	$f = O(\sqrt{tn})$	$\exp(t^2)$	$\text{poly}(n)$
Bracha [13]	$f < n/3$	$2^{\Theta(n)}$	$\text{poly}(n)$
King & Saia [33, 34]	$f < n/400$	$\tilde{O}(n^{5/2})$	$\exp(n)$
	$f < n/(0.87 \times 10^9)$	$O(n^3)$	$\text{poly}(n)$
new	$f < n/(3+\epsilon)$	$\tilde{O}(n^4/\epsilon^8)$	$\text{poly}(n)$

Here $\epsilon \geq 1/f = \Omega(1/n)$.

mechanism of [28] (Section 3.2), which has resilience $f < n/4$, then consider several (failed) attempts to increase it to the optimum $f < n/3$ (Section 3.3). The specific nature of these failed attempts highlights difficulties of Byzantine Agreement in asynchronous networks and motivates certain design choices in our final protocol that might seem unjustified at first.

Section 4 begins with a high-level description of our new Byzantine-Agreement protocol (Algorithm 3) and its key invariants. It is based on a new collective coin-flipping protocol Coin-Flip (Algorithm 4) presented in Section 4.2, which uses the Iterated-Blackboard protocol. If the adversary foils its operation, we will see *negative correlations* between pairs of good and bad players; these are analyzed in Section 4.3. The blacklisting procedure is presented in Section 4.4. It is based on a fractional maximal matching algorithm called Rising-Tide applied to a complete graph weighted according to pairwise correlations. Section 4.5 bounds numerical disagreements between players based on their different histories, and establishes high probability bounds on Byzantine-Agreement reaching agreement with polynomial latency.

The analysis of the protocol depends on two theorems, each of which can be understood in isolation. Theorem 6 specifies the properties and efficiency of the Iterated-Blackboard primitive. The implementation and analysis of Iterated-Blackboard (Algorithm 7) are presented in Section 5. Theorem 17 claims that Rising-Tide is continuous Lipschitz, that is, small numerical perturbations of the input cause small numerical changes in the output. It is proved in Section 6.

We conclude with some remarks and open problems in Section 7.

2 PRELIMINARIES

2.1 The Model

We assume a standard asynchronous message passing model. The formalization below follows Attiya and Welch [6, Chapter 2.1] and is equivalent to that of Lynch [37, Chapter IIB].

There are n *players* $p_1, \ldots, p_n$, and $2n^2$ *message buffers*, $\text{Out}_{i\to j}$ and $\text{In}_{j\to i}$, for all $i, j \in [n]$. All players are initially *good* (they obey the protocol) and the adversary can dynamically *corrupt* up to f players. A *Byzantine*/*corrupted* player is under complete control of the adversary and can behave arbitrarily. The adversary controls the pace at which progress is made by scheduling two types of *events*.

— A compute(i) event lets p_i process all messages in the buffers $\text{In}_{j\to i}$, deposit new messages in $\text{Out}_{i\to j}$, and change state.
— A deliver(i, j) event moves a message from $\text{Out}_{i\to j}$ to $\text{In}_{i\to j}$.

ALGORITHM 1: Reliable-Broadcast(p, ℓ)

1: **if** $\ell > 1$ **then wait** until Reliable-Broadcast$(p, \ell - 1)$ accepts $m_{p,\ell-1}$.
2: **if** I am player p **then** generate $m_{p,\ell}$ and send (init, $m_{p,\ell}$) to all players.
3: **wait** until receipt of one (init, $m_{p,\ell}$) message from p, or $(n + f + 1)/2$ (echo, $m_{p,\ell}$) messages, or $f + 1$ (ready, $m_{p,\ell}$) messages.
send (echo, $m_{p,\ell}$) to all players.
4: **wait** until the receipt of $(n + f + 1)/2$ (echo, $m_{p,\ell}$) messages or $f + 1$ (ready, $m_{p,\ell}$) messages.
send (ready, $m_{p,\ell}$) to all players.
5: **wait** until receipt of $2f + 1$ (ready, $m_{p,\ell}$) messages.
accept $m_{p,\ell}$.

Note that the adversary may choose a malicious order of events but cannot, for example, misdeliver or forge messages. Every message must eventually be delivered, and every player i allowed to compute(i) infinitely often. The adversary is computationally unbounded and is aware, at all times, of the internal state of all players. Thus, cryptography is not helpful, but randomness is potentially useful, since the adversary cannot predict the outcome of future coin flips.

In this model, the *communication time* or *latency* of a protocol is defined w.r.t. a hypothetical timed execution in which all local computation occurs instantaneously and all messages are delivered within some latency in the interval $[0, \Delta]$. The latency of the protocol is L if all non-corrupt players finish by time $L\Delta$.

The parameter Δ is introduced simply to *define* latency. There is, in fact, no *a priori* upper bound on the delivery time of any message. Indeed, the adversary cannot increase the latency of the algorithm simply by forestalling the delivery of certain messages, as this *also* increases the empirical maximum delivery time Δ.

2.2 Reliable Broadcast

The goal of Reliable-Broadcast is to simulate a broadcast channel using the underlying point-to-point message passing system. In Byzantine Agreement protocols, each player initiates a series of Reliable-Broadcasts. Call $m_{p,\ell}$ the ℓth message broadcast by player p. In Theorem 1, the property that $m_{p,\ell}$ is only accepted after $m_{p,\ell-1}$ is accepted is sometimes called *FIFO broadcast*.

Theorem 1 (FIFO Broadcast; see [13]). *If a good player p initiates the Reliable-Broadcast of $m_{p,\ell}$, then all good players q eventually* accept *$m_{p,\ell}$. Now suppose that a corrupt player p does so and some good q* accepts *$m_{p,\ell}$. Then all other good q' will eventually accept $m_{p,\ell}$, and no good q' will accept any other $m'_{p,\ell} \neq m_{p,\ell}$. Moreover, all good players accept $m_{p,\ell-1}$ before $m_{p,\ell}$, if $\ell > 1$.*

The term *accept* has no semantics outside the guarantees of Theorem 1, that is, a message $m_{p,\ell}$ accepted by q will eventually be accepted by every good player.

Proof. According to Line 1 of Reliable-Broadcast, no message $m_{p,\ell}$ can be accepted until $m_{p,\ell-1}$ is accepted, if $\ell > 1$. This establishes the FIFO property. The other correctness properties follow from several claims.

We claim that if two good players q, q' send (ready, $m_{p,\ell}$) and (ready, $m'_{p,\ell}$) at Line 4, then $m_{p,\ell} = m'_{p,\ell}$. Suppose not; in particular, let (q, q') be the *first* pair to send conflicting ready messages $m_{p,\ell} \neq m'_{p,\ell}$. Due to (q, q') being first, it must be q and q' were spurred to send ready messages after receiving $(n+f+1)/2$ (echo, $m_{p,\ell}$) and (echo, $m'_{p,\ell}$) messages. Thus, at least $2((n+f+1)/2)-n \geq f+1$ players sent both q and q' conflicting (echo, $\cdot$) messages, and therefore some *good* player sent conflicting (echo, $\cdot$) messages in Line 3, which is impossible.

We now claim that if a good player q accepts $m_{p,\ell}$ then every good player eventually accepts $m_{p,\ell}$. It follows that q has already accepted $m_{p,1}, \ldots, m_{p,\ell-1}$ in Line 1. By induction, every other good player eventually accepts $m_{p,\ell-1}$. Before accepting $m_{p,\ell}$, q received at least $2f+1$ (ready, $m_{p,\ell}$) messages in Line 5, and hence at least $f+1$ from good players. These $f+1$ messages will eventually be delivered to all $n-f \geq 2f+1$ good players, causing all to send their own (ready, $m_{p,\ell}$) messages in Line 4 and eventually accept the same value.

If the sender p is good, then every good player will clearly eventually accept $m_{p,\ell}$. Moreover, as a consequence of the above claims, if p is corrupt, then it is impossible for good players to accept different messages $m_{p,\ell} \neq m'_{p,\ell}$. □

2.3 Validation and Bracha's Protocol

Consider a protocol Π of the following form. In each round r, each player reliably broadcasts its *state* to all players, waits until it has accepted and *validated* at least $n - f$ messages from round r, then processes all validated messages, changes its state, and advances to round $r + 1$. A good player *validates* a round-r state (message) $s_{q,r}$ accepted from another player q only if (i) it has validated the state $s_{q,r-1}$ of q at round $r - 1$, and (ii) it has validated $n - f$ messages that, if they *were* received by a correct q, would cause it to transition from $s_{q,r-1}$ to $s_{q,r}$. The key property of validation (introduced by [13]) is:

Lemma 2. *A good player p validates the message of another player q in an admissible execution α of Π if and only if there is an admissible execution β of Π in which q is a good player and the state of every other good player (including p) is the same in α and β with respect to their validated messages.*

To recap, reliable broadcast prevents the adversary from sending conflicting messages to different parties, that is, it is forced to participate as if the communication medium were a broadcast channel, albeit one where the time to receive broadcasts is irregular. The validation mechanism *forces* its internal state transitions to be consistent with the protocol Π. Note, however, that in general Π is *probabilistic* and validation permits a series of transitions that are logically possible but statistically unlikely. In summary, after validation, the following powers characterize the adversary.

Full Information and Scheduling. The adversary knows the internal state of all players and controls the order in which messages are delivered. It may delay messages.

Corruption and Coin Flipping. The adversary may *adaptively* corrupt up to f players as the execution of the protocol progresses. Once corrupted, a player *continues to follow protocol*, except that the adversary now chooses the outcomes of all of its coin flips.

Remark 1. One key implication of the *Corruption and Coin Flipping* simplification is that we usually do not need to differentiate between good and bad *players*. There are some players with *functioning* random number generators and at most f with *corrupt* random number generators, but both types of players follow protocol. Consider Lemma 3, which will be presented shortly. It refers to a supermajority of *players* holding a single value v^*, not *good players*, because the validation mechanism effectively prevents a bad player from broadcasting values that are inconsistent with the protocol. On the other hand, it says that *good players* will **decide** v^*. **Deciding** is a private action and it makes no sense to talk about the private actions of corrupt players. Validation only governs the *public* broadcasts of players.

Bracha's protocol improves the resilience of Ben-Or's protocol to the optimum $f < n/3$. Each player p initially holds a value $v_p \in \{-1, 1\}$. It repeats the same steps until it **decides** a value $v \in \{-1, 1\}$ (Line 8). As we will see, if some good player **decides** v, all good players will **decide** v in this or the following iteration. Thus, good players continue to participate in the protocol until all other good players have executed Line 8. Here, $\text{sgn}(x) = 1$ if $x \geq 0$ and -1 if $x < 0$.

ALGORITHM 2: Bracha-Agreement() *from the perspective of player p*

Require: $v_p \in \{-1, 1\}$.

1: **loop**
2: Reliable-Broadcast v_p and **wait** until $n - f$ messages are validated from a set of players S_p.
set $v_p \leftarrow \text{sgn}(\sum_{q \in S_p} v_q)$. ▷ $\text{sgn}(x) = 1$ if $x \geq 0$ and -1 otherwise.
3: Reliable-Broadcast v_p and **wait** until $n - f$ messages are validated.
if more than $n/2$ messages have some value v^* **then** set $v_p \leftarrow v^*$, **else** set $v_p \leftarrow \perp$.
4: Reliable-Broadcast v_p and **wait** until $n - f$ messages are validated.
let x_p be the number of $v^* \neq \perp$ messages validated by p.
5: **if** $x_p \geq 1$ **then**
6: set $v_p \leftarrow v^*$.
7: **if** $x_p \geq f + 1$ **then**
8: **decide** v^*.
9: **if** $x_p = 0$ **then**
10: $v_p \leftarrow$ Coin-Flip(). ▷ Returns value in $\{-1, 1\}$.

The correctness of Bracha-Agreement follows from Lemmas 3 and 4, and its efficiency from Lemma 5.

Lemma 3. *If a supermajority of $(n + f + 1)/2$ players hold the same value v^* at Line 2, then all good players will have **decided** v^* by the end of the loop.*

Proof. In Line 2 each player updates their value based on a majority vote of a set of $n - f$ players, presumably selected by the adversary. However, in any $n - f$ messages, there are at least $(n + f + 1)/2 - f = (n - f + 1)/2$ votes for v^*. Thus, all players set $v_p \leftarrow v^*$ in Line 2, broadcast v^* in Line 3 and retain it since $n - f > n/2$. They broadcast v^* again on Line 4 and then set $x_p = n - f$, so every player p **decides** v^* in Line 8. □

Lemma 4. *If any good player executes Line 8 and **decides** v^*, then all good players will have **decided** v^* by the end of the next iteration through the loop.*

Proof. After each player p executes Line 4 it becomes a member of one of the following three sets.

$$A_{\text{dec}} = \{p \ : \ x_p \geq f + 1\},$$
$$A_{\text{keep}} = \{p \ : \ x_p \in [1, f]\},$$
$$A_{\text{flip}} = \{p \ : \ x_p = 0\}.$$

Regardless of how messages are delivered, for any p and q, $|x_p - x_q| \leq f$, hence all players are in $A_{\text{dec}} \cup A_{\text{keep}}$ or $A_{\text{keep}} \cup A_{\text{flip}}$. If $A_{\text{dec}} \neq \emptyset$, then all players (good and bad) go into the next iteration holding v^*, and by Lemma 3, all good players **decide** v^* by the end of that iteration. (Note that the validation mechanism prohibits a bad player from switching its value from v^* to $-v^*$ at the next iteration, for no one will be able to validate a message "$-v^*$" broadcast in Line 2.) □

Lemma 5. *In any execution of one loop of Bracha-Agreement, the probability that a supermajority of $(n + f + 1)/2$ players adopt the same value is at least 2^{-n}.*

Proof. By Lemma 4, we can focus on the case where the population is $A_{\text{keep}} \cup A_{\text{flip}}$, that is, those in A_{keep} know the majority value v^* and keep it, and every good $p \in A_{\text{flip}}$ chooses v_p on the

basis of a coin flip. With probability at least 2^{-n}, every good $p \in A_{\text{flip}}$ chooses v^*, in which case a supermajority of $n - f \geq (n + f + 1)/2$ players hold v^*. If $A_{\text{keep}} = \emptyset$ then v^* is undefined, and any unanimous coin flip outcome (-1 or 1) among A_{flip} is acceptable. □

3 COLLECTIVE COIN-FLIPPING

The probability of achieving a supermajority of $(n + f + 1)/2$ in Line 10 of Bracha-Agreement is much higher when f is small, for example, $\Omega(\exp(-t^2))$ when $f = O(\sqrt{tn})$. However, the easiest way to achieve such a supermajority with probability 1/2 is for all players to somehow flip an unbiased common *global* coin. Rabin [39] and Toueg [42] solved Byzantine Agreement in constant rounds (in expectation) by assuming the existence of such a global coin, or, equivalently, a mechanism to distribute shared randomness to all players in advance. The protocols of King and Saia [33, 34] and those in Section 4 work in the framework of Bracha-Agreement, but substitute for Coin-Flip (Line 10) a collective coin flipping protocol that *aspires* to have two properties.

Property (i) all players agree on the *same* value returned by Coin-Flip(), and
Property (ii) the output of Coin-Flip() is close to unbiased.

The problem of flipping a bounded-bias coin against adversarial manipulation is well studied. The problem can be solved against surprisingly large coalitions of corrupt players [2, 3, 9, 12, 20, 27, 30, 40, 41]. This body of work assumes reliable communication (no dropped or delayed messages) and reliable computation (no crash failures). Aspnes [4] gave a lower bound that models aspects of an *adaptive* adversary in an *asynchronous* network. In his coin-flipping game, a vector of values $(v_1, \ldots, v_N)$ is generated as follows. Once $(v_1, \ldots, v_{i-1})$ are known, a random value v_i' is generated[3] and the adversary may set $v_i \leftarrow v_i'$ or *suppress* it, setting $v_i \leftarrow \perp$. The outcome of the coin flip is some function $g(v_1, \ldots, v_N) \in \{-1, 1\}$. If the adversary can suppress t values, then $N = \Omega(t^2)$ for g to have constant bias and $N = \Omega(t^2/\log^2 t)$ if the probability that $g = 1$ and -1 are both at least $1/\text{poly}(t)$. Aspnes [4] proved that this result implies $\tilde{\Omega}(n)$ latency lower bounds on Byzantine Agreement in the asynchronous model, which was improved to $\Omega(n)$ by Attiya and Censor-Hillel [5]. The moral of [4, 5] and related lower bounds against *adaptive* adversaries, such as Haitner and Karidi-Heller's [27], is that the aggregation function g that implements majority voting is at least close to optimal. However, Byzantine Agreement protocols against *non-adaptive* adversaries, such as [10, 26, 31] can afford to implement clever coin-flipping protocols that are not based exclusively on majority voting [20, 40].

The coin-flipping protocols of King and Saia [33], [28], and the one presented in Section 4 do not attempt to guarantee Properties (i) and (ii) immediately. Rather, after a sufficiently large number of invocations of Coin-Flip, if the adversary foils Properties (i) and (ii), it will leave behind enough statistical evidence that proves incriminating, allowing us to *blacklist* suspicious players, removing their explicit influence over subsequent calls to Coin-Flip. When all corrupt players are blacklisted, the adversary still has the power of scheduling, but this power is insufficient to significantly delay agreement.

The basis of King and Saia's [33] implementation of Coin-Flip is a shared *blackboard* primitive, whose resiliency was improved from $f < n/4$ to (optimal) $f < n/3$ by Kimmett [32]. A blackboard is conceptually an $m \times n$ matrix BB, initially all blank ($\perp$). The goal is to have each player i write m values successively to column i via Reliable-Broadcasts, and once the blackboard is full, to have all players *agree* on its contents. Because up to f players may crash, a *full* blackboard is one in which $n - f$ columns have m writes, and the remaining f columns may be *partial*. Due to the scheduling power of the adversary, every player p sees a slightly different version $\text{BB}^{(p)}$ of the

[3]The distribution and range of v_i' are arbitrary, and may depend on $(v_1, \ldots, v_{i-1})$.

"true" blackboard BB, which is derived by replacing the last write in some of the f partial columns with $\perp$. Thus, $\mathrm{BB}^{(p)}$ and $\mathrm{BB}^{(q)}$ differ in at most f entries.

In [33] the Coin-Flip routine is implemented as follows: every write to BB is a value in $\{-1, 1\}$ chosen uniformly at random. When p finishes participating in the construction of BB it has a view $\mathrm{BB}^{(p)}$ and sets the output of Coin-Flip to be $\mathrm{sgn}(\Sigma^{(p)})$, where $\Sigma^{(p)} = \sum_{j,q} \mathrm{BB}^{(p)}(j, q)$ is the sum of all entries, treating $\perp$ as zero. Note that whenever $\Sigma^{(p)} \notin [-f, f]$, p can be sure that Coin-Flip generates the same output for all players, even corrupt ones.[4]

In the context of [33], a happy outcome is when all players agree on the same coin flip value, and if $A_{\mathrm{keep}} \neq \emptyset$ is holding v^*, that value is v^*. The new Coin-Flip protocol developed in Section 4 is similar in spirit. It uses two blackboards, the first being used to generate a *bias*, which depends on A_{keep} and v^*, and the second is populated with random $\{-1, 1\}$ variables, as in [33]. In contrast to King and Saia [33] and our first extended abstract [28], *any* outcome where all players agree on the output of Coin-Flip is a happy outcome.

3.1 The Iterated Blackboard

We can of course execute the blackboard primitive iteratively [33], but two players may disagree on the contents of *each* blackboard in up to f cells. The Iterated-Blackboard protocol specified in Theorem 6 guarantees a stronger form of agreement. An iterated blackboard is an endless sequence $(\mathrm{BB}_1, \mathrm{BB}_2, \ldots)$ of blackboards, where BB_t is an $m(t)\times n$ matrix. After p observes a sufficient number of writes to BB_t, p fixes a view $\mathrm{BB}^{(p,t)} = (\mathrm{BB}_1^{(p,t)}, \ldots, \mathrm{BB}_t^{(p,t)})$ of the first t blackboards. It is guaranteed that $\mathrm{BB}_t^{(p,t)}$ and $\mathrm{BB}_t^{(q,t)}$ differ in at most f cells in partial columns; it is also guaranteed that $\mathrm{BB}^{(p,t)}$ and $\mathrm{BB}^{(q,t)}$ differ in at most f cells *in total*, over all t blackboards. In order to make this type of guarantee, during the construction of BB_{t+1}, p may record *retroactive updates* to an earlier $\mathrm{BB}_{t'}$, $t' \leq t$, so that $\mathrm{BB}_{t'}^{(p,t+1)}$ records some writes to cells that were still $\perp$ in $\mathrm{BB}_{t'}^{(p,t)}$. (The blackboard protocols of [32, 33] did not allow for retroactive updates. This feature was introduced in [28].)

We use the iterated blackboard for two different tasks. First, the outcome of a shared coin-flip is determined by the sum of entries in some BB_t, and for this reason it is important that any players p and q agree on this sum, up to $\pm f$. After a long series of calls to Coin-Flip that fail to bring Bracha-Agreement to an end, we blacklist some players based on the history $(\mathrm{BB}_1, \ldots, \mathrm{BB}_t)$, and here it is important that when players p and q *locally* calculate their blacklist, they come to numerically similar conclusions. This is more critical to our protocol than [33] since resiliency $f < n/3$ demands dramatically higher levels of historical agreement between good players.

Theorem 6. *An iterated blackboard* $(\mathrm{BB}_1, \mathrm{BB}_2, \ldots, \mathrm{BB}_\tau)$ *is a sequence of matrices,* BB_t *being an* $m(t) \times n$ *matrix with all cells initially* $\perp$*. It is constructed by calling Iterated-Blackboard*(1)*, . . ., Iterated-Blackboard*(τ)*. In the execution of Iterated-Blackboard*(t)*, every good player* p *fixes a view* $\mathrm{BB}^{(p,t)} = (\mathrm{BB}_1^{(p,t)}, \ldots, \mathrm{BB}_t^{(p,t)})$ *of the true blackboards* $(\mathrm{BB}_1, \ldots, \mathrm{BB}_t)$*. Iterated-Blackboard*$(t)$ *is resilient to* $f < n/3$ *Byzantine faults, and has the following properties.*

(1) Player i *writes values successively to cells in column* $\mathrm{BB}_t(\cdot, i)$ *and only player* i *may write values to this column. Thus, at all times,* $\mathrm{BB}_t(\cdot, i)$ *consists of a prefix of non-*$\perp$ *values and a suffix of* $\perp$*s.*

(2) When player p *fixes its historical view* $\mathrm{BB}^{(p,t)}$*, it contains all writes recorded in* $\mathrm{BB}^{(p,t-1)}$*, if* $t > 1$*. Moreover,* $\mathrm{BB}_t^{(p,t)}$ *contains at least* $n - f$ *full columns, with* $m(t)$ *writes, and at most* f

[4] In particular, in Line 2 of Bracha-Agreement, if a message v_q broadcast from q is purportedly the output of the last iteration's call to Coin-Flip(), player p will not validate it if it is *impossible* that $\mathrm{sgn}(\Sigma^{(q)}) = v_q$.

partial columns. It is guaranteed that for every $t' \in [t]$, $i \in [n]$, $r \in [m(t')]$,

$$\mathsf{BB}_{t'}^{(p,t)}(r,i) \in \{\mathsf{BB}_{t'}(r,i), \perp\}.$$

Moreover, there are at most f *tuples* (t', r, i) *such that for some* p, $\mathsf{BB}_{t'}^{(p,t)}(r,i) = \perp \neq \mathsf{BB}_{t'}(r,i)$. *All such tuples have distinct 3rd coordinates.*

(3) *If a player* q *writes any non-*$\perp$ *value to* BB_t, *then by the time any player* p *fixes* $\mathsf{BB}^{(p,t)}$, p *can locally reconstruct* q*'s view* $\mathsf{BB}^{(q,t-1)}$ *of the history up to blackboard* $t-1$.

(4) *The latency of constructing* $\mathsf{BB}^{(p,\tau)}$ *is linear in the total number of rows, namely* $O(\sum_{t \le \tau} m(t))$.

Remark 2. We use Theorem 6(3) to make the following simulation argument. If player q sets a certain variable α based on *its* view $\mathsf{BB}^{(q,t-1)}$ and subsequently writes anything to $\mathsf{BB}_{t'}$, $t' \ge t$, then once p fixes $\mathsf{BB}^{(p,t')}$, it learns $\mathsf{BB}^{(q,t-1)}$ and can therefore *simulate* q's computation of α.

3.2 Coin Flipping and Fraud Detection

The King-Saia [33, 34] protocol and the one presented in our first extended abstract [28] rely on the fact that the f corrupt players, being a small minority, must collectively generate coin flips whose sum is conspicuously large, as they must often *counteract* the coin flips of $n-2f$ good players.[5] At the end of the tth iteration of Bracha-Agreement, the [28, 33] protocols call a Coin-Flip procedure, which populates the blackboard BB_t with random $\{-1, 1\}$ coin flips. Define $X_i(t)$ to be the sum of the coin flips in $\mathsf{BB}_t(\cdot, i)$ generated by player i.[6]

At the very least, the adversary wants at least one player in A_{flip} to believe that the outcome of the global coin is $\sigma(t) = -v^*$ (i.e., the opposite value held by those in the A_{keep} population), which is called the *adversarial direction*. Let $\Sigma_G(t)$ and $\Sigma_B(t)$ be the sum of the good and bad (corrupt) coin flips written to BB_t. If $\operatorname{sgn}(\Sigma_G(t)) = \sigma(t)$ then the adversary is happy, and if $\operatorname{sgn}(\Sigma_G(t)) = -\sigma(t)$ then the adversary needs to counteract the good coin flips and get the total sum $\Sigma_G(t) + \Sigma_B(t)$ in the interval $[-f, f]$ in order for at least one player to believe the coin flip outcome (sign of the sum) is $\sigma(t)$. Thus,

$$|\Sigma_B(t)| \ge \max\{0, -\sigma(t)\Sigma_G(t) - f\}.$$

$\Sigma_G(t)$ is the sum of at least $m(n-2f) = \Omega(mn)$ coin flips; suppose for simplicity that $\Sigma_G(t)$ is the sum of *exactly* this many flips. When $m \gg n > 3f$, f is much smaller than the standard deviation of $\Sigma_G(t)$, so let us also ignore the "$-f$" term for simplicity. By symmetry, $\Sigma_G(t)$ is positive and negative with equal probability, so up to these simplifications, $\mathbf{E}[\Sigma_B(t)^2] \ge \frac{1}{2}\mathbf{E}[\Sigma_G(t)^2] = \frac{1}{2}m(n-2f)$.[7] On the other hand, if these f bad players *were* flipping fair coins, then $\mathbf{E}[\Sigma_B(t)^2] \le mf$.

The statistics tracked in [28] are pairwise *correlations* and individual *deviations* over a series of calls to Coin-Flip.

$$\operatorname{corr}(i,j) = \langle X_i, X_j \rangle = \sum_t X_i(t)X_j(t),$$

$$\operatorname{dev}(i) = \langle X_i, X_i \rangle = \sum_t (X_i(t))^2.$$

[5] Recall that the adversary can fail to deliver messages of up to f players in a timely fashion, so there can be as few as $f + (n-2f)$ players fully participating in any given blackboard/coin-flip.

[6] Recall that the players appear in $A_{\text{keep}} \cup A_{\text{flip}}$, where those in A_{keep} will keep the majority value $v^* \in \{-1, 1\}$, regardless of the outcome of Coin-Flip. Nonetheless, *every* player in $A_{\text{keep}} \cup A_{\text{flip}}$ participates in the Coin-Flip protocol.

[7] It is a small abuse of notation to measure the expectation of $\Sigma_B(t)^2$ since it has no well-defined distribution. The expectation is naturally w.r.t. *any* fixed adversarial strategy that convinces at least one player that the outcome of the global coin is $\sigma(t)$.

Note that $(\Sigma_B(t))^2$ can be decomposed into terms that contribute to corr(i,j) scores ($X_i(t)X_j(t)$, $i \neq j$) and dev(i) scores ($X_i(t)^2$). When $f < n/4$ there is a *gap* between $\frac{1}{2}m(n-2f)$ and mf, which implies that after a sufficient number of iterations, either some bad player i has an unusually large dev(i) score, or two bad players i, j have an unusually large corr(i,j) score. *Unusually large* here means one beyond what any good players flipping fair coins could generate, with high probability. If corr(i,j) is unusually large, it follows that *at least one* of i, j must be bad. The protocol introduced in [28] blacklists pairs of players with high correlation scores and individual players with high deviation scores.

3.3 Misfires and Dead Ends

Resilience $f < n/4$ is good, but we want to push it to the absolute maximum $f < n/3$. In this section, we illustrate why certain natural approaches to achieving optimal resiliency fail in subtle ways. This leads us to a simple approach that works, which is outlined in Section 3.4 and fully developed in Section 4.

As f tends toward $n/3$, many natural statistics worth tracking lose traction, and "$n/3$" is the point at which coin-flipping games become perfectly balanced between the influence of $n-2f$ good and f bad players. For example, when $f = n/4$, $mf = \frac{1}{2}m(n-2f)$, and bad players may not be detected by looking for large dev(i) or corr(i,j) scores alone. When $n = 3f+1$, we can assume that $n - f = 2f+1$ players fully participate in the coin-flipping protocol, at least $f+1$ of which are good and at most f of which are bad. To illustrate why this is a uniquely difficult setting to perform fraud detection, consider a simple **Mirror-Mimic** strategy deployed by the adversary.

Mirror-Mimic Strategy. When $\text{sgn}(\Sigma_G(t)) = -\sigma(t)$, the adversary chooses its coin-flips so that $\Sigma_B(t) = -\Sigma_G(t)$ (mirror). When $\text{sgn}(\Sigma_G(t)) = \sigma(t)$, it sets $\Sigma_B(t) = \Sigma_G(t)$ (mimic). There is some flexibility in the mirror case as it only needs $\Sigma_B(t) + \Sigma_G(t)$ to hit the interval $[-f, f]$. In any case, we do not expect to see large good-good corr(i,j) scores outside of random noise, nor large bad-bad correlations since they are mirroring/mimicking the distribution of good players. Because the mirror/mimic cases occur about equally often, the aggregate positive correlations between good and bad players in the mimic case and negative correlations between good and bad players in the mirror case cancel out. Thus, against the mirror-mimic adversary, tracking pairwise correlations *alone* seems insufficient to efficiently detect fraud.

σ-Correlation. When we attempt to flip a global coin, the good players are generally unaware of the adversarial direction $\sigma(t)$[8] but we *can* ensure that $\sigma(t)$ eventually becomes known, and can estimate *σ-correlation* over the long term. In the context of Bracha-Agreement, $\sigma(t)$ should be defined as:

$$\sigma(t) = \begin{cases} -v^* & \text{if } A_{\text{keep}} \neq \emptyset \text{ keeps the majority value } v^* \in \{-1, 1\}, \\ 0 & \text{if } A_{\text{keep}} = \emptyset. \end{cases}$$

Define the σ-correlation score as:

$$\sigma\text{-corr}(i) = \langle \sigma, X_i \rangle = \sum_t \sigma(t) X_i(t).$$

Note that good players flip fair coins, so values of σ-corr(i) that are inconsistent with random noise should indicate that i is corrupt. However, it is rather easy for the adversary to keep σ-corr(i) scores close to zero for corrupt players as well. Regardless of $\text{sgn}(\Sigma_G(t))$, it should set $\Sigma_B(t) = -\Sigma_G(t)$, or at least put $\Sigma_B(t) + \Sigma_G(t)$ in the interval $[-f, f]$. One may easily verify that this strategy is

[8](if they all knew what it was, there would be no need to flip a coin).

consistent with mirror-mimic when $\text{sgn}(\Sigma_G(t)) = -\sigma(t)$, but that it prescribes exactly the *opposite* behavior when $\text{sgn}(\Sigma_G(t)) = \sigma(t)$! In fact, there is no general strategy for setting $\Sigma_B(t)$ as a function of $\Sigma_G(t)$ and $\sigma(t)$ that keeps *all* $\text{corr}(i,j)$ and $\sigma\text{-corr}(i)$ scores close to zero.[9]

Tracking $\text{corr}(i,j)$ and $\sigma\text{-corr}(i)$ scores seems to be a winning combination, that will eventually let us blacklist individual players for having large $\sigma\text{-corr}(i)$ scores, or pairs of players for having large $-\text{corr}(i,j)$ scores. In the latter case, we are blacklisting good and bad players at the same rate, which is fine so long as good players retain their slim majority ($f+1$ vs. f initially) among any set of $n-f = 2f+1$ participating players.

A Scheduling Attack. There is a serious flaw in the reasoning above! Recall that $\sigma(t) \in \{-1, 0, 1\}$, where $\sigma(t) = 0$ means that in Bracha-Agreement, the population $A_{\text{keep}} = \emptyset$ committed to keeping the true majority value v^* is empty. The value v^* is determined by the scheduling of messages in Line 2. Whether v^* becomes *known* to any particular player in Line 3 is generally at the discretion of the adversarial scheduler. Thus, in general, the adversary can control whether $A_{\text{keep}} = \emptyset$, and hence whether $\sigma(t) = -v^*$ or 0. Moreover, because the protocol is asynchronous, it can even do so *after* BB_t is populated with coin flips.[10]

These observations give rise to the following attack. The adversary targets two good players i_0, i_1. When Coin-Flip is initiated, the adversary has two choices for $\sigma(t) \in \{-v^*, 0\}$ and can decide which way to set $\sigma(t)$ late in the game. If $\text{sgn}(X_{i_0}(t)) = \text{sgn}(X_{i_1}(t)) = -v^*$, it sets $\sigma(t) = -v^*$; otherwise, it sets $\sigma(t) = 0$. In general, it makes sure $\Sigma_B(t) + \Sigma_G(t) \in [-f, f]$ so it can force roughly equal numbers of players to have Coin-Flip return -1 and 1. Players i_0, i_1 will show unusually large σ-correlation and be blacklisted, and any other blacklisting (from negative correlations) will apply equally to good and bad players. At this point, the *corrupt* players have now attained a slim majority, and are entirely content to let further blacklisting hurt good and bad players equally.

The problem here is that $\sigma(t)$ and $X_{i_0}(t)$ are *not* independent. In reality, $\sigma(t)$ can be chosen maliciously *after* $X_{i_0}(t)$ is known.

A Finger on the Scale. The issue with the previous scheme is that the notions of $\sigma(t)$, the population A_{keep}, and even v^* are too indeterminate. On the other hand, if any good player p finds itself with knowledge of v^* and has set $x_p \in [1, f]$, there is nothing indeterminate *from p's perspective* about the fact that $\{p\} \subseteq A_{\text{keep}} \neq \emptyset$ or that $\sigma(t) = -v^*$. This leads to a natural question: why should p participate in the Coin-Flip protocol *as if it were ignorant of the desired outcome* v^*? Why not "put a finger on the scale" and just write v^* to every entry in column $\text{BB}_t(\cdot, p)$? (We would naturally refrain from judging such special columns according to statistical tests, for example, deviations and correlations.)

The problem with this simple-minded scheme is that if $|A_{\text{keep}}|$ is small, the adversary has the discretion to suppress or allow $p \in A_{\text{keep}}$ to write its column, or any prefix thereof. This allows for a mirror-mimic type attack, in which the sum of BB_t always lies in $[-f, f]$, and yet there are no negative correlations in aggregate between good players flipping fair coins and bad players.

3.4 A Key Observation

To simplify the description of the coin-flipping problem, in Section 3.2, we originally stated the adversary chooses $\sigma(t) = -v^*$, and *wins* the coin-flipping game if it convinces *one* player

[9]When $\text{sgn}(\Sigma_G(t)) = -\sigma(t)$, the adversary is *forced* to set $\sigma(t)\Sigma_B(t) \geq -\sigma(t)\Sigma_G(t)$, increasing the aggregate σ-correlation of bad players and increasing the aggregate negative good-bad correlation. When $\text{sgn}(\Sigma_G(t)) = \sigma(t)$, the adversary can choose to reverse either of these trends and exacerbate the other, that is, reduce good-bad negative correlations but increase bad σ-correlations, or reduce bad σ-correlations but increase good-bad negative correlations.

[10]One player p will be held back at Line 3 of Bracha-Agreement, which can be made to set $v_p = v^*$ or $v_p = 0$, which will be preserved in Line 4. In this way, the adversarial *scheduler* may decide if $A_{\text{keep}} = \{p\}$ or $\emptyset$ after observing BB_t.

to believe the output of Coin-Flip is $\sigma(t)$. It turns out that this simplification gives the adversary too much flexibility, and does not fully capture how coin-flipping is used in Bracha's algorithm.

Consider the sizes of the sets A_{keep} and A_{flip}. There are two relevant cases to consider:

Case $|A_{\text{keep}}| \in [0, f]$**.** When $A_{\text{keep}} \neq \emptyset$, $\sigma(t) = -v^*$ is defined, and it would be bad for the adversary if everyone agreed the output of Coin-Flip were v^*, that is, $\text{sgn}(\Sigma^{(p)}) = v^*$ for all $p \in [n]$. In fact, it would be equally bad for the adversary if $\text{sgn}(\Sigma^{(p)}) = -v^*$ for all p. If so, then a supermajority of $|A_{\text{flip}}| \geq n - f \geq (n + f + 1)/2$ go into the next iteration of Bracha-Agreement holding $-v^*$, and by Lemma 3, all good players will **decide** $-v^*$. To summarize, when $|A_{\text{keep}}| \leq f$, it is *critical* for the adversary to create *disagreement* on the outcome of the global coin flip.[11]

Case $|A_{\text{keep}}| \geq f + 1$**.** If this is the case, then some kind of "finger on the scale" strategy should force the outcome of the coin flip to be v^*. Any player that validates the state of $n - f$ players must necessarily validate the state of some $p \in A_{\text{keep}}$, and hence learn the value of v^*. If any player that knows v^* writes only v^* to its entries in the blackboard, this will surely be the outcome of the global coin flip.

In light of this dichotomy on the size of A_{keep}, we design a coin-flipping protocol in Section 4.2 that (i) forces all players to see the same outcome v^* whenever $|A_{\text{keep}}| \geq f + 1$ — thereby letting Bracha-Agreement terminate — or (ii) reverts to a more standard collective coin-flipping game in which the adversary is obligated to land the sum in the interval $[-f, f]$. Because of the certainty of the outcome in case (i) and the specific strategy forced upon the adversary in case (ii), fraud can now be detected by tracking just one statistic: the correlation scores $\text{corr}(i, j)$ between all pairs of players. Our Coin-Flip procedure makes the notion of the *adversarial direction* $\sigma(t)$ irrelevant. This is in sharp contrast to the protocols of [33] and [28], whose suboptimal resiliency could be partially attributed to working around the idea of the adversarial direction.

4 AN AGREEMENT PROTOCOL WITH OPTIMAL RESILIENCE

4.1 Overview

Our Byzantine-Agreement algorithm (Algorithm 3) consists of $K_{\max} = O(f)$ *epochs*, each of which executes $T \gg n^2$ iterations of Bracha-Agreement, with Line 10 implemented by a new collective coin-flipping protocol Coin-Flip (Algorithm 4, Section 4.2). The tth call to Coin-Flip in epoch k constructs two blackboards BB_{o+2t-1} and BB_{o+2t}, where $o = 2(k-1)T$ is the number of blackboards used in epochs $1, \ldots, k-1$. The odd- and even-numbered blackboards have $\tilde{O}(\sqrt{m})$ and m rows, respectively, for an $m \gg n$ to be determined. In each epoch k there is a *weight vector* $(w_{i,k})_{i \in [n]}$ that influences all calls to Coin-Flip in that epoch. Player i's coin flips are weighted by $w_{i,k} \in [0, 1]$. At the end of each epoch, the procedure Weight-Update generates a new reduced weight vector $(w_{i,k+1})_{i \in [n]}$ based on $(w_{i,k})$ and the history of epoch k. Weights are stable during an epoch and non-increasing with time. The process of reducing players' weights can be thought of as *fractional blacklisting*.[12]

We write the number of players as $n = (3 + \epsilon)f$, where we assume without loss of generality that $\epsilon \in [1/f, 1/2]$. The parameters T, m depend on n, f, ϵ. All the important notation used by the algorithms and their analyses is summarized in Table 2.

[11] Recall from Remark 1 that w.l.o.g., all players follow protocol, but up to f have corrupted random number generators. The analysis of the case $|A_{\text{keep}}| \in [0, f]$ does *not* depend on how the players with corrupted random number generators are distributed among $A_{\text{keep}} \cup A_{\text{flip}}$.

[12] *Fractional* blacklisting was introduced in our first extended abstract [28].

Table 2. Summary of Notation

Notation	Definition and Commentary
n, f, ϵ	The number of players is n, indexed by $[n]$. The number of Byzantine players is $f = n/(3+\epsilon)$.
v^*	The true majority value (if any) at Line 3 of Bracha-Agreement.
$A_{\text{dec}}, A_{\text{keep}}, A_{\text{flip}}$	The populations with $x_p \geq f+1$ (A_{dec}), $x_p \in [1, f]$ (A_{keep}), and $x_p = 0$ (A_{flip}) at the end of an iteration of Bracha-Agreement.
c	A parameter. All *with high probability* bounds hold with probability $1 - n^{-\Omega(c)}$.
$K_{\max}$	The number of epochs is $K_{\max} = 3f + 1$.
T	The number of iterations per epoch is $T = \Theta(n^2 \ln^3 n/\epsilon^4)$.
m	The number of rows in the even blackboards BB_{o+2t} is $m = \Theta(n \ln n/\epsilon^4)$.
m_0	The number of rows in the odd blackboards BB_{o+2t-1} is $m_0 = \sqrt{m \cdot c \ln n}$.
$\mathsf{BB}^{(p,t')}$	$\mathsf{BB}^{(p,t')} = (\mathsf{BB}_1^{(p,t')}, \ldots, \mathsf{BB}_{t'}^{(p,t')})$ is p's historical view of the true iterated blackboard $(\mathsf{BB}_1, \ldots, \mathsf{BB}_{t'})$.
$\text{bias}(t), \Sigma(t)$	In epoch k, the sum of non-$\perp$ entries in BB_{o+2t-1} and BB_{o+2t} are $\text{bias}(t)$ and $\Sigma(t)$, respectively, where $o = 2(k-1)T$ is the number of blackboards used in epochs $1, \ldots, k-1$.
$X_i(t), X_{\max}$	In epoch k, $X_i(t)$ is the sum of all non-$\perp$ entries in $\mathsf{BB}_{o+2t}(\cdot, i)$; it is enforced that $\lvert X_i(t)\rvert \leq X_{\max} = m_0 = \sqrt{m \cdot c \ln n}$.
$X_i^{(p)}(t), \text{bias}^{(p)}(t), \Sigma^{(p)}(t)$	p's view of these values. In epoch k, these are computed from $\mathsf{BB}^{(p,o+2t)}$.
G, B	G and B are the sets of good and bad (Byzantine) players, with respect to some moment in time.
$\Sigma_G(t), \Sigma_B(t)$	In epoch k, the sum of values in BB_{o+2t} written by good players (unbiased coin flips) and bad players, respectively.
$(w_{i,k})_{i\in[n]}$, $(w_i)_{i\in[n]}$	$w_{i,k}$ is the weight of player i during epoch k; w_i is short for the current weight of player i.
$(w_{i,k+1}^{(p)})_{i\in[n]}$	The weight vector computed by p at the end of epoch k from information in $\mathsf{BB}^{(p,2kT)}$. The consensus weight vector $(w_{i,k+1})_{i\in[n]}$ is computed by setting $w_{i,k+1} = w_{i,k+1}^{(i)}$ if $w_{i,k+1}^{(i)} > w_{\min} \stackrel{\text{def}}{=} \sqrt{n}/T$ and 0 otherwise.
$\text{corr}(i,j), \text{corr}^{(p)}(i,j)$	The weighted correlation scores accumulated in one epoch. Here $\text{corr}(i,j) = \langle w_i X_i, w_j X_j \rangle = w_i w_j \sum_t X_i(t) X_j(t)$ and $\text{corr}^{(p)}(i,j)$ is p's view of $\text{corr}(i,j)$, using $X_i^{(p)}, X_j^{(p)}$.

ALGORITHM 3: Byzantine-Agreement()

loop
  Initialize the weight vector $(w_{i,1})_{i\in[n]} = (1, 1, \ldots, 1)$.
  Initialize an empty iterated blackboard BB.
  for k from 1 to $K_{\max} = 3f + 1$ **do** ▷ Epoch index
    for t from 1 to T **do** ▷ Iteration index
      Execute one iteration of Bracha-Agreement (Algorithm 2), using Coin-Flip (Algorithm 4)
      in Line 10. Coin-Flip builds blackboards BB_{o+2t-1} and BB_{o+2t}, where $o = 2(k-1)T$.
    Generate a reduced weight vector $(w_{k+1,i})_{i\in[n]}$ using Weight-Update (Algorithm 6).
  If Bracha-Agreement has failed to reach agreement after $K_{\max} + 1$ epochs, restart **loop** with the initial weight vector and empty iterated blackboard.

We guarantee that Invariant 1 is maintained, with high probability. Let G and B be the good and bad (Byzantine) players at some moment, and let w_i refer to the current weight of player i, that is, $w_i = w_{i,k}$ during epoch k.

Invariant 1. *At all times,*

$$\sum_{i\in G}(1 - w_i) \le \sum_{i\in B}(1 - w_i) + \epsilon^4 f.$$

In other words, the total weight reduction of good players is upper bounded by the total weight reduction of bad players, up to a small $\epsilon^4 f$ error. We prove that with high probability, Byzantine-Agreement reaches agreement after at most $K_{\max}$ epochs. There is a tiny probability that Byzantine-Agreement fails to reach agreement after $K_{\max}$ epochs. If so, the algorithm restarts at epoch 1 with the initial weight vector $(w_{i,1}) = (1, 1, \ldots, 1)$. This guarantees that the algorithm reaches agreement with probability 1 after a finite number of steps. (King and Saia [33, 34] also used this restarting mechanism.)

Theorem 7. *Suppose $n = (3 + \epsilon)f$ where $\epsilon > 0$.* Byzantine-Agreement *(Algorithm 3) achieves agreement with probability 1. With high probability the latency is $O(fmT) = \tilde{O}(n^4/\epsilon^8)$, where $m = \Theta(n \ln n/\epsilon^4)$, $T = \Theta(n^2 \ln^3 n/\epsilon^4)$. The local computation of each player is polynomial in n.*

Proof. It will be proved in Lemma 19 that if the players have failed to reach agreement after $K_{\max} - 1 = 3f$ epochs, all bad players' weights are zero with high probability. However, by Invariant 1 the good players still have weight at least

$$(n - f) - f - \epsilon^4 f = (1 + \epsilon)f - \epsilon^4 f > f + 1 - o(1).$$

Under these circumstances, Lemma 20 states that the good players reach agreement in the next epoch with high probability. If, by chance, the players have not reached agreement after epoch $K_{\max}$, they restart the algorithm at Line 9 of Byzantine-Agreement, returning the epoch counter to 1 and the weight vector to $(1, 1, \ldots, 1)$. Thus, the algorithm terminates with probability 1.

By Theorem 6, the latency to construct $\mathrm{BB}_{t'}$ is $O(m(t')) = O(m)$, so if the algorithm is not restarted in Line 9, the total latency is $O(K_{\max}mT) = O(fmT) = \tilde{O}(n^4/\epsilon^8)$. Even considering the possibility of restarts, this latency bound still holds with high probability and in expectation. □

Organization of Section 4. In Section 4.2, we explain how Coin-Flip is implemented using the Iterated-Blackboard protocol specified in Theorem 6. In Section 4.3, we prove that if the adversary persistently manipulates the outcome of the Coin-Flip protocol, then there will be a detectable

ALGORITHM 4: Coin-Flip() *from the perspective of player p*

Require: $v_p \in \{v^*, \perp\}$, $v^* \in \{-1, 1\}$. This is the tth call to Coin-Flip in epoch k of Byzantine-Agreement. Let $o = 2(k-1)T$ be the number of blackboards used in epochs $1, \ldots, k-1$.

1: ***Stage 1:***
2: Reliable-Broadcast v_p and wait for $n - f$ messages to be validated from some set S_p of players.
3: $\text{val}_p \leftarrow \begin{cases} v^* & \text{if } v_q = v^* \text{ for some } q \in S_p, \\ 0 & \text{if } v_q = \perp \text{ for all } q \in S_p. \end{cases}$
4: Construct BB_{o+2t-1}, writing val_p to every cell in column p.
5: ***Stage 2:***
6: Construct BB_{o+2t}, writing independent coin flips in $\{-1, 1\}$ to cells in column p.
7: $\text{bias}^{(p)} \leftarrow \sum_{j,q} \text{BB}_{o+2t-1}^{(p,o+2t)}(j, q)$ ▷ Substitute $\perp = 0$
8: $\Sigma^{(p)} \leftarrow \sum_{j,q} w_q \cdot \text{BB}_{o+2t}^{(p,o+2t)}(j, q)$ ▷ Substitute $\perp = 0$
9: **return** $(\text{sgn}(\text{bias}^{(p)} + \Sigma^{(p)}))$

negative correlation between some bad player and some good player. In Section 4.4, we give the procedure for reducing weights between epochs, and prove that it maintains Invariant 1 with high probability. The weight reduction algorithm is based on computing a *fractional* maximal matching in a certain graph representing pairwise correlations. In Section 4.5, we prove that agreement is reached after $O(f)$ epochs, with high probability. The analysis of this section is self-contained, except for the proofs of Theorems 6 and 17, which appear in Sections 5 and 6, respectively.

4.2 Implementation of Coin-Flip

Consider epoch k, iteration t of Byzantine-Agreement. It executes one iteration of Bracha-Agreement, which in turn executes Coin-Flip (Algorithm 4) in Line 10 of Bracha-Agreement. When each player p begins executing Coin-Flip, it has a value $v_p \in \{-1, 1, \perp\}$, where $v_p \in \{-1, 1\}$ indicates that $v_p = v^*$ is *the* majority value at Line 3 of Bracha-Agreement, and $v_p = \perp$ indicates that p did not learn the majority value and will adopt the output of Coin-Flip as its value going into the next iteration of Bracha-Agreement.

Recall that x_p is the number of $v^* \in \{-1, 1\}$ messages validated by p in Line 4 of Bracha-Agreement and A_{keep} is the set of all p such that $x_p \in [1, f]$ before executing Coin-Flip. The first stage of Coin-Flip is to populate the next blackboard BB_{o+2t-1} that will help end the game quickly if $|A_{\text{keep}}| \geq f + 1$ and cause no harm if $|A_{\text{keep}}| \in [0, f]$. Here, the offset $o = 2(k-1)T$ is the number of blackboards used in epochs $1, \ldots, k-1$. The contents of BB_{o+2t-1} are used to generate a *bias*. The second stage of Coin-Flip populates a blackboard BB_{o+2t} with random values in $\{-1, 1\}$. Let

$$\Sigma = \sum_{j,q} w_q \cdot \text{BB}_{o+2t}(j, q),$$

be the *weighted* sum of the contents of BB_{o+2t}, mapping $\perp$ to 0. The output of Coin-Flip is $\text{sgn}(\text{bias} + \Sigma)$. Due to the scheduling power of the adversary, each player has a slightly different view of these two blackboards. Naturally p outputs $\text{sgn}(\text{bias}^{(p)} + \Sigma^{(p)})$, where a superscript of (p) in any variable indicates p's opinion of its value.

The even and odd blackboards BB_{o+2t} and BB_{o+2t-1} have the following number of rows

$$m(o + 2t) = m,$$
$$m(o + 2t - 1) = m_0 = \sqrt{m \cdot c \ln n}.$$

Here, c is the parameter that controls the error probability. Henceforth, *with high probability* means an event holds with probability $1 - n^{-\Omega(c)}$.

Lemma 8. *If* $|A_{\text{keep}}| \geq f+1$ *then* $v^* \cdot \text{bias} > (n-f)m_0 > \frac{2}{3}n\sqrt{m \cdot c \ln n}$. *If* $|A_{\text{keep}}| = 0$ *then* bias $= 0$. *If* $|A_{\text{keep}}| \in [1, f]$ *then* $v^* \cdot \text{bias} \in [0, nm_0]$, *and can be selected by the adversary.*

Proof. If $|A_{\text{keep}}| \geq f+1$ then for every player p, there exists a $q \in S_p$ with $q \in A_{\text{keep}}$ and $v_q = v^*$, hence $\text{val}_p = v^*$. By Theorem 6, the number of values written to BB_{o+2t-1} is at least $(n-f)m_0$ and hence $v^* \cdot \text{bias} \geq (n-f)m_0 > \frac{2}{3}n\sqrt{m \cdot c \ln n}$.

If $|A_{\text{keep}}| = 0$ then every player will set $\text{val}_p = 0$ hence bias $= 0$. □

In the second stage of Coin-Flip, the players will populate BB_{o+2t} with coin flips in $\{-1, 1\}$. Define $X_i(t)$ as the sum of all non-$\perp$ entries in column $\text{BB}_{o+2t}(\cdot, i)$. By Chernoff bounds, if i is uncorrupted then $|X_i(t)| \leq X_{\max}$ with high probability, where

$$X_{\max} = m_0 = \sqrt{m \cdot c \ln n}.$$

We will force $|X_i(t)| \leq X_{\max}$ to hold with probability 1 by rounding $X_i(t)$ to $\pm X_{\max}$ if it lies outside $[-X_{\max}, X_{\max}]$.

Lemma 9. *If* $|A_{\text{keep}}| \geq f+1$, *the output of Coin-Flip will be* v^* *for all players, with high probability.*

Proof. By Lemma 8, $|\,\text{bias}\,| \geq (n-f)m_0 > \frac{2}{3}n\sqrt{m \cdot c \ln n}$. The number of good coin flips in BB_{o+2t} is between $m(n-2f)$ and mn, which we model as a martingale with an optional stopping time controlled by the adversary. By Azuma's inequality,[13] the sum of all $\Theta(mn)$ good coin flips is $\sqrt{mn \cdot c \ln n}$ in absolute value, with high probability. Due to the $X_{\max}$ ceiling, the contribution of corrupt players to the sum is at most $fX_{\max} < \frac{1}{3}n\sqrt{m \cdot c \ln n}$ in absolute value. Since $\frac{1}{3}n\sqrt{m \cdot c \ln n} + \tilde{O}(\sqrt{mn}) + f < \frac{2}{3}n\sqrt{m \cdot c \ln n}$, the contribution of corrupt and non-corrupt players will be much smaller than bias, with high probability, and $\text{sgn}(\text{bias}^{(p)} + \Sigma^{(p)}) = v^*$ for all p. □

Lemma 10. *If* $|A_{\text{keep}}| \leq f$, *and for some player* p, $\text{bias}^{(p)} + \Sigma^{(p)} \notin [-f, f]$, *all good players will* ***decide*** *the value* $\text{sgn}(\text{bias}^{(p)} + \Sigma^{(p)})$ *by the next iteration of Bracha-Agreement.*

Proof. By Theorem 6, two players p and q disagree in at most f locations in BB_{o+2t-1} and BB_{o+2t}. Since the absolute value of any cell in either matrix is at most 1, if $\text{bias}^{(p)} + \Sigma^{(p)} \notin [-f, f]$ then for any p, q, $\text{sgn}(\text{bias}^{(p)} + \Sigma^{(p)}) = \text{sgn}(\text{bias}^{(q)} + \Sigma^{(q)})$. Thus, at the end of this iteration of Bracha-Agreement, a supermajority of at least $|A_{\text{flip}}| \geq n - f \geq (n+f+1)/2$ players will hold the same value and, by Lemma 3, will reach agreement in the next iteration of Bracha-Agreement. Observe that this lemma does not care how the players with corrupted random number generators are distributed among the population $A_{\text{keep}} \cup A_{\text{flip}}$. See Remark 1. □

To summarize, Lemmas 9 and 10 imply that if the adversary has an interest in *prolonging* the moment of agreement, it must

— Force $|A_{\text{keep}}|$ to be at most f in *every* iteration of Bracha-Agreement.

— Force $\text{bias}^{(p)} + \Sigma^{(p)}$ to lie within the interval $[-f, f]$ in *every* invocation of Coin-Flip.

When reasoning about the protocol, we will therefore assume that the adversary picks some strategy that satisfies these two requirements, whenever possible.

[13] Azuma's inequality [18] states that if $Y_0, Y_1, Y_2, \ldots, Y_n$ is a (super)martingale and each $|Y_k - Y_{k-1}| \leq c_k$, then $\Pr(Y_n - Y_0 \geq \rho) \leq \exp(-\rho^2/(2\sum_{k=1}^n c_k^2))$. If $Y_0, Y_1, \ldots, Y_n$ is a (sub)martingale with each $|Y_k - Y_{k-1}| \leq c_k$, then $\Pr(Y_n - Y_0 \leq -\rho) \leq \exp(-\rho^2/(2\sum_{k=1}^n c_k^2))$.

4.3 Negative Correlations

Recall that k is the current epoch, $t \in [T]$ is the iteration index in epoch k, and $(w_i) = (w_{i,k})$ is the current weight vector used in epoch k. In each epoch, we track *weighted* pairwise correlations, defined as:

$$\operatorname{corr}(i,j) = \langle w_i X_i, w_j X_j \rangle = w_i w_j \sum_{t \in [T]} X_i(t) X_j(t).$$

Let G and B be the sets of good and bad players.

Lemma 11. *Suppose the weight vector $(w_i)_{i\in[n]}$ used in an epoch satisfies Invariant 1, but Bracha-Agreement fails to reach agreement within the epoch. Let $m = \Omega(n \ln n/\epsilon^4)$ and $T = \Omega(n^2 \ln^3 n/\epsilon^4)$. Then, with high probability,*

(1) Every pair of distinct $i, j \in G$ has $-\operatorname{corr}(i,j) \le w_i w_j \beta$, where $\beta = m\sqrt{T(c \ln n)^3}$.
(2) If the adversary does not corrupt any new players during the epoch, then

$$\sum_{(i,j)\in G\times B} \max\{0, -\operatorname{corr}(i,j) - w_i w_j \beta\} \ge \frac{1}{8}\epsilon^2 f m T.$$

Proof of Part 1. Fix an iteration $t \in [T]$. If $i \in G$, let $\delta^i_{t,r} \in \{-1, 0, 1\}$ be value player i writes to $\mathrm{BB}_{o+2t}(r, i)$, or 0 if it never makes such a write. For any $r, s \ge 1$ and $i, j \in G$, $\mathbf{E}[\delta^i_{t,r}\delta^j_{t,s}] = 0$ since each of $\delta^i_{t,r}, \delta^j_{t,s}$ is either 0 or a fair coin flip independent of the other. By linearity of expectation this implies $\mathbf{E}[X_i(t)X_j(t)] = 0$ as well.

Consider the martingale $(S_t)_{t\in[0,T]}$, where

$$S_0 = 0$$
$$\text{and } S_t = S_{t-1} + X_i(t)X_j(t).$$

For any t, $|S_t - S_{t-1}| \le X^2_{\max}$. By Azuma's inequality, $|S_T| \le X^2_{\max}\sqrt{T \cdot c \ln n} = m\sqrt{T(c \ln n)^3}$ with high probability. Therefore, by a union bound, for every pair of distinct $i, j \in G$, $-\operatorname{corr}(i,j) \le w_i w_j m\sqrt{T(c \ln n)^3} = w_i w_j \beta$ with high probability. □

Part 2 of Lemma 11 will be proved following Lemmas 12–16. It only applies to epochs in which the adversary corrupts no one, so we shall assume that G, B are stable throughout the epoch.

In the Coin-Flip algorithm, the construction of BB_{o+2t-1} *logically* precedes the construction of BB_{o+2t}, but because of asynchrony some of the contents of BB_{o+2t-1} may actually depend on the coin flips written to BB_{o+2t}.[14] We eliminate these mild dependencies as follows. Suppose that $\hat{p}$ is the *first* player in G to fix its historical view $\mathrm{BB}^{(\hat{p},o+2t-1)}$. At this moment, define $\overline{\mathrm{bias}}(t)$ as

$$\overline{\mathrm{bias}}(t) = \sum_{j,q} \mathrm{BB}^{(\hat{p},o+2t-1)}_{o+2t-1}(j,q) \qquad \text{(Treating } \perp \text{ as 0)}$$

Write $\Sigma(t) = \Sigma_G(t) + \Sigma_B(t)$, where $\Sigma_G(t)$ and $\Sigma_B(t)$ are the sum of weighted coin flips in BB_{o+2t} originating from good and bad players, respectively.

Lemma 12. *In any iteration t,*

(1) For any player q, $\left|\overline{\mathrm{bias}}(t) - \mathrm{bias}^{(q)}(t)\right| \le f$.
(2) $\mathbf{E}\left[\overline{\mathrm{bias}}(t)\Sigma_G(t)\right] = 0$.

[14] The construction of BB_{o+2t} can proceed as soon as $n - f$ players are finished with BB_{o+2t-1}. Thus, a group of f slow and corrupt players can choose whether to perform their last write in BB_{o+2t-1} based on the contents of BB_{o+2t}.

(3) If Bracha-Agreement does not terminate by iteration $t+1$, then

$$-\Sigma_G(t)\Sigma_B(t) \geq \Sigma_G(t)^2 + \overline{\text{bias}}(t)\Sigma_G(t) - 2f|\Sigma_G(t)|.$$

Proof. *Part 1.* By Theorem 6, $\text{BB}_{o+2t-1}^{(\hat{p},o+2t-1)}$ and $\text{BB}_{o+2t-1}^{(q,t')}$ disagree in at most f cells, for any $q \in [n]$ and $t' \geq o+2t-1$, hence $|\overline{\text{bias}}(t) - \text{bias}^{(q)}(t)| \leq f$.

Part 2. By definition, $\overline{\text{bias}}(t)$ is fixed before any good players have written anything to BB_{o+2t}. Thus $\mathbf{E}[\overline{\text{bias}}(t)\Sigma_G(t)] = \overline{\text{bias}}(t) \cdot \mathbf{E}[\Sigma_G(t)] = 0$.

Part 3. By Lemmas 9 and 10, if the adversary avoids termination by iteration $t+1$, then $\text{sgn}(\text{bias}^{(p)}(t)+\Sigma^{(p)}(t)) \neq \text{sgn}(\text{bias}^{(q)}(t)+\Sigma^{(q)}(t))$ for two players p, q. Since $|\overline{\text{bias}}(t)-\text{bias}^{(p)}(t)| \leq f$ and $|\Sigma(t) - \Sigma^{(p)}(t)| \leq f$, it follows from $\Sigma(t) = \Sigma_G(t) + \Sigma_B(t)$ that

$$-2f \leq \overline{\text{bias}}(t) + \Sigma_G(t) + \Sigma_B(t) \leq 2f.$$

Rearranging terms, we have both

$$-\Sigma_B(t) \geq -2f + \overline{\text{bias}}(t) + \Sigma_G(t)$$

and

$$\Sigma_B(t) \geq -2f - \overline{\text{bias}}(t) - \Sigma_G(t).$$

Depending on $\text{sgn}(\Sigma_G(t))$, we multiply the first inequality by $\Sigma_G(t) \geq 0$ or the second by $-\Sigma_G(t) \geq 0$, which implies the following.

$$-\Sigma_G(t)\Sigma_B(t) \geq \Sigma_G(t)^2 + \overline{\text{bias}}(t)\Sigma_G(t) - 2f|\Sigma_G(t)|. \tag{1}$$

□

Lemmas 13–16 analyze the terms of (1). Note that since $\mathbf{Var}(|\Sigma_G(t)|) = \mathbf{E}[\Sigma_G(t)^2] - \mathbf{E}[|\Sigma_G(t)|]^2 \geq 0$, it follows that $\mathbf{E}[|\Sigma_G(t)|] \leq \sqrt{\mathbf{E}[\Sigma_G(t)^2]}$. Thus, a bound on the first term of (1) will imply a bound on the third.

Lemma 13. *If (w_i) satisfy Invariant 1, then for any $\hat{G} \subseteq G$ with $|\hat{G}| = n-2f = (1+\epsilon)f$, $\sum_{i\in\hat{G}} w_i^2 \geq \frac{1}{2}\epsilon^2 f$.*

Proof. We compute:

$$\sum_{i\in\hat{G}} w_i = |\hat{G}| - \sum_{i\in\hat{G}}(1-w_i) \geq |\hat{G}| - \sum_{i\in B}(1-w_i) - \epsilon^4 f \geq |\hat{G}| - \left(1+\epsilon^4\right) f = \left(\epsilon - \epsilon^4\right) f \geq \frac{7\epsilon}{8} f.$$

The first inequality follows from Invariant 1 and the fact that the total weight deduction of $\hat{G}$ is at most that of G. The second inequality follows from $w_i \in [0,1]$, so the total weight deduction of B is at most f. The equality follows from $|\hat{G}| = n - 2f = (1+\epsilon)f$. Finally, the last inequality follows from the assumption that $\epsilon \leq 1/2$. Consequently:

$$\sum_{i\in\hat{G}} w_i^2 = |\hat{G}| \sum_{i\in\hat{G}} w_i^2 \frac{1}{|\hat{G}|} \geq |\hat{G}| \left(\sum_{i\in\hat{G}} w_i \frac{1}{|\hat{G}|}\right)^2 \geq |\hat{G}| \left(\frac{7\epsilon}{8(1+\epsilon)}\right)^2 \geq \frac{1}{2}\epsilon^2 f\,,$$

where the first inequality is Jensen's inequality, the middle inequality is from above, and the last inequality follows from $|\hat{G}| = (1+\epsilon)f$ and the assumption $\epsilon \leq \frac{1}{2}$. □

Lemma 14. *No matter how the coin flips of G are scheduled in iteration t, $\mathbf{E}[\Sigma_G(t)^2] \geq \frac{1}{2}\epsilon^2 mf$.*

Proof. The good players write between $m(n-2f)$ and mn coin flips to BB_{o+2t}, at the adversary's discretion. For $r \in [0, 2mf]$, let S_r be the sum of the first $m(n-2f)+r$ coin flips generated by the good players. Then $\mathbf{E}[\Sigma_G(t)^2] = \mathbf{E}[S^2_{2fm}]$, which we claim is at least $\mathbf{E}[S_0]$. In general $S_r = S_{r-1} + w_i\delta_r$, where $\delta_r \in \{-1, 1\}$ if the adversary lets player i flip the next coin and $\delta_r = 0$ if the adversary chooses to stop allowing coin flips. If $\delta_r = 0$ then $S_r = S_{r-1}$ and if $\delta_r \in \{-1, 1\}$ then

$$S_r^2 = \begin{cases} (S_{r-1} + w_i)^2 = S_{r-1}^2 + 2w_iS_{r-1} + w_i^2 & \text{with probability } \frac{1}{2}, \\ (S_{r-1} - w_i)^2 = S_{r-1}^2 - 2w_iS_{r-1} + w_i^2 & \text{with probability } \frac{1}{2}. \end{cases}$$

Thus, $\mathbf{E}[S_r^2 \mid \delta_r \neq 0] = S_{r-1}^2 + w_i^2 > S_{r-1}^2$, and in general, $\mathbf{E}[S_r^2] \geq \mathbf{E}[S_{r-1}^2] \geq \cdots \geq \mathbf{E}[S_0^2]$. Thus, the adversarial strategy minimizing $\Sigma_G(t)^2$ is to allow as few coin flips as possible, and from those $n-2f$ players $\hat{G}$ with the smallest weights. By Lemma 13 we have

$$\mathbf{E}[S_0^2] \geq m \sum_{i \in \hat{G}} w_i^2 \geq \frac{1}{2}\epsilon^2 mf. \qquad \square$$

Lemma 15. *With high probability, in any epoch k, $\sum_{t\in[T]} \Sigma_G(t)^2 \geq \frac{1}{2}\epsilon^2 mfT - mn\sqrt{T(c\ln n)^3}$.*

Proof. Consider the sequence $(A_t)_{t\in[T]}$, where

$$A_0 = 0,$$
$$A_t = A_{t-1} + \Sigma_G(t)^2 - \frac{1}{2}\epsilon^2 mf.$$

By Lemma 14, $\mathbf{E}[\Sigma_G(t)^2] \geq \frac{1}{2}\epsilon^2 mf$, so $(A_t)_t$ is a submartingale. Since $\Sigma_G(t)$ is a sum of at most mn coin flips, by a Chernoff bound, $|A_t - A_{t-1}| \leq \Sigma_G(t)^2 \leq mn \cdot c\ln n$ with high probability. By Azuma's inequality, with high probability, $A_T \geq -(mn \cdot c\ln n)\sqrt{T \cdot c\ln n}$ and

$$\sum_{t\in[T]} \Sigma_G(t)^2 = \frac{1}{2}\epsilon^2 mfT + A_T \geq \frac{1}{2}\epsilon^2 mfT - mn\sqrt{T(c\ln n)^3}. \qquad \square$$

Lemma 16. *With high probability, we have both $\sum_{t\in[T]} \overline{\mathrm{bias}(t)}\Sigma_G(t) \leq mn\sqrt{nT(c\ln n)^3}$ and $\sum_{t\in[T]} |\Sigma_G(t)| \leq T\sqrt{mnc\ln n}$.*

Proof. By Lemma 12, $\mathbf{E}[\overline{\mathrm{bias}}(t)\Sigma_G(t)] = 0$ and hence the sequence $(A_t)_{t\in[T]}$ is a martingale, where

$$A_0 = 0,$$
$$A_t = A_{t-1} + \overline{\mathrm{bias}}(t)\Sigma_G(t).$$

With high probability $|\Sigma_G(t)| \leq \sqrt{mn(c\ln n)}$ and $\overline{\mathrm{bias}}(t) \leq nm_0 = n\sqrt{m(c\ln n)}$, hence by Azuma's inequality, $\sum_{t\in[T]} \overline{\mathrm{bias}}(t)\Sigma_G(t) \leq mn\sqrt{nT(c\ln n)^3}$ with high probability. $\square$

We are now equipped to prove the second part of Lemma 11.

Proof of Part 2 of Lemma 11. Recall G, B are the sets of good and bad players, which, by assumption, do not change during the epoch.

$$\begin{aligned} -\sum_{(i,j)\in G\times B} \mathrm{corr}(i,j) &= \sum_{t\in[T]} -\Sigma_G(t)\Sigma_B(t) \\ &\geq \sum_{t\in[T]} \left(\Sigma_G(t)^2 - 2f|\Sigma_G(t)| + \overline{\mathrm{bias}}(t)\Sigma_G(t)\right) \qquad \text{(Lemma 12)} \end{aligned}$$

ALGORITHM 5: Rising-Tide($H = (V, E, c_V, c_E)$)

$E' \leftarrow \{\{i,j\} \in E \mid c_E(i,j) > 0\}$.
$\mu(i,j) \leftarrow 0$ for all $i, j \in V$.
while $E' \neq \emptyset$ **do**
  Let $\mu_{E'}(i,j) = \begin{cases} 1 & \text{if } \{i,j\} \in E' \\ 0 & \text{otherwise.} \end{cases}$.
  Choose maximum $\epsilon > 0$ such that $\mu' = \mu + \epsilon\mu_{E'}$ is a feasible fractional matching.
  Set $\mu \leftarrow \mu'$.
  $E' \leftarrow E' - \{\{i,j\} \mid i \text{ or } j \text{ or } \{i,j\} \text{ is saturated}\}$ ▷ $\mu(i,j)$ cannot increase
return μ.

$$\geq \frac{1}{2}\epsilon^2 mfT - O(mn\sqrt{T(c\ln n)^3}) - 2fO(T\sqrt{mnc\ln n}) - O(mn\sqrt{nT(c\ln n)^3})$$

(W.h.p., by Lemmas 15 and 16)

$$= \left(\frac{1}{2}\epsilon^2 - O\left(\sqrt{\frac{nc\ln n}{m}}\right)\right) mfT - O(mn\sqrt{nT(c\ln n)^3})$$

$$\geq \left(\frac{1}{2}\epsilon^2 - o(\epsilon^2)\right) mfT - O(mn\sqrt{nT(c\ln n)^3}) \qquad \text{(whenever } m = \Omega(n\ln n/\epsilon^4)\text{)}$$

$$\geq \frac{1}{4}\epsilon^2 fmT. \qquad \text{(whenever } T = \Omega(n\ln^3 n/\epsilon^4)\text{)}$$

Finally, since $\max\{0, -\operatorname{corr}(i,j) - w_i w_j\beta\} \geq -\operatorname{corr}(i,j) - w_i w_j\beta$, Lemma 11(2) follows from the above inequality and the fact that

$$\sum_{(i,j)\in G\times B} w_i w_j\beta \leq |G|\cdot|B|\cdot\beta \leq nf\cdot m\sqrt{T(c\ln n)^3} \leq \frac{1}{8}\epsilon^2 fmT$$

holds whenever $T = \Omega(n^2\ln^3 n/\epsilon^4)$. □

4.4 Blacklisting via Fractional Matching

After the T iterations of epoch k are complete, we reduce the weight vector (w_i) in preparation for epoch $k+1$. According to Lemma 11, if a correlation score $-\operatorname{corr}(i,j)$ is too large, $B \cap \{i,j\} \neq \emptyset$ w.h.p., so reducing *both* of i's and j's weights by the *same* amount preserves Invariant 1. With this end in mind, Weight-Update (Algorithm 6) constructs a complete, vertex- and edge-capacitated graph H on $[n]$, finds a fractional maximal matching μ in H, then docks the weights of i and j by $\mu(i,j)$, for each pair $\{i,j\}$.

Definition 1 (Fractional Maximal Matching). Let $H = (V, E, c_V, c_E)$ be a graph where $c_V : V \to \mathbb{R}_{\geq 0}$ are vertex capacities and $c_E : E \to \mathbb{R}_{\geq 0}$ are edge capacities. A function $\mu : E \to \mathbb{R}_{\geq 0}$ is a *feasible fractional matching* if $\mu(i,j) \leq c_E(i,j)$ and $\sum_j \mu(i,j) \leq c_V(i)$. It is *maximal* if it is not strictly dominated by any feasible μ'.

The Rising-Tide algorithm initializes $\mu = 0$ and simply simulates the continuous process of increasing all $\mu(i,j)$-values in lockstep, so long as i, j, and $\{i,j\}$ are not saturated. At the moment one becomes saturated, $\mu(i,j)$ is frozen at its current value. See Algorithm 5.

Rounding Weights Down. At the end of epoch k, player p generates a local weight vector $(w^{(p)}_{i,k+1})_{i\in[n]}$, which is a function of $(w_{i,k})_{i\in[n]}$ and its historical view $\mathsf{BB}^{(p,2kT)}$ of the first k epochs. (There are $2T$ blackboards in each epoch.) The consensus weight vector $(w_{i,k+1})_{i\in[n]}$ is obtained

ALGORITHM 6: Weight-Update *from the perspective of player p.*

Output: Weights $(w_{i,k})_{i\in[n],k\geq 0}$ where $w_{i,k}$ refers to the weight w_i *after* processing epoch $k-1$, and is used throughout epoch k.

1: Set $w_{i,1} \leftarrow 1$ for all i. ▷ All weights are 1 in epoch 1.

2: **for** epoch $k = 1, 2, \ldots, K_{\max} - 1$ **do** ▷ $K_{\max} - 1$ = last epoch followed by Weight-Update.

3: Execute T iterations of Bracha-Agreement, using weights $(w_{i,k})$ in Coin-Flip. Let $\mathrm{corr}^{(p)}$ be the resulting correlation scores known to p. Construct the excess correlation graph $H_k^{(p)}$ with capacities:

$$c_V(i) = w_{i,k},$$

$$c_E^{(p)}(i,j) = \frac{8}{\epsilon^2 fmT} \cdot \max\left\{0, -\mathrm{corr}^{(p)}(i,j) - w_{i,k}w_{j,k}\beta\right\}.$$

4: $\mu_k^{(p)} \leftarrow$ Rising-Tide$(H_k^{(p)})$ ▷ A maximal fractional matching

5: For each i, set

$$w_{i,k+1}^{(p)} \leftarrow w_{i,k} - \sum_j \mu_k^{(p)}(i,j).$$

6: Once player i's vector $(w_{j,k+1}^{(i)})_{j\in[n]}$ is known, set the value

$$w_{i,k+1} = \begin{cases} w_{i,k+1}^{(i)} & \text{if } w_{i,k+1}^{(i)} > w_{\min} \stackrel{\text{def}}{=} \frac{\sqrt{n}}{T}, \\ 0 & \text{otherwise.} \end{cases}$$

by everyone adopting the weight of i according to player i's local view, and rounding down if it is too close to zero.

$$w_{i,k+1} = \begin{cases} w_{i,k+1}^{(i)} & \text{if } w_{i,k+1}^{(i)} > w_{\min} \stackrel{\text{def}}{=} \frac{\sqrt{n}}{T}, \\ 0 & \text{otherwise.} \end{cases}$$

Recall from Theorem 6(3) that if i writes anything to any blackboard in epoch $k+1$, that every player can deduce what its view $\mathrm{BB}^{(i,2kT)}$ looked like at the end of epoch k, and hence what $w_{i,k+1}^{(i)}$ and $w_{i,k+1}$ are. By ensuring that all participating players use exactly the *same* weight function, we eliminate one source of potential numerical disagreement.

We will see that the maximum pointwise disagreement in the local weight vectors $|w_{i,k+1}^{(p)} - w_{i,k+1}^{(q)}|$ is less than $w_{\min}$. As a consequence, if any p thinks that $w_{i,k+1}^{(p)} = 0$ then all players will agree that $w_{i,k+1} = 0$.

Excess Graph. The *excess correlation graph* $H = (V, E, c_V, c_E)$ used in Algorithm 6 is a complete undirected graph on $V = [n]$, capacitated as follows:

$$c_V(i) = w_{i,k}$$

$$c_E(i,j) = \frac{8}{\epsilon^2 fmT} \cdot \max\{0, -\mathrm{corr}(i,j) - w_{i,k}w_{j,k}\beta\},$$

where β is the quantity from Lemma 11. By Part 1 of Lemma 11, $c_E(i,j) = 0$ whenever both i and j are good.

The Weight-Update algorithm from the perspective of player p is presented in Algorithm 6. We want to ensure that the fractional matchings computed by good players are numerically very close

to each other, and for this reason we cannot use just *any* maximal matching algorithm, for example, the standard "greedy" algorithm will not work. The Rising-Tide maximal matching algorithm (Algorithm 5) has a continuous Lipschitz property, meaning bounded perturbations to its input yield bounded perturbations to its output.

4.4.1 Properties of Rising-Tide. Recall that $c_V(i)$ is initialized to be the old weight $w_{i,k}$ and the new weight in p's local view is set to $w^{(p)}_{i,k+1} = c_V(i) - \sum_j \mu^{(p)}_k(i,j)$. We are mainly interested in differences in the new weight vector computed by players that begin with slightly different graphs $H^{(p)}, H^{(q)}$. Theorem 17 bounds the distance between outputs in terms of the distance between inputs.

Theorem 17. *Let $H^{(p)} = (V, E, c^{(p)}_V, c^{(p)}_E)$ and $H^{(q)} = (V, E, c^{(q)}_V, c^{(q)}_E)$ be two capacitated graphs, which differ by $\eta_E = \sum_{i,j} |c^{(p)}_E(i,j) - c^{(q)}_E(i,j)|$ in their edge capacities and $\eta_V = \sum_i |c^{(p)}_V(i) - c^{(q)}_V(i)|$ in their vertex capacities. Let $\mu^{(p)}$ and $\mu^{(q)}$ be the fractional matchings computed by* Rising-Tide *(Agorithm 5). Then:*

$$\sum_i \left| \left(c^{(p)}_V(i) - \sum_j \mu^{(p)}(i,j) \right) - \left(c^{(q)}_V(i) - \sum_j \mu^{(q)}(i,j) \right) \right| \le \eta_V + 2\eta_E.$$

Observe that $c_V(i) - \sum_j \mu(i,j)$ is the new capacity of vertex i after deducting the maximal matching edges adjacent to i. Thus, the expression in Theorem 17 measures the *total* difference in all new vertex capacities after deducting the matchings computed from the perspectives of p and q.

4.5 Error Accumulation and Reaching Agreement

We perform fractional blacklisting after the first $K_{\max} - 1 = 3f$ epochs. Let $k \in [1, K_{\max} - 1]$ be the index of the current epoch, and let $(w_{i,k})$ be the weights that were used in the execution of Coin-Flip during epoch k. Upon completing epoch k, each player p applies Weight-Update(Algorithm 6) to update the consensus weight vector $(w_{i,k})_{i\in[n]}$ to produce a local weight vector $(w^{(p)}_{i,k+1})_{i\in[n]}$, and then the consensus weight vector $(w_{i,k+1})_{i\in[n]}$ used throughout epoch $k+1$.

Lemma 18 (Maintaining Invariant 1). *Suppose for some $\epsilon > 0$ that $n = (3+\epsilon)f$, $m = \Omega(n \ln n/\epsilon^4)$, and $T = \Omega(n^2 \ln^3 n/\epsilon^4)$. At any point in epoch $k \in [1, K_{\max}]$, with high probability,*

$$\sum_{i\in G}(1 - w_{i,k}) \le \sum_{i\in B}(1 - w_{i,k}) + \frac{\epsilon^4}{\sqrt{n \ln^6 n}} \cdot (k-1).$$

Proof. By induction on k. For the base case $k = 1$ all the weights are 1 so the claim clearly holds. Now suppose the claim holds for k and consider $k+1$. Fix any player p. A consequence of Lemma 11 (Part 1) is that with high probability, player p's view of the weight vector, $(w^{(p)}_{i,k+1})$, is derived from $(w_{i,k})$ by deducting at least as much weight from bad players as from good players. By the inductive hypothesis,

$$\sum_{i\in G}(1 - w^{(p)}_{i,k+1}) \le \sum_{i\in B}(1 - w^{(p)}_{i,k+1}) + \frac{\epsilon^4}{\sqrt{n \ln^6 n}} \cdot (k-1).$$

Subsequently, player p derives the consensus weight vector $(w_{i,k+1})$ from $(w^{(q)}_{i,k+1})_{q\in[n],i\in[n]}$ by setting $w_{q,k+1} = w^{(q)}_{q,k+1}$, rounding the value down to 0 if it is at most $w_{\min}$. Therefore,

$$\sum_{i\in G}(1 - w_{i,k+1}) \le \sum_{i\in B}(1 - w_{i,k+1}) + \frac{\epsilon^4}{\sqrt{n \ln^6 n}} \cdot (k-1) + \sum_{q\in[n]} \left| w^{(p)}_{q,k+1} - w^{(q)}_{q,k+1} \right| + w_{\min} n_0, \qquad (2)$$

where n_0 is the number of players whose weight is rounded down to 0 after epoch k.

Hence, it suffices to show that $\sum_{q\in[n]} |w^{(p)}_{q,k+1} - w^{(q)}_{q,k+1}| + w_{\min} n_0 \le \epsilon^4/\sqrt{n \ln^6 n}$. By Theorem 17, the computed weight difference between player p and any player q can be bounded by twice the sum of all edge capacity differences (η_E), since they completely agree on the vertex capacities ($\eta_V = 0$).[15] According to Algorithm 6, the edge capacities differ due to underlying disagreement on the $\mathrm{corr}(i,j)$ values. Thus,

$$\left|w^{(p)}_{q,k+1} - w^{(q)}_{q,k+1}\right| \le 2\eta_E \le 2 \cdot \frac{8}{\epsilon^2 f m T} \sum_{i\neq j} \left|\mathrm{corr}^{(p)}(i,j) - \mathrm{corr}^{(q)}(i,j)\right|. \tag{3}$$

By Theorem 6, two players p, q may only disagree in up to f cells of the blackboards

$$(\mathrm{BB}_{2(k-1)T+2}, \mathrm{BB}_{2(k-1)T+4}, \ldots, \mathrm{BB}_{2kT}),$$

that is, those used to compute corr-values in epoch k. Since the sum of each column in each blackboard is bounded by $X_{\max}$, $|X^{(p)}_i(t)X^{(p)}_j(t) - X^{(q)}_i(t)X^{(q)}_j(t)| \le 2X_{\max}$. Each of the f cells that p and q disagree on affects $n-1$ corr-values. Therefore, the right hand side of (3) is upper bounded by:

$$\begin{aligned}
&\le 2 \cdot \frac{8}{\epsilon^2 f m T} \cdot nf \cdot 2X_{\max} \\
&\le \frac{32 n X_{\max}}{\epsilon^2 m T} \\
&\le \frac{\sqrt{n}}{T} && (m = \Omega(n \ln n/\epsilon^4) \text{ and } X_{\max} = \Theta(\sqrt{m \ln n})) \\
&= w_{\min}.
\end{aligned}$$

Now the inductive step for $k+1$ holds by noticing that in (2).

$$\begin{aligned}
\sum_{q\in[n]} \left|w^{(p)}_{q,k+1} - w^{(q)}_{q,k+1}\right| + w_{\min} n_0 &\le 2 w_{\min} n = 2n^{3/2}/T \\
&\le \frac{\epsilon^4}{\sqrt{n \ln^6 n}}. && (T = \Omega(n^2 \ln^3 n/\epsilon^4))
\end{aligned}$$

Since $k \le K_{\max} - 1 = 3f$, we conclude that Invariant 1 holds in every epoch, with high probability. That is, if $(w_i), G, B$ are the weight vector, good players, and bad players at any point in time, then

$$\begin{aligned}
\sum_{i\in G}(1-w_i) &\le \sum_{i\in B}(1-w_i) + \frac{\epsilon^4}{\sqrt{n \ln^6 n}} \cdot (K_{\max} - 1) \\
&\le \sum_{i\in B}(1-w_i) + \epsilon^4 f. && (\sqrt{n \ln^6 n} \ge 3)
\end{aligned}$$

□

The next observation and Lemma 19 shows that the weight of every bad player becomes 0 after running $K_{\max} - 1$ epochs of Weight-Updates without reaching agreement.

Observation 1. *For any q and k, if there exists a player p such that $w^{(p)}_{q,k} = 0$, then $w_{q,k} = 0$.*

[15] It is somewhat paradoxical that they can *agree* on the vertex capacities when *agreement* is the whole problem. Consider the moment at the end of epoch k when p is computing its weight vector $(w^{(p)}_{q,k+1})_{q\in[n]}$. If $c_V(q)$ (the weight $w_{q,k}$ computed after epoch $k-1$) is unknown to p, then, by Theorem 6(3), q must not have successfully written anything to any blackboard in epoch k, in which case all edges incident to q have capacity zero. This situation is indistinguishable from one in which all players agree that $c_V(q) = 0$.

Proof. In the proof of Lemma 18 it was shown that $|w^{(p)}_{q,k} - w^{(q)}_{q,k}| \le \sqrt{n}/T = w_{\min}$, hence if $w^{(p)}_{q,k} = 0$, $w^{(q)}_{q,k} \le w_{\min}$ and $w_{q,k}$ is rounded down to 0. See Algorithm 6. □

Lemma 19. *If agreement has not been reached after $K_{\max} - 1 = 3f$ epochs, then with high probability, there are f corrupt players with weight* 0.

Proof. There are at most f epochs in which the adversary corrupts at least one player. We argue below that for all other epochs, in the call to Weight-Update, the sum of the capacities of edges with at least one endpoint in B is at least 1. This implies that in each iteration of Weight-Update, either some $i \in B$ with $c_V(i) = w_i > w_{\min}$ becomes saturated (and thereafter $w_i = 0$ by Observation 1), or the total weight of all players in B drops by at least 1. Each of these cases also occurs at most f times, hence $K_{\max} - 1 = 3f$ epochs suffice to push the weight of f bad players to zero, with high probability.

We now prove that the sum of the capacities of edges with at least one endpoint in B is at least 1.

$$\begin{aligned}
\sum_{(i,j)\in[n]\times B} c_E(i,j) &= \frac{8}{\epsilon^2 f m T} \sum_{(i,j)\in[n]\times B} \max\{0, \operatorname{corr}(i,j) - w_{i,k}w_{j,k}\beta\} \\
&\ge \frac{8}{\epsilon^2 f m T} \sum_{(i,j)\in G\times B} \max\{0, \operatorname{corr}(i,j) - w_{i,k}w_{j,k}\beta\} \\
&\ge \frac{8}{\epsilon^2 f m T} \left(\frac{1}{8}\epsilon^2 f m T\right) \qquad \text{(by Lemma 11(2))} \\
&= 1.
\end{aligned}$$

□

Lemma 20. *Suppose Invariant 1 holds in an epoch in which corrupt players have zero weight. With high probability, Bracha-Agreement terminates with agreement in this epoch.*

Proof. The proof of Lemma 11(2) states that, with high probability, $-\sum_{(i,j)\in G\times B} \operatorname{corr}(i,j) \ge \frac{1}{4}\epsilon^2 f m T > 0$ in any epoch in which Bracha-Agreement fails to reach agreement. On the other hand, by assumption $-\sum_{(i,j)\in G\times B} \operatorname{corr}(i,j) = 0$. Thus, with high probability, Bracha-Agreement reaches agreement in this epoch. □

Remark 3. Assuming the preconditions of Lemma 20, only good players participate and flip fair coins. Nonetheless, the output of Coin-Flip can still be *strongly* biased by the adversary's scheduling power alone. Consider the following adversarial strategy. Suppose $n = 3f + 1$ and the population is partitioned into $B \cup G_{\text{low}} \cup G_{\text{high}}$, where B is the corrupt set, $|B| = f$, $|G_{\text{low}}| = f + 1$, and $|G_{\text{high}}| = f$. Players in B have weight 0; players in G_{low} have weight $1/(f + 1)$ (consistent with Invariant 1), and players in G_{high} still have weight 1. The adversary permits $B \cup G_{\text{low}}$ to write $m(2f + 1) - 1$ coin flips to a blackboard, that is, one coin-flip shy of completing it. At this point, the weighted sum of these coin flips is some Σ_0, where $|\Sigma_0| = O(\sqrt{m(f + 1)\log n}/(f + 1)) = \tilde{O}(\epsilon^{-2}) = \tilde{O}(n^2)$ with high probability. The adversary then allows the weight-1 players in G_{high} to write coin-flips to the blackboard one-by-one in a round-robin fashion. If the weighted sum ever gets near the origin, for example, within $[-f/3, f/3]$, the adversary can create arbitrary disagreements by having everyone in G_{high} flip one more coin, then having $f/3$ with the same sign write their coin flip to the blackboard, which can be revealed to any subset of the players. At this point, the adversary allows the final coin flip from $B \cup G_{\text{low}}$ to be written, completing the blackboard. The probability of *not* hitting $[-f/3, f/3]$ in a weighted random walk with length $fm = \tilde{O}(n^6)$, starting from $\Sigma_0 = \tilde{O}(n^2)$, is $\tilde{O}(1/n)$.

ALGORITHM 7: Iterated-Blackboard(t) *from the perspective of player p*

1: Set complete(t) $\leftarrow$ *false* and set $\zeta \leftarrow \text{maxlast}^{(p,t-1)}$ if $t > 1$ or any dummy value if $t = 1$. *Broadcast* the write $\text{BB}_t(0,p) \leftarrow \zeta$.

2: **upon** *validating* $\geq n - f$ **ack**$(\text{BB}_t(m(t), q))$'s for $\geq n - f$ different q for the first time:

set complete(t) $\leftarrow$ *true* and $\text{last}^{(p,t)} \leftarrow \text{last}^{(p)}$, then *broadcast* the vector $\text{last}^{(p,t)}$.

3: **upon** *validating* **ack**$(\text{BB}_t(r,p))$'s from $\geq n - f$ different players for the first time:

if $\neg$ complete(t) $\wedge$ $(r < m(t))$

then *generate* a value ζ and *broadcast* the write $\text{BB}_t(r+1,p) \leftarrow \zeta$.

4: **upon** *validating* $\text{BB}_t(r,q)$ from player q for the first time:

set $\text{BB}_t^{(p)}(r,q) \leftarrow \text{BB}_t(r,q)$ and $\text{last}^{(p)}(q) \leftarrow (t,r)$;

if $\neg$ complete(t) **then** *broadcast* **ack**$(\text{BB}_t(r,q))$.

5: **upon** *validating* $\text{last}^{(q,t)}$ vectors from $\geq n - f$ different players q for the first time:

set $\text{maxlast}^{(p,t)}(i) \leftarrow \max_q\{\text{last}^{(q,t)}(i)\}$ (point-wise maximum, lexicographically).

At this point $\text{BB}^{(p,t)} = \left(\text{BB}_1^{(p,t)}, \ldots, \text{BB}_t^{(p,t)}\right)$ is fixed as follows:

$$\text{BB}_{t'}^{(p,t)}(r,i) = \begin{cases} \text{BB}_{t'}^{(p)}(r,i) & \text{if } (t',r) \leq \text{maxlast}^{(p,t)}(i) \text{ (lexicographically) and } r \in [1, m(t)], \\ \perp & \text{otherwise.} \end{cases}$$

This concludes the analysis of Byzantine-Agreement (Algorithm 3). The outstanding claims are now Theorem 6, which is proved in Section 5, and Theorem 17, which is proved in Section 6.

5 THE ITERATED BLACKBOARD

In this section, we prove Theorem 6, restated below, by designing and analyzing an Iterated-Blackboard protocol (Algorithm 7). Our protocol builds on Kimmett's [32] improvement to King and Saia's [33] Blackboard protocol. Throughout this section "t" simply refers to the blackboard index, not the index within an epoch of Byzantine-Agreement.

Theorem 6. *An iterated blackboard* $(\text{BB}_1, \text{BB}_2, \ldots, \text{BB}_\tau)$ *is a sequence of matrices,* BB_t *being an* $m(t) \times n$ *matrix with all cells initially* $\perp$*. It is constructed by calling Iterated-Blackboard*$(1), \ldots,$ *Iterated-Blackboard*(τ)*. In the execution of Iterated-Blackboard*(t)*, every good player* p *fixes a view* $\text{BB}^{(p,t)} = (\text{BB}_1^{(p,t)}, \ldots, \text{BB}_t^{(p,t)})$ *of the true blackboards* $(\text{BB}_1, \ldots, \text{BB}_t)$*. Iterated-Blackboard*$(t)$ *is resilient to* $f < n/3$ *Byzantine faults, and has the following properties.*

(1) Player i writes values successively to cells in column $\text{BB}_t(\cdot, i)$ and only player i may write values to this column. Thus, at all times, $\text{BB}_t(\cdot, i)$ consists of a prefix of non-$\perp$ values and a suffix of $\perp$s.

(2) When player p fixes its historical view $\text{BB}^{(p,t)}$, it contains all writes recorded in $\text{BB}^{(p,t-1)}$, if $t > 1$. Moreover, $\text{BB}_t^{(p,t)}$ contains at least $n - f$ full columns, with $m(t)$ writes, and at most f partial columns. It is guaranteed that for every $t' \in [t]$, $i \in [n]$, $r \in [m(t')]$,

$$\text{BB}_{t'}^{(p,t)}(r,i) \in \{\text{BB}_{t'}(r,i), \perp\}.$$

Moreover, there are at most f tuples (t', r, i) such that for some p, $\text{BB}_{t'}^{(p,t)}(r,i) = \perp \neq \text{BB}_{t'}(r,i)$. All such tuples have distinct 3rd coordinates.

(3) If a player q writes any non-$\perp$ value to BB_t, then by the time any player p fixes $\mathsf{BB}^{(p,t)}$, p can locally reconstruct q's view $\mathsf{BB}^{(q,t-1)}$ of the history up to blackboard $t-1$.

(4) The latency of constructing $\mathsf{BB}^{(p,\tau)}$ is linear in the total number of rows, namely $O(\sum_{t\le\tau} m(t))$.

Iterated-Blackboard uses Reliable-Broadcast (Theorem 1) to construct a series of blackboards $\mathsf{BB} = (\mathsf{BB}_1, \mathsf{BB}_2, \ldots)$, the columns of which are indexed by player IDs in $[n]$ and the rows BB_t are indexed by $[0, m(t)]$. The blackboard proper consists of rows $1, \ldots, m(t)$; the purpose of row zero is to reduce disagreements between the views of good players. Every player p maintains $\mathsf{BB}^{(p)} = (\mathsf{BB}_1^{(p)}, \mathsf{BB}_2^{(p)}, \ldots)$, where $\mathsf{BB}_t^{(p)}(r,i)$ records the value written by player i to $\mathsf{BB}_t(r,i)$ and validated by player p, or $\perp$ if no such value has yet been validated by p. Each player maintains a vector $\mathrm{last}^{(p)}$ indicating the position of the last validated write from each player, that is, $\mathrm{last}^{(p)}(i) = (t,r)$ if p validated i's write to $\mathsf{BB}_t(r,i)$, but has yet to validate any subsequent writes from i to BB_t, nor to $\mathsf{BB}_{t+1}, \mathsf{BB}_{t+2}, \ldots$. Let us emphasize the $\mathsf{BB}^{(p)}$ and $\mathrm{last}^{(p)}$ are *dynamic global variables* of p; they are not tied to any specific invocation of Iterated-Blackboard(t).

Algorithm 7 gives the algorithm Iterated-Blackboard(t) for generating BB_t from the perspective of player p. Player p may only begin executing it if $t = 1$ or if it has fixed $\mathsf{BB}^{(p,t-1)}$ in Iterated-Blackboard($t-1$). At some point in Iterated-Blackboard(t), p fixes a view $\mathsf{BB}^{(p,t)} = (\mathsf{BB}_1^{(p,t)}, \ldots, \mathsf{BB}_t^{(p,t)})$ of the first t blackboards. In particular, when $t > 1$, p begins Iterated-Blackboard(t) having already fixed $\mathsf{BB}^{(p,t-1)}$ and the vector $\mathrm{maxlast}^{(p,t-1)}$, which indicates the position of the last write in $\mathsf{BB}^{(p,t-1)}$ of each player.

The Iterated-Blackboard protocol (Algorithm 7) is defined by five reactive rules, which obscures the flow and structure of the protocol. It consists of three phases: *Initialization* (Line 1), *Populating the Blackboard*, (Lines 3 and 4), and *Synchronization* (Lines 2 and 5). We describe these three phases from the perspective of p.

Initialization. Once p has fixed $\mathsf{BB}^{(p,t-1)}$ and $\mathrm{maxlast}^{(p,t-1)}$ in Iterated-Blackboard($t-1$), p may execute the first step of Iterated-Blackboard(t) (Line 1), which is to write $\mathrm{maxlast}^{(p,t-1)}$ to the zeroth row ($\mathsf{BB}_t(0,p)$), and to initialize the boolean complete(t) to be *false*. (Player p will set complete(t) to be *true* when p sees $n-f$ full columns in $\mathsf{BB}_t^{(p)}$.)

Populating the Blackboard. The blackboard is populated via *writes* and *acknowledgements*, both of which are sent by Reliable-Broadcast. In general, when a player has validated $n-f$ acknowledgements $\mathbf{ack}(\mathsf{BB}_t(r,p))$ to its own write to the rth row, $r < m(t)$, it generates a value ζ and writes it to $\mathsf{BB}_t(r+1,p)$ (Line 3).[16] When p validates one of q's writes, say to $\mathsf{BB}_t(r,q)$, it broadcasts an acknowledgement $\mathbf{ack}(\mathsf{BB}_t(r,q))$, *but only if* complete(t) = *false* (Line 4).

Synchronization. Once a player has recorded $n-f$ full columns (with acknowledgements), it sets the boolean complete(t) $\leftarrow$ *true*, broadcasts the current state of its *last* vector $\mathrm{last}^{(p,t)} \leftarrow \mathrm{last}^{(p)}$ (Line 2), and ceases broadcasting acknowledgements (Line 4). Because of asynchrony, the writes recorded by p and q at the moment they set complete(t) $\leftarrow$ *true* can differ dramatically. The purpose of Line 5 is to reach *near*-agreement on the contents of BB_t. Player p waits until it validates $\mathrm{last}^{(q,t)}$ vectors from $n-f$ players q, and defines $\mathrm{maxlast}^{(p,t)}$ to be their point-wise maximum. By definition, $\mathsf{BB}^{(p,t)}$ records all of q's writes up to $\mathrm{maxlast}^{(p,t)}(q)$. As we shall see, $\mathrm{maxlast}^{(p,t)}$ and $\mathrm{maxlast}^{(q,t)}$ can only disagree in f entries, and by at most 1.

Every message M from q accepted by p in Reliable-Broadcast must be *validated* before Iterated-Blackboard can react to it. In general, p validates M when it has validated messages that, were they

[16] In the context of our Byzantine Agreement protocol, ζ is either v^* (the majority value) or 0 when t is odd, and a uniformly random value in $\{-1, 1\}$ when t is even. See Coin-Flip (Algorithm 4).

to be processed by q, would have resulted in q broadcasting M. Specifically, validation entails the following behavior in each step:

Line 1. Player p will only validate a write $\mathsf{BB}_t(0,q) \leftarrow \zeta$ if $t = 1$, or if $t > 1$ and $\zeta = \text{maxlast}^{(q,t-1)}$ is the point-wise maximum of $n - f$ $\text{last}^{(q',t-1)}$ vectors validated by p.

Lines 2, 4. Player p will only validate an $\mathbf{ack}(\mathsf{BB}_t(r,q))$ if it has validated q's write to $\mathsf{BB}_t(r,q)$.

Line 3. Player p will only validate q's write $\mathsf{BB}_t(r,q) \leftarrow \zeta$, $r > 0$, if ζ is a legal value and it has validated $n - f$ acknowledgements to q's write to $\mathsf{BB}_t(r-1,q)$.[17] The rules for validating a write to $\mathsf{BB}_t(0,q)$ were covered above.

Line 5. Player p will validate $\text{last}^{(q,t)}$ if $\mathsf{BB}_{t'}^{(p)}(r,i) \neq \perp$ whenever $(t',r) \leq \text{last}^{(q,t)}(i)$. In other words, p will validate $\text{last}^{(q,t)}$ once it has recorded *all* the writes that q purports to have recorded, at the moment it set complete$(t) \leftarrow$ *true*.

An important point is that the conditions of the "**upon**" statements of Iterated-Blackboard are checked whenever a message is validated. In particular, player p may record a write to $\mathsf{BB}_t^{(p)}(r,q)$ (Line 3) *after* it has fixed $\mathsf{BB}^{(p,t)}$ and moved on to execute Iterated-Blackboard$(t+1)$. These can be thought of as *retroactive* corrections to BB_t, that will be reflected in $\mathsf{BB}^{(p,t')}$ for some $t' > t$.

In the next lemma, we show that the validation mechanism does not cause deadlocks.

Lemma 21. *Suppose $t = 1$ or $n - f$ good players fix* $\text{maxlast}^{(p,t-1)}$ *and* $\mathsf{BB}^{(p,t-1)}$*. Then every good player that executes Iterated-Blackboard(t) eventually fixes* $\text{maxlast}^{(p,t)}$ *and* $\mathsf{BB}^{(p,t)}$*.*

Proof. First, we claim that, if any good player considers BB_t complete (sets complete$(t) =$ *true*), then every good player that executes Iterated-Blackboard(t) eventually considers BB_t complete. Indeed, if a good player considers BB_t complete, then it must have validated $\mathbf{ack}(\mathsf{BB}_t(m(t),q))$s from at least $n-f$ players, for at least $n-f$ values of q (Line 2). By the properties of reliable broadcast (Theorem 1), every other good player eventually accepts and validates the same messages. Thus, every good player that executes Iterated-Blackboard(t) eventually considers BB_t complete.

Next, we claim that, if at least $n-f$ good players consider BB_t complete (setting their respective variables complete$(t) =$ *true*), then every good player p that executes Iterated-Blackboard(t) will eventually set $\text{maxlast}^{(p,t)}$. Indeed, by Line 2, every good player q that considers BB_t complete will broadcast a $\text{last}^{(q,t)}$ vector. By the properties of reliable broadcast, any blackboard values validated by q will eventually be validated by every good player and, hence, every good player will eventually participate in q's broadcast of $\text{last}^{(q,t)}$. Thus, every good player p eventually validates $\text{last}^{(q,t)}$ vectors from at least $n - f$ different players q and sets $\text{maxlast}^{(p,t)}$ (Line 5).

Finally, since at least $n - f$ good players execute the procedure Iterated-Blackboard(t), by our preceding discussion, it suffices to show that at least one such player considers BB_t complete. Suppose, for a contradiction, that this is not the case. Consider any good player p that executes Iterated-Blackboard(t) with the minimum number of writes to its column of BB_t.

Suppose p writes to row $m(t)$. Then, by minimality, every good player q that executes the procedure Iterated-Blackboard(t) writes to row $m(t)$ in their respective columns, that is, broadcasts a write to $\mathsf{BB}_t(m(t),q)$. By the properties of reliable broadcast, the $n - f$ $\mathbf{ack}(\mathsf{BB}_t(m(t)-1,q))$'s that allow each such player q to broadcast $\mathsf{BB}_t(m(t),q)$ will eventually be validated by every good player. Thus, every good player will eventually validate $\mathsf{BB}_t(m,q)$ and, as they do not consider BB_t complete by assumption, they will broadcast $\mathbf{ack}(\mathsf{BB}_t(m(t),q))$. Therefore, every good player will

[17] In the context of our Byzantine Agreement protocol, ζ is *legal* if $\zeta \in \{-1, 1\}$ when t is even, and $\zeta \in \{v^*, 0\}$ when t is odd. Moreover, in the odd case, each player must only write v^* or 0 consistently; see Coin-Flip (Algorithm 4). The value v^* is possible only if some player broadcast the majority value v^* in Line 4 of Bracha-Agreement, whereas 0 is possible only if $n - f$ players broadcast $\perp$ in Line 4 of Bracha-Agreement.

accept (and validate) at least $n-f$ **ack**$(\mathrm{BB}_t(m(t),q))$ for at least $n-f$ different q and, consequently, consider BB_t complete (Line 2), which is a contradiction.

Now suppose p last writes to row $r < m(t)$. If $r > 0$, then the $n-f$ **ack**$(\mathrm{BB}_t(r-1,p))$'s that allow p to broadcast $\mathrm{BB}_t(r,p)$ will eventually be validated by every good player. Hence, every good player will eventually participate in p's broadcast of $\mathrm{BB}_t(r,p)$ and validate $\mathrm{BB}_t(r,p)$. Similarly, if $r = 0$. Since p does not write to row $r+1 \le m(t)$, it never validates $n-f$ **ack**$(\mathrm{BB}_t(r,p))$'s. Since at least $n-f$ good players validate $\mathrm{BB}_t(r,p)$, it follows that at least one such player does not broadcast an **ack**$(\mathrm{BB}_t(r,p))$. By Line 4, this player must have considered BB_t to be complete by the time it validates $\mathrm{BB}_t(r,p)$, which is a contradiction.

Therefore, in both cases, we reach the desired contradiction. □

Recall that after executing Iterated-Blackboard(t), p's view of the history is $\mathrm{BB}^{(p,t)}$, defined to be:

$$\mathrm{BB}^{(p,t)}_{t'}(r,i) = \begin{cases} \mathrm{BB}^{(p)}_{t'}(r,i) & \text{if } r \in [1,m(t)] \text{ and } (t',r) \le \mathrm{maxlast}^{(p,t)}(i) \text{ (lexicographically)} \\ \perp & \text{otherwise} \end{cases}$$

In other words, we obtain $\mathrm{BB}^{(p,t)}_{t'}$ by stripping off the zeroth row of every $\mathrm{BB}^{(p)}_{t'}$ matrix and replacing any values in column i after $\mathrm{maxlast}^{(p,t)}(i)$ with $\perp$. We emphasize that, in contrast to p's local blackboard variable $\mathrm{BB}^{(p)}_t$, once $\mathrm{BB}^{(p,t)}$ is set, it never changes.

Lemma 22. *Suppose some good player q validates the* ***ack***$(\mathrm{BB}_t(r,i))$ *messages broadcast from at least $n-f$ different players. Then* every *good player p that finishes iteration t has*

$$\mathrm{BB}^{(p)}_t(r,i) = \mathrm{BB}^{(p,t)}_t(r,i) = \mathrm{BB}_t(r,i).$$

Proof. Let S_0 be the set of players broadcasting **ack**$(\mathrm{BB}_t(r,i))$ messages that q validates. It follows that $\mathrm{last}^{(q',t)}(i) \ge (t,r)$ for all $q' \in S_0$. When p finishes iteration t, it validates $\mathrm{last}^{(q',t)}$ vectors for $n-f$ players $q' \in S_1$. Since $S_0 \cap S_1 \neq \emptyset$, $\mathrm{maxlast}^{(p,t)}(i) = \max_{q' \in S_1}\{\mathrm{last}^{(q',t)}(i)\} \ge (t,r)$, meaning that p will not finish Line 5 until it accepts and validates player i's write to $\mathrm{BB}_t(r,i)$, recording it in $\mathrm{BB}^{(p)}_t$ and hence $\mathrm{BB}^{(p,t)}_t$. □

Lemma 23. *Suppose that each good player executes Iterated-Blackboard*(1)*, …, Iterated-Blackboard*(t)*, beginning iteration $t'+1$ only after it has executed Line 5 of iteration t'. Then, for any two good players p,q that finish iteration t, $\mathrm{BB}^{(p,t)}$ and $\mathrm{BB}^{(q,t)}$ disagree in at most f positions* in total. *If they disagree on the contents of any position, one is $\perp$.*

Proof. The properties of reliable broadcast ensures that $\mathrm{BB}^{(p)}_t$ and $\mathrm{BB}^{(q)}_t$ cannot contain *distinct* non-$\perp$ values in any position. Therefore, we must argue that they differ in at most f positions. Fix a player i and let $\mathrm{BB}_{t_i}(r_i,i)$ be the *last* of i's blackboard writes for which it validated at least $n-f$ **ack**$(\mathrm{BB}_{t_i}(r_i,i))$s. By Lemma 22, $\mathrm{BB}^{(p,t)}_{t_i}(r_i,i) = \mathrm{BB}^{(q,t)}_{t_i}(r_i,i)$. Moreover, p,q have both validated all of i's blackboard writes prior to $\mathrm{BB}_{t_i}(r_i,i)$. Subsequent blackboard writes of i that could appear in the local matrices of p and q are limited to $\mathrm{BB}_{t_i}(r_i+1,i)$ (if $r_i < m(t_i)$) and $\mathrm{BB}_{t_i+1}(0,i), \ldots, \mathrm{BB}_t(0,i)$ (if $t_i < t$). This follows by assumption on (t_i,r_i): at the first time when both p and q finish iteration t, i has not validated sufficiently many acknowledgements to attempt any writes beyond these. Since we strip off the zeroth rows of each local view to form $\mathrm{BB}^{(p,t)}$, $\mathrm{BB}^{(q,t)}$, they may only disagree in column i at $\mathrm{BB}_{t_i}(r_i+1,i)$. Now, for at least $n-f$ players i we have $(t_i,r_i) = (t,m(t))$. Thus, $\mathrm{BB}^{(p,t)}$, $\mathrm{BB}^{(q,t)}$ may only disagree in f cells in total. □

Lemma 24. *If $\mathrm{BB}_t(1,q) \neq \perp$, then by the time p fixes $\mathrm{BB}^{(p,t)}$, it can reconstruct q's history $\mathrm{BB}^{(q,t-1)}$ through blackboard $t-1$, assuming $t > 1$.*

Proof. Before q wrote anything to $\mathrm{BB}_t(1,q)$ it must have written $\mathrm{BB}_t(0,q) \leftarrow \mathrm{maxlast}^{(q,t-1)}$ and caused $n-f$ acknowledgements $\mathbf{ack}(\mathrm{BB}_t(0,q))$ to be broadcast. By Lemma 22 every player will accept and validate q's write to $\mathrm{BB}_t(0,q)$ before fixing $\mathrm{BB}^{(p,t)}$, and hence be able to reconstruct $\mathrm{BB}^{(q,t-1)}$ from $\mathrm{maxlast}^{(q,t-1)}$. □

This concludes the proof of Theorem 6.

6 RISING-TIDE IS LIPSCHITZ CONTINUOUS

The goal of this section is to prove Theorem 17, which is restated below.

Theorem 17. *Let $H^{(p)} = (V, E, c_V^{(p)}, c_E^{(p)})$ and $H^{(q)} = (V, E, c_V^{(q)}, c_E^{(q)})$ be two capacitated graphs, which differ by $\eta_E = \sum_{i,j} |c_E^{(p)}(i,j) - c_E^{(q)}(i,j)|$ in their edge capacities and $\eta_V = \sum_i |c_V^{(p)}(i) - c_V^{(q)}(i)|$ in their vertex capacities. Let $\mu^{(p)}$ and $\mu^{(q)}$ be the fractional matchings computed by* Rising-Tide *(Algorithm 5). Then:*

$$\sum_i \left| \left(c_V^{(p)}(i) - \sum_j \mu^{(p)}(i,j) \right) - \left(c_V^{(q)}(i) - \sum_j \mu^{(q)}(i,j) \right) \right| \le \eta_V + 2\eta_E.$$

6.1 Dependency Graphs

We first introduce the idea of a *dependency graph* that captures the moments when vertices become saturated in Rising-Tide (Algorithm 5). We will then use structural properties of dependency graphs to finally prove Theorem 17. Throughout this section, the two capacitated graphs under consideration are $G = (V, E, c_V^G, c_E^G)$ and $H = (V, E, c_V^H, c_E^H)$.

Definition 2 (Dependency Graph). Let D_G be a *directed* graph on the same vertex set: $V(D_G) = V$. Consider the execution of Algorithm 5 on G. For each edge $e = \{i,j\} \in E$, if at the moment e is removed from the working set E' (Line 7), i (resp. j) is saturated, then we include in D_G a directed edge $j \to i$ (resp. $i \to j$). Notice that if both i and j are saturated simultaneously, then D_G includes both edges $i \to j$ and $j \to i$.

We first state a useful continuity property of Rising-Tide, that if we continuously deform the input capacities, the output fractional matching also changes continuously.

Lemma 25 (The Continuity Lemma). *Let G and H be two fractional matching instances where every vertex- and edge-capacity differs by at most ξ. Then, for every edge e, $|\mu_G(e) - \mu_H(e)| \le F(n)\xi$ for some function F which depends only on the size of the graph.*

Proof. Without loss of generality, we can assume that each edge capacity $c_E^G(i,j) \le \min\{c_V^G(i), c_V^G(j)\}$ is always bounded by the capacities of its endpoints.

Imagine running Rising-Tide simultaneously on both G and H, stopping at the first saturation event that occurs in, say, G but not H. (A "saturation event" is the saturation of a vertex or edge with *non-zero* capacity.) Let μ'_G, μ'_H be the fractional matchings at this time and G', H' be the residual graphs, that is, obtain new capacities by subtracting each $\mu'_G(i,j)$ from $c_V^G(i), c_V^G(j)$, and $c_E^G(i,j)$. The maximum difference in vertex- or edge-capacities between G', H' is $n\xi$. The argument can be applied inductively to G', H', and since there are $O(n^2)$ saturation events, the maximum difference between any capacity (and hence an μ-value) is always bounded by $F(n)\xi$, where $F(n) = n^{O(n^2)}$. □

The magnitude of F is immaterial to our argument, so long as it depends only on n. Lemma 25 allows us to make several simplifying assumptions, which are ultimately justified in the final proof of Theorem 17.

A1. Although we are comparing two graphs G, H with possibly many capacity differences, we can assume without loss of generality that they differ in precisely *one* vertex- or edge-capacity.
A2. We can assume that the dependency graphs for G and H are identical.
A3. We can assume, via infinitesimal perturbations, that no two vertices are saturated simultaneously. In particular, this implies that D_G is acyclic. (See Lemma 26.)

Lemma 26 (Basic Properties of μ and D_G). *Assume graph G satisfies assumption (A3). Let μ be the output of G from Algorithm 5. Then:*

(1) For any two edges $e_1 \in E$ and $e_2 \in E$, if e_1 gets removed from E' before e_2, then $\mu(e_1) < \mu(e_2)$.
(2) For each $u \in V$, all edges directed towards u in D_G have the same μ-value.
(3) For any edge $u \to v$ in D_G and any edge $\{v, w\} \in E$, $\mu(u, v) \geq \mu(v, w)$.
(4) (Monotonic Path Property) The μ-values along any directed walk $u_0 \to u_1 \to u_2 \to \cdots$ on D_G are non-increasing. That is, $\mu(u_0, u_1) \geq \mu(u_1, u_2) \geq \cdots$.
(5) (Directed Acyclic Graph Property) D_G is a DAG.

Proof. To show (1), it suffices to observe that in Algorithm 5 the fractional matching μ grows strictly increasing at each iteration.

To show (2), it suffices to show that for each vertex $u \in V$ with any two incoming edges $v \to u$ and $w \to u$ on D_G, $\mu(v, u) = \mu(w, u)$. Suppose conversely and without loss of generality $\mu(v, u) > \mu(w, u)$. By the time $\{w, u\}$ gets removed from the working set E', u is already saturated. However, it is now impossible to increase $\mu(v, u)$ anymore, contradicting the assumption that $\mu(v, u) > \mu(w, u)$.

To show (3), we notice that at the time $\{u, v\}$ is removed from E', v is saturated. At this moment, any edge $\{v, w\} \in E$ incident to v cannot increase its μ value anymore. Hence, $\{v, w\}$ will be removed from E' at the same time with $\{u, v\}$ or prior to the time when $\{u, v)\}$ is removed from E'. Thus, by (1) we have $\mu(u, v) \geq \mu(v, w)$. (4) follows from (3).

To show (5), assume that there exists a cycle $u_0 \to u_1 \to \cdots \to u_0$ in D_G. By the monotonic path property (4), all edges $\mu(u_i, u_{i+1})$ have the same fractional value when they were removed from the working set E' in the Rising-Tide algorithm. Moreover, by definition of D_G, all vertices are simultaneously saturated, which contradicts (A3). □

Henceforth, (A3) is assumed to hold in all graphs.

6.2 The Proof

To prove Theorem 17, it suffices to show (via Lemma 25 and an interpolation argument) that the statement holds in the special case that (c_V^G, c_E^G) and (c_V^H, c_E^H) differ in exactly one vertex capacity or exactly one edge capacity, and that $D_G = D_H$.

We start with some observations when there is only one change on the capacities between G and H.

Lemma 27. *Let G and H be two input graphs with $c_E^G = c_E^H$ and $D \stackrel{\text{def}}{=} D_G = D_H$. If $\{i, j\}$ is an edge for which neither $i \to j$ nor $j \to i$ appear in D, then $\mu_G(i, j) = \mu_H(i, j)$.*

Proof. If both i and j are not saturated by the time $\{i, j\}$ gets removed from E', then $\{i, j\}$ itself must be saturated, hence $\mu_G(i, j) = c_E^G(i, j) = c_E^H(i, j) = \mu_H(i, j)$. □

Suppose graphs G and H have the same capacities except at some vertex $s \in V$. Then Lemma 27 implies that if we run the Rising-Tide algorithm on both instances G and H, the first moment they differ must be when one graph saturates s but the other does not. In this case, we can think of s being the *source* of all disagreements. Intuitively, if we look at an edge e where $\mu_G(e) \neq \mu_H(e)$, we should be able to trace this disagreement back to s.

Lemma 28. *Assume that G and H differ only in the capacity of one vertex s and that $D \stackrel{\text{def}}{=} D_G = D_H$. Consider any edge $\{i,j\}$ such that $\mu_G(i,j) \neq \mu_H(i,j)$. Then, there exists a path in the dependency graph D containing $\{i,j\}$ and s. Moreover, every edge e on this path satisfies $\mu_G(e) \neq \mu_H(e)$.*

Proof. Without loss of generality, when we consider an edge $\{i,j\}$ with $\mu_G(i,j) < \mu_H(i,j)$, we may always assume $j \to i$ appears in D. (This edge must exist by Lemma 27.) That is, when $\{i,j\}$ is removed from the Rising-Tide algorithm, it is because i is saturated. Now we prove this lemma by induction on all edges from the smallest μ_G value to the largest μ_G value.

Base Case. Suppose $\{i,j\}$ is one of the edges with the minimum μ_G-value such that $\mu_G(i,j) < \mu_H(i,j)$. Since this is the first moment when the algorithm behaves differently, and we assume that $j \to i$ on G, it follows that at time $\mu_G(i,j)$, the vertex i is saturated in G but not in H. Moreover, all other edges incident to i have the same μ_G-value at this time. Therefore $c_V^G(i) < c_V^H(i)$, and hence $i = s$. There is a trivial path in D including $\{j,i\}$ and s.

Inductive Case. Now let us prove the inductive case. Suppose $\mu_G(i,j) \neq \mu_H(i,j)$ and when $\{i,j\}$ is removed from E', the vertex i is saturated. If $i = s$ then we are done. Otherwise, we have $c_V^G(i,j) = c_V^H(i,j)$. By Lemma 26(2,3) and summing up all fractional matching values around the vertex i, we know that there exists an edge $\{i,j'\}$ with $\mu_G(i,j') \neq \mu_G(i,j)$ and also $\mu_G(i,j') \neq \mu_H(i,j')$. By Lemma 26(1) we know that $\mu_G(i,j') < \mu_G(i,j)$. By the induction hypothesis and Lemma 27, we know that $i \to j'$ in D and there must be a path from $\{i,j'\}$ to s in D. Therefore, there exists a path including $\{j,i\}$ and s in D as well. □

Now, we prove the simplest version of Theorem 17, where G, H differ in one vertex capacity and have the same dependency graph.

Lemma 29. *Assume G and H only differ in the capacity of one vertex s, and that $D \stackrel{\text{def}}{=} D_G = D_H$. Then, the total differences among the remaining vertex capacities can be bounded by*

$$\sum_i \left| \left(c_V^G(i) - \sum_j \mu_G(i,j) \right) - \left(c_V^H(i) - \sum_j \mu_H(i,j) \right) \right| \leq \left| c_V^G(s) - c_V^H(s) \right|.$$

Proof. By Lemma 28, all edges that have different fractional matching values form a subgraph D_{diff} of D with s being the only minimal element. If s is not saturated then there are no incoming edges to s. By Lemma 28, we know that $D_{\text{diff}} = \emptyset \implies \mu_G = \mu_H$ and in this case the equality holds for the statement.

Observe that whenever there is an incoming edge to a vertex i in D, the vertex i must be saturated. Since we are measuring differences in the remaining vertex capacities, the only place where such disagreement could happen is on all maximal vertices of D_{diff}. Let T be the set of maximal vertices, that is, those without incoming edges.

We prove a certain inequality by induction over all sets S such that $S \subseteq V - T$ and S is *downward closed*, meaning there is no outgoing edge from S to $V - S$. As a consequence $s \in S$. Let ∂S be the set of incoming edges from $V - S$ to S. We will prove that for any coefficients $\{\nu_{i\to j} \in [-1,1]\}_{(i\to j)\in\partial S}$ we have

$$\left| \sum_{(i\to j)\in\partial S} \nu_{i\to j} (\mu_G(i,j) - \mu_H(i,j)) \right| \leq \left| c_V^G(s) - c_V^H(s) \right|.$$

Base Case. The minimal downward closed set is $S = \{s\}$. By Lemma 26(2) all incoming edges have the same $\mu_G(i,s) - \mu_H(i,s)$ values. That is, all terms in $\{\mu_G(i,s) - \mu_H(i,s)\}$ are of the same sign and hence the claim is true for the base case.

Inductive Case. Consider any downward-closed set $S \subseteq V - T$ with $|S| \geq 2$, and let $\{v_{i\to j} \in [-1,1]\}$ be any set of coefficients on the fringe ∂S. Let $u \neq s$ be any maximal element in S.

Let X_{in} and X_{out} be the set of incoming and outgoing edges incident to u. Since S is downward-closed, we have

$$\partial S = \partial(S - \{u\}) \cup X_{\text{in}} - X_{\text{out}}.$$

Now, by Lemma 26 we know that all incoming edges $(i \to u)$ in X_{in} have the same fractional matching value in μ_G, and the same value in μ_H. We denote the difference between these values by $\Delta \stackrel{\text{def}}{=} \mu_G(i,u) - \mu_H(i,u)$.

Let $v_u = \frac{1}{|X_{\text{in}}|}\left(\sum_{(i\to u)\in X_{\text{in}}} v_{i\to u}\right) \in [-1,1]$ be the average coefficient among all incoming edges. Since u is saturated, we have

$$\begin{aligned}
&\sum_{(u\to j)\in X_{\text{out}}} v_u(\mu_G(u,j) - \mu_H(u,j)) + \sum_{(i\to u)\in X_{\text{in}}} v_{i\to u}(\mu_G(i,u) - \mu_H(i,u)) \\
&= \sum_{(u\to j)\in X_{\text{out}}} v_u(\mu_G(u,j) - \mu_H(u,j)) + v_u|X_{\text{in}}| \cdot \Delta && \text{(by definition of } v_u\text{)} \\
&= v_u\left(\sum_{(u\to j)\in X_{\text{out}}} (\mu_G(u,j) - \mu_H(u,j)) + \sum_{(i\to u)\in X_{\text{in}}} (\mu_G(i,u) - \mu_H(i,u))\right) \\
&= v_u(c_V^G(u) - c_V^H(u)) \\
&= 0.
\end{aligned}$$

By removing u from S we obtain a smaller subset on which we can apply the inductive hypothesis. Define coefficients $\{v'_{i\to j}\}$ with $v'_{u\to j} = -v_u$ for all $(u \to j) \in X_{\text{out}}$ and $v'_{i\to u} = v_{i\to j}$ for all unrelated edges not incident to u. Then, we have

$$\begin{aligned}
&\left|\sum_{(i\to j)\in\partial S} v_{i\to j}(\mu_G(i,j) - \mu_H(i,j))\right| \\
&\leq \left|\left(\sum_{(i\to j)\in\partial(S-\{u\})} v'_{i\to j}(\mu_G(i,j) - \mu_H(i,j))\right) + v_u \cdot 0\right| && \text{(vertex } u \text{ is saturated)} \\
&\leq \left|c_V^G(i) - c_V^H(i)\right|. && \text{(by induction hypothesis)}
\end{aligned}$$

By choosing $S = V \setminus T$ and coefficients $v_{i\to j} = \operatorname{sgn}(\mu_G(i,j) - \mu_H(i,j))$ for every edge $(i \to j) \in \partial S$, we conclude that

$$\begin{aligned}
&\sum_i \left|\left(c_V^G(i) - \sum_j \mu_G(i,j)\right) - \left(c_V^H(i) - \sum_j \mu_H(i,j)\right)\right| \\
&= \sum_{i\neq s}\left|\sum_j \mu_G(i,j) - \sum_j \mu_H(i,j)\right| && \text{(for all } i \neq s,\ c_V^G(i) = c_V^H(i), \text{ and } s \text{ is saturated)} \\
&= \sum_{i\in T}\left|\sum_j \mu_G(i,j) - \sum_j \mu_H(i,j)\right| \\
&= \sum_{(i\to j)\in\partial S} v_{i\to j}(\mu_G(i,j) - \mu_H(i,j)) && \text{(use } v_{i\to j} \text{ to remove the absolute value operation)}
\end{aligned}$$

Fig. 1. Left: suppose G, H differ only in the capacity of edge $\{s,t\}$. Right: applying the transformation, G', H' now only differ in the capacity of vertex x.

$$= \left| \sum_{(i \to j) \in \partial S} \nu_{i \to j}(\mu_G(i,j) - \mu_H(i,j)) \right| \qquad \text{(this sum is positive)}$$
$$\leq \left| c_V^G(s) - c_V^H(s) \right|.$$

□

Lemma 29 handled the case when G, H differ in one *vertex* capacity. Lemma 30 reduces the case where they differ in one *edge* capacity to Lemma 29.

Lemma 30. *Assume that G and H differ only in the capacity of one edge $\{s,t\} \in E$. Assume that $D \stackrel{\text{def}}{=} D_G = D_H$. Then,*

$$\sum_i \left| \left(c_V^G(i) - \sum_j \mu_G(i,j) \right) - \left(c_V^H(i) - \sum_j \mu_H(i,j) \right) \right| \leq 2 \left| c_E^G(s,t) - c_E^H(s,t) \right|.$$

Proof. By reduction to Lemma 29. Create G' by subdividing $\{s,t\}$ into $\{s,x\}, \{x,t\}$ with $c_E^{G'}(s,x) = c_E^{G'}(x,t) = \infty$ and $c_V^{G'}(x) = 2c_E^G(s,t)$. Create H' from H in the same way. Since $D_G = D_H$, the same vertices must be saturated in both, and in particular, among s, t, and $\{s,t\}$, both executions saturate the same element first. If they both saturate s or t first, then the capacity of $\{s,t\}$ has no influence on the execution and $\mu_G = \mu_H$. If they both saturate $\{s,t\}$ first, then the executions on G, H proceed identically to the corresponding executions on G', H'. Note that G', H' differ in one vertex capacity, with $\left| c_V^{G'}(x) - c_V^{H'}(x) \right| = 2 \left| c_E^G(s,t) - c_E^H(s,t) \right|$. The lemma then follows from Lemma 29 applied to G', H'. □

We can now prove Theorem 17.

Proof of Theorem 17. Imagine continuously transforming the capacities (c_V^G, c_E^G) into (c_V^H, c_E^H) by modifying one vertex capacity or one edge capacity at a time. In this continuous process, there are two types of *breakpoints* to pay attention to. The first is when we switch from transforming one capacity to another, and the second is when the dependency graph changes. Let $G = G_0, G_1, \ldots, G_k = H$ be the sequence of graphs at these breakpoints. Up to a tie-breaking perturbation, we can assume each pair (G_i, G_{i+1}) differ in one edge or vertex capacity, and have the same dependency graph. By Lemma 25 the objective function is continuous in the input, and does not have any discontinuities at breakpoints. Let $\eta_V(i), \eta_E(i)$ be, respectively, the difference in vertex and edge capacities between G_i and G_{i+1}. By Lemmas 29 and 30 the objective function is bounded by $\sum_{i=0}^{k-1} (\eta_V(i) + 2\eta_E(i)) = \eta_V + 2\eta_E$. □

7 CONCLUSION

Our main result is the first polynomial-latency agreement protocol in the *full-information* model resilient to $f = n/(3+\epsilon)$ *adaptive* Byzantine failures. When ϵ is bounded away from zero it has expected latency $\tilde{O}(n^4)$ and in the extreme case when $\epsilon = 1/f$ ($n = 3f + 1$) it has latency $\tilde{O}(n^{12})$.

This is the first improvement to Bracha's [13] 1984 protocol when $n = 3f + 1$, and improves on the resilience of King and Saia's [33, 34] protocols; see Table 1.

We see some interesting directions for future work.

— Let $n^{g(\epsilon)}$ be the optimum latency of a Byzantine Agreement protocol with resiliency $f = n/(3 + \epsilon)$ against an adaptive adversary. What does the function g look like and what is $\lim_{\epsilon\to\infty} g(\epsilon)$? From [33] we know that $\lim_{\epsilon\to\infty} g(\epsilon) \leq 2.5$, at least for protocols with exponential local computation, and from [4, 5] we know $\lim_{\epsilon\to\infty} g(\epsilon) \geq 1$. What is the correct limit of $g(\epsilon)$? Are there qualitatively different protocols achieving latency $n^{g(\epsilon)}$ for various ranges of ϵ? One can also look at the optimal latency-resiliency tradeoff when $f = n^{\gamma}$, $\gamma \in (0, 1)$, is expressed as a polynomial of n.

— Each step in the protocols we use (Reliable-Broadcast and Bracha-Agreement [13], Iterated-Blackboard, and Coin-Flip) typically consists of sending a message to all players and waiting for $n - f$ messages before making some state transition. If we were to wait for $n - f + 1$ messages, we may wait forever if f players crashed and never sent any messages. On the other hand, once we introduce *blacklisting* it is not clear that waiting for just $n - f$ messages is necessary anymore. For example, suppose that $\sum_i w_i = n - 2\rho f$ and that we have reduced the weight of good and bad players each by ρf, with high probability. Rather than wait for $n - f$ messages, we could wait for messages from players whose *total weight* is at least $n - (\rho + 1)f$.[18] This would help speed up later epochs since we could then access the weight advantage of good players. However, since there is *some* non-zero probability of blacklisting pairs of good players, there is *some* non-zero probability that a protocol will deadlock if it waits for $n-(\rho+1)f$ weight before proceeding. This raises the possibility that there is a complexity separation in Byzantine agreement between *Las Vegas* protocols (which terminate in agreement with probability 1) and *Monte Carlo* protocols (which terminate in agreement with probability $1-o(1)$, and may deadlock or terminate without agreement with $o(1)$ probability). Cf. [31].

ACKNOWLEDGEMENT

We thank the two reviewers for suggesting numerous improvements to the presentation.

REFERENCES

[1] Ittai Abraham, Danny Dolev, and Joseph Y. Halpern. 2008. An almost-surely terminating polynomial protocol forasynchronous Byzantine agreement with optimal resilience. In *Proceedings of the 27th Annual ACM Symposium on Principles of Distributed Computing*. 405–414. DOI: https://doi.org/10.1145/1400751.1400804

[2] Miklós Ajtai and Nathan Linial. 1993. The influence of large coalitions. *Combinatorica* 13, 2 (1993), 129–145. DOI: https://doi.org/10.1007/BF01303199

[3] Noga Alon and Moni Naor. 1993. Coin-flipping games immune against linear-sized coalitions. *SIAM Journal on Computing* 22, 2 (1993), 403–417. DOI: https://doi.org/10.1137/0222030

[4] James Aspnes. 1998. Lower bounds for distributed coin-flipping and randomized consensus. *Journal of the ACM* 45, 3 (1998), 415–450. DOI: https://doi.org/10.1145/278298.278304

[5] Hagit Attiya and Keren Censor. 2008. Tight bounds for asynchronous randomized consensus. *Journal of the ACM* 55, 5 (2008), 1–26.

[6] Hagit Attiya and Jennifer L. Welch. 2004. *Distributed Computing, 2nd Ed.* Wiley.

[7] Laasya Bangalore, Ashish Choudhury, and Arpita Patra. 2020. The power of shunning: Efficient asynchronous Byzantine agreement revisited. *Journal of the ACM* 67, 3 (2020), 14:1–14:59. DOI: https://doi.org/10.1145/3388788

[8] Michael Ben-Or. 1983. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing*. 27–30. DOI: https://doi.org/10.1145/800221.806707

[18] More generally, players should make a state transition only when it is *plausible, w.h.p.*, that all un-received messages were to be sent by Byzantine players.

[9] Michael Ben-Or and Nathan Linial. 1985. Collective coin flipping, robust voting schemes and minima of Banzhaf values. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*. 408–416. DOI: https://doi.org/10.1109/SFCS.1985.15

[10] Michael Ben-Or, Elan Pavlov, and Vinod Vaikuntanathan. 2006. Byzantine agreement in the full-information model in $O(\log n)$ rounds. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. 179–186.

[11] Piotr Berman and Juan A. Garay. 1993. Randomized distributed agreement revisited. In *Proceedings of the 23rd Annual International Symposium on Fault-Tolerant Computing*. 412–419. DOI: https://doi.org/10.1109/FTCS.1993.627344

[12] Ravi B. Boppana and Babu O. Narayanan. 2000. Perfect-information leader election with optimal resilience. *SIAM Journal on Computing* 29, 4 (2000), 1304–1320. DOI: https://doi.org/10.1137/S0097539796307182

[13] Gabriel Bracha. 1987. Asynchronous Byzantine agreement protocols. *Information and Computation* 75, 2 (1987), 130–143. DOI: https://doi.org/10.1016/0890-5401(87)90054-X

[14] Gabriel Bracha and Sam Toueg. 1985. Asynchronous consensus and broadcast protocols. *Journal of the ACM* 32, 4 (1985), 824–840. DOI: https://doi.org/10.1145/4221.214134

[15] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. *Journal of Cryptology* 18, 3 (2005), 219–246. DOI: https://doi.org/10.1007/s00145-005-0318-0

[16] Ran Canetti and Tal Rabin. 1993. Fast asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*. 42–51. DOI: https://doi.org/10.1145/167088.167105

[17] Benny Chor and Brian A. Coan. 1985. A simple and efficient randomized Byzantine agreement algorithm. *IEEE Transactions on Software Engineering* 11, 6 (1985), 531–539.

[18] D. P. Dubhashi and A. Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

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

[20] Uriel Feige. 1999. Noncryptographic selection protocols. In *Proceedings 40th Annual IEEE Symposium on Foundations of Computer Science*. 142–153. DOI: https://doi.org/10.1109/SFFCS.1999.814586

[21] Pesech Feldman and Silvio Micali. 1997. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing* 26, 4 (1997), 873–933. DOI: https://doi.org/10.1137/S0097539790187084

[22] Michael J. Fischer and Nancy A. Lynch. 1982. A lower bound for the time to assure interactive consistency. *Information Processing Letters* 14, 4 (1982), 183–186. DOI: https://doi.org/10.1016/0020-0190(82)90033-3

[23] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. 1986. Easy impossibility proofs for distributed consensus problems. *Distributed Computing* 1, 1 (1986), 26–39. DOI: https://doi.org/10.1007/BF01843568

[24] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM* 32, 2 (1985), 374–382. DOI: https://doi.org/10.1145/3149.214121

[25] Juan A. Garay and Yoram Moses. 1998. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM Journal on Computing* 27, 1 (1998), 247–290. DOI: https://doi.org/10.1137/S0097539794265232

[26] Shafi Goldwasser, Elan Pavlov, and Vinod Vaikuntanathan. 2006. Fault-tolerant distributed computing in full-information networks. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. 15–26. DOI: https://doi.org/10.1109/FOCS.2006.30

[27] Iftach Haitner and Yonatan Karidi-Heller. 2020. A tight lower bound on adaptively secure full-information coin flip. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. 1268–1276. DOI: https://doi.org/10.1109/FOCS46700.2020.00120

[28] Shang-En Huang, Seth Pettie, and Leqi Zhu. 2022. Byzantine agreement in polynomial time with near-optimal resilience. In *Proceedings of the 54th ACM Symposium on Theory of Computing*. 502–514.

[29] Shang-En Huang, Seth Pettie, and Leqi Zhu. 2023. Byzantine agreement with optimal resilience via statistical fraud detection. In *Proceedings of the 34th ACM-SIAM Symposium on Discrete Algorithms*. 4335–4353.

[30] Jeff Kahn, Gil Kalai, and Nathan Linial. 1988. The influence of variables on boolean functions (extended abstract). In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*. 68–80. DOI: https://doi.org/10.1109/SFCS.1988.21923

[31] Bruce M. Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. 2010. Fast asynchronous Byzantine agreement and leader election with full information. *ACM Transactions on Algorithms* 6, 4 (2010), 68:1–68:28. DOI: https://doi.org/10.1145/1824777.1824788

[32] Ben Kimmett. 2020. *Improvement and partial simulation of King & Saia's expected-polynomial-time Byzantine agreement algorithm*. Master's Thesis. University of Victoria, Canada.

[33] Valerie King and Jared Saia. 2016. Byzantine agreement in expected polynomial time. *Journal of the ACM* 63, 2 (2016), 13:1–13:21. DOI: https://doi.org/10.1145/2837019

[34] Valerie King and Jared Saia. 2018. Correction to Byzantine agreement in expected polynomial time, JACM 2016. arXiv:1812.10169. Retrieved from http://arxiv.org/abs/1812.10169

[35] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (1982), 382–401. DOI: https://doi.org/10.1145/357172.357176

[36] Allison B. Lewko. 2011. The contest between simplicity and efficiency in asynchronous Byzantine agreement. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC) (Lecture Notes in Computer Science)*, Vol. 6950. 348–362. DOI: https://doi.org/10.1007/978-3-642-24100-0_35

[37] Nancy A. Lynch. 1996. *Distributed Algorithms*. Morgan Kaufmann.

[38] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. 1980. Reaching agreement in the presence of faults. *Journal of the ACM* 27, 2 (1980), 228–234. DOI: https://doi.org/10.1145/322186.322188

[39] Michael O. Rabin. 1983. Randomized Byzantine generals. In *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science*. 403–409. DOI: https://doi.org/10.1109/SFCS.1983.48

[40] Alexander Russell, Michael E. Saks, and David Zuckerman. 2002. Lower bounds for leader election and collective coin-flipping in the perfect information model. *SIAM Journal on Computing* 31, 6 (2002), 1645–1662. DOI: https://doi.org/10.1137/S0097539700376007

[41] Michael E. Saks. 1989. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics* 2, 2 (1989), 240–244. DOI: https://doi.org/10.1137/0402020

[42] Sam Toueg. 1984. Randomized Byzantine agreements. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*. 163–178. DOI: https://doi.org/10.1145/800222.806744

Received 10 January 2023; revised 30 August 2023; accepted 15 December 2023