Randomized Early-Stopping Byzantine Agreement with Applications to Round-Efficient MPC *

Michele Ciampi† Pouyan Forghani‡ Yun Lu§ Vassilis Zikas†

November 18, 2026

Abstract

We consider two main classes of synchronous round-efficient Byzantine Agreement (BA) protocols: *expected constant rounds* and *early-stopping*. Randomized protocols can terminate in expected constant rounds, and early-stopping protocols' worst-case round complexity depends on the actual number of corrupted parties $f \leq t$, with $\min(t+1, f+2)$ being optimal. In this work, we combine the benefits of both classes, proposing a protocol that has constant round-complexity in expectation and $O(f)$ in the worst case. We present our protocol in the form of a compiler that can lift a broad class of expected-constant-round BA protocols, including Feldman–Micali and Katz–Koo, to *also* achieve near-optimal $O(f)$ worst-case complexity.

This yields the *first* treatment of perfectly secure BA protocols for $t < n/3$ that is both expected constant-round and near-optimal ($O(f)$) early stopping. Assuming a PKI, we show a variant of the compiler with a better round complexity, tolerating $t < n/2$ corruptions.

We also extend our results to parallel broadcast, also known as interactive consistency (IC). We give a compiler that runs N Byzantine agreement instances in parallel while preserving both guarantees, expected constant rounds and $O(f)$ rounds in the worst case. Parallel broadcast then follows with one extra round for $t < n/2$. To the best of our knowledge, this is the first compiler of its kind, and the first IC protocol with such expected and worse round-complexity.

Finally, by combining our protocol with standard (fast) sequential composition techniques, we obtain a valid candidate to execute *multi-party computation protocols* that require broadcast as their hybrid. By running existing MPC constructions using our broadcast protocol, we obtain the first MPC protocol that requires constant rounds of P2P communication in expectation and $O(f)$ rounds in the worst case.

*Research supported in part by Sunday Group, Inc. Vassilis Zikas was supported in part by JPMorgan Chase and the Stellar Development Foundation.

†School of Informatics, University of Edinburgh. E-mail: mciampi@ed.ac.uk.

‡School of Cybersecurity and Privacy, Georgia Institute of Technology. E-mail: {pforghani3,vzikas}@gatech.edu.

§Department of Computer Science, University of Victoria. E-mail: yunlu@uvic.ca.

Contents

1 Introduction **1**
- 1.1 Our contribution in detail . . . 2
- 1.2 Related works . . . 4
- 1.3 Technical overview . . . 5

2 Preliminaries and Model **8**
- 2.1 Primitives and Building Blocks . . . 10
  - 2.1.1 Cryptographic Building Blocks and Setups . . . 11
  - 2.1.2 Simulation-Based Security . . . 12

3 Randomized Early-Stopping BA **14**
- 3.1 Randomized Early-Stopping BA Compilers . . . 15
  - 3.1.1 Compiler without Setup . . . 17
  - 3.1.2 Compiler in Signature PKI Model . . . 22

4 Randomized Early-Stopping IC **29**
- 4.1 Transformation for Randomized Early-Stopping Parallel BA . . . 29
- 4.2 From Parallel BA to Interactive Consistency . . . 34

5 Randomized Early-Stopping MPC **35**
- 5.1 From Property-Based to Simulation-Based IC . . . 35
- 5.2 Reactive IC and Sequential Composition . . . 36
- 5.3 From Reactive IC to MPC over Point-to-Point Channels . . . 38

Bibliography **38**

A Constructions for Standard Building Blocks **43**

B A Binary-to-Multivalued Transformation for $t < \frac{n}{2}$ **44**

C Explicit Round Complexities **45**

D Related Work **50**

1 Introduction

Byzantine agreement (BA) and Byzantine broadcast (BB) are both core problems in cryptography and distributed computing. A BA protocol allows a set of n parties (each with their own input) to agree on a common output (Agreement property) while ensuring the output is an honest input when all honest inputs are the same (Validity property). In Byzantine broadcast, a designated sender distributes a value so that all honest parties output the same value, and they output the sender's value whenever the sender is honest. In cryptography, agreement and broadcast are widely used primitives with fundamental applications, with one of the most prominent being secure multiparty computation (MPC)[1] [Yao86, CDvdG88, GMW87, BGW88]. In particular, MPC protocols require parties to have access to a stronger form of broadcast channel, which allows each party to individually broadcast a different value at the same time (i.e., perform parallel broadcast). We refer to this primitive as *Interactive Consistency (IC)*.

Relying on this type of broadcast channels, has been proven to instrumental for the design round-efficient MPC protocols, and significant effort effort has been devoted to minimize the usage of such broadcast resources, leading to protocols that requires only a constant number of invocation to this special broadcast functionality [BMR90, DI05, CKKZ12, GIKR01, KOS03, HOSS18, EGPS22, AKP20, CDR+25, AKP23]. In particular, Damgård et al. [DMR+21] prove that in the computational setting a round-optimal MPC protocols requires at least one of the two rounds to be over broadcast. In [CDR+25] it is proven that for statistical security at least two, out of the four round needed, must be over broadcast. This means that the concrete round complexity of round-efficient MPC protocols is effectively dominated by the cost of realizing broadcast channels over peer-to-peer channels[2]. Hence, the round complexity of realizing broadcast (and IC in particular) protocols has significant impact on the overall round complexity of MPC protocols.

Unfortunately, it is impossible to implement Byzantine broadcast (and BA) via a deterministic protocol in a sublinear (in the number of corrupted parties) number of rounds. One way to circumvent this impossibility is to rely on randomized protocols to obtain constant round complexity in *expectation*. However, no guarantees may be given for when the protocol terminates in the worst case. To this end, one may design protocols that are *early-stopping*, i.e., all honest parties terminate in the worst case in rounds that depend on $f \leq t$, the number of actually corrupted parties. It is known that the optimal worst case round complexity is $\min(t+1, f+2)$ [DRS90] for any BA protocol, and thus we treat $O(f)$-round early stopping as *near optimal*.

We then ask the natural question of whether IC protocols can achieve the *best of both worlds*:

> *Can we design Interactive Consistency protocols with expected constant-round complexity, and $O(f)$ (with $f \leq t$) rounds in the worst case?*

We answer the above question positively. Our results are presented in the form of compilers. The first compiler lifts a wide class of expected constant-round BA protocols to protocols that *also* achieve early-stopping in addition to expected constant termination. Informally, we prove the following theorem.

Theorem (informal). *There exists a BA protocol information-theoretically secure against up to $t < n/3$ corruptions that, in expectation, requires constant rounds, and in the worst case terminates in $O(f)$ rounds.*

[1] Unless otherwise specified, MPC here has guaranteed output delivery.

[2] Unless otherwise specified, round complexity always refers to the peer-to-peer synchronous setting: one round corresponds to all parties sending one message over each peer-to-peer channel available.

We then use the above result as a component in to our second compiler, which composes N Byzantine agreement instances in parallel while preserving both guarantees, expected constant rounds and $O(f)$ rounds in the worst case, with both bounds independent of N. Parallel broadcast, and hence IC, follows with one extra round for $t < n/2$. This yields our main theorem:

Theorem (informal). *There exist Interactive Consistency protocols information-theoretically secure against up to $t < n/3$ corruptions that, in expectation, require constant rounds, and in the worst case terminate in $O(f)$ rounds.*

We show that our results can be extended to the $t < n/2$ setting assuming PKI. In the next section, we give a more detailed overview of all our results.

1.1 Our contribution in detail

Compiler for setup-free BA with expected constant, near-optimal early stopping. Given an expected constant round BA protocol π_{rand}, our compiler produces a BA protocol that terminates in expected constant rounds and has near-optimal early-stopping worst case round complexity ($O(f)$). The compiler preserves the adaptive-security and corruption threshold of the underlying protocol and introduces no additional assumptions. From a techniques perspective, our work introduces a new method to obtain early-stopping protocols in the information-theoretic setting that does not rely on exponential information gathering (EIG, e.g., [AD15]) or iteratively choosing a new leader, e.g., [LS22]. Our compiler requires π_{rand} to belong to a special class of BA protocols that we call *injection-tolerant* (more details in Section 1.3), and we show that the well known protocols presented in [FM97, KK06, KK09, Mic17] (e.g., Feldman–Micali, Katz–Koo) belong to this class. A slightly more formal version of one of our main theorems:

Informal description of Theorem 3.3. Let π_{rand} be an injection-tolerant expected constant round BA protocol (Def. 3.2)[3] secure against at most t corruptions. Then, our compiler Fig. 1 constructs an expected constant round, near-optimal early stoppping BA protocol against t corruptions assuming crusader agreement (Def. 2.1, which requires no signatures/setup when $t < n/3$).

First information theoretic, expected constant, early stopping BA for $t < n/3$. As corollary, when our compiler is instantiated with an expected-constant injection-tolerant information-theoretic BA protocol with $t < n/3$ resilience (Cor. 3.4, based on [FM97], and Cor. 3.5, based on [KK09]) then we obtain an information-theoretic protocol that tolerates $t < n/3$ corruptions, and terminates in expected constant rounds and near-optimal $O(f)$ rounds.

Compiler with additional PKI assumption. We construct a version of our BA compiler above, that additionally uses PKI. We instantiate this compiler with a expected constant round BA under under $t < n/3$, that assumes PKI [Mic17], achieving better round complexity (Cor. 3.7). We also instantiate our PKI-based compiler with a PKI-based expected constant round BA that is secure under $t < n/2$ [KK06][4] (Cor. 3.8). Notably, when compared with the SOTA for expected constant, early-stopping BA for $t < n/2$, ours protocol achieves better expected round complexity when compared with schemes that achieve similar guarantees.

For a concrete comparison, we refer to Table 1. This summarizes our instantiations alongside the randomized protocols of Elsheimy et al. [ELP24] and [EKL25]. The former is the closest point of comparison. It is likewise expected-constant-round and early-stopping under a PKI at $t < \frac{n}{2}$, with expected round complexity $(\frac{2}{p} + 2)(d + 9)$ and worst-case round complexity $(d + 9)(\lfloor f/d \rfloor + 2)$

[3] Described at a high level in Technical Overview (Section 1.3).

[4] We replace the OLE protocol in [KK06] with a faster, VRF-based protocol.

for a tunable constant d governing how many corrupted parties are detected (and how many extra rounds are spent) per iteration, where p is the agreement probability of the common coin their Theorem 2 assumes.

	Threshold	Setup/model	Expected	Worst case
Corollary 3.5	$t < n/3$	plain model	24	$14f + 31$
Corollary 3.7	$t < n/3$	signature PKI and ROM†	14	$4f + 10$
Corollary 3.8	$t < n/2$	signature and VRF PKI	37	$15f + 29$
[ELP24]	$t < n/2$	signature PKI and ideal common coin§	$6(d+9) \geq 60^\S$	$(d+9)(\lfloor f/d \rfloor + 2)^\ddagger$
[EKL25]	$t < (1-\alpha)\frac{n}{2}$	signature PKI, VRFs and ROM*	$6(d+11) \geq 72$	$(d+11)(\lfloor f/d \rfloor + 2)^\ddagger$

Table 1: Byzantine agreement protocols that are simultaneously expected-constant-round and early-stopping. Only the first two rows are unconditional; the others inherit the computational assumptions of their setup, so their guarantees hold except with negligible probability. †With unique signatures. $^\ddagger d \geq 1$ is a tunable integer. §The ideal coin of Theorem 2 of [ELP24] agrees with probability p, which enters the expected bound; the figure shown is at $p = \frac{1}{2}$. *For a constant $\alpha > 0$. The PKI also holds VRF keys, which corrupted parties may generate maliciously. The VRFs are modelled by the ideal functionality of [DGKR18], implementable in the random oracle model We instead for Corollary 3.8 require a stronger form of setup where VRF keys are all generated honestly, but in turn the corollary does not rely on the RO.

The two protocols have different trade-offs. Elsheimy et al. [ELP24] has the better worst-case bound, particularly when d is large. Our protocol instead has better expected round-complexity, and no choice of d closes that gap. At the most favorable coin agreement probability $p = \frac{1}{2}$, their protocol expected rounds is $(\frac{2}{p} + 2)(d + 9) = 6(d + 9) \geq 60$ for every $d \geq 1$, against our 37, and the minimum is moreover attained at $d = 1$, where the coefficient of f is at its worst. The protocol of [EKL25] builds on that of [ELP24] and targets communication efficiency. Each step is carried out by a small committee elected through VRFs, in the player-replaceable style, and the slightly smaller threshold $t < (1-\alpha)\frac{n}{2}$ is what keeps an honest majority in every committee of expected size λ with overwhelming probability. Its expected round complexity, $6(d+11) \geq 72$, is again above our 37.

Parallel BA and interactive consistency (IC). We give the first, to our knowledge, treatment of parallel BA and parallel BB (a.k.a., interactive consistency IC) with expected constant rounds and (near-optimal) early-stopping. Also in this case, our result is presented in the form of a compiler that, given an early-stopping, an expected constant, and an early-stopping-and-expected-constant protocol[5], produces an N-parallel Byzantine agreement (and thus broadcast for $t < n/2$) protocol for any N. The parallel protocol terminates in expected constant rounds while retaining $O(f)$ early stopping, both independent of N. The compiler preserves the assumptions and corruption threshold of the component protocols. The proof of Cor. 4.3 instantiates this compiler for $t < n/3$, and $t < n/2$ assuming PKI. Notably, without any amortisation [CDG+25], the expected round complexity remains on the same order as the SOTA for general parallel BB composition [BOEY03] even with the added benefit of early-stopping. We leave optimizations (e.g., by amortization) open to further investigation.

Informal description of Theorem 4.2. Let π_{rand} be an expected constant BA protocol where honest parties terminate within a constant number of rounds of each other, and terminate

[5]We require these input protocols to enjoy some special properties.

in constant rounds when honest parties unanimously hold the same input (Def. 4.1 for the precise requirement). Let π_{es} be a near-optimal early stopping BA protocol, and π_{res} be an expected constant, near-optimal early stopping BA protocol. Then our compiler (Fig. 3) returns an expected constant, near-optimal early stopping N-parallel (for any N) BA protocol, assuming Oblivious Leader Election (OLE). The combined protocol inherits the corruption threshold and assumptions of the component protocols.

We recall that OLE (Def. 2.2) can be achieved without setup for $t < n/3$ and with PKI for $t < n/2$). Table 2 compares our compiler for IC) with the other interactive-consistency compilers with expected constant round complexity.

	Threshold	Setup/model	Components	Expected	Worst case
Theorem 4.2	$t < \frac{n}{3}$ $t < \frac{n}{2}$	plain model signature PKI	expected-constant-round BA with δ-shift and unanimity convergence; early-stopping BA; expected-constant-round early-stopping BA; leader election	$O(1)$	$O(f)$
[BOEY03]	$t < \frac{n}{3}$	plain model	expected-constant-round BA with δ-shift; leader election	$O(1)$	$\times$
[CEG26]	$t \leq (1-\varepsilon)n$	signature PKI	expected-constant-round BB with δ-shift; leader election; Learn-or-Expose	$O(1)$	$\times$

Table 2: Expected-constant-round IC compilers, with the setup each one uses and the components it needs. Even with expected-constant-round early-stopping components, [BOEY03] and [CEG26] have no worst-case bound ($\times$), since each iteration elects a fresh random leader, which is corrupted with probability $\Omega(f/n)$ and can then make the iteration fail. The compiler of [CEG26] is proven against static adversaries only, with $\varepsilon > 0$ a constant.

Round optimal MPC. Our final objective is to replace broadcast invocations in MPC protocols with our IC protocol(s), thus obtaining an MPC protocol that is expected constant rounds, and $O(f)$-round in the worst case.

This raises two challenges; first, our IC protocols must be shown simulation-based secure, not merely property-based, and second, they must be shown sequentially composable, which is delicate because parties may terminate in different rounds. We settle both for static corruptions in Section 5. Below is a slightly more formal version of the theorem we prove.

Informal description of Cor. 5.6 Let g be a functionality computable with guaranteed output delivery by an MPC protocol that makes a constant number of calls to the broadcast functionality. Then g is also computable with guaranteed output delivery, against a static adversary corrupting $t < n/3$ parties (or corrupting $t < n/2$ parties when assuming a PKI) over point-to-point channels in expected constant number of rounds with worst-case round complexity $O(f)$.

1.2 Related works

The goals of designing protocols with expected constant round-complexity and early-stopping schemes have largely been pursued separately, with a few works that achieve both. We highlight the most relevant works here (discussing them just in terms of round-complexity) and present a more complete picture of related works in Sec D. For *expected constant round* protocols (without

early-stopping), Feldman-Micali [FM97] presented the first work for the optimal $t < n/3$, information theoretic setting, against a strongly adaptive adversary, and Katz-Koo [KK06] adapted their techniques to $t < n/2$ assuming PKI. Subsequent works improved either the concrete expected number of rounds [ADD+19] or brought the corruption threshold to dishonest majority [GKKO07, WXSD20, SLM+23, CEG26] for the problem of Byzantine broadcast (BB), which is not possible for BA. For *early stopping* protocols (without expected constant rounds), the first to achieve such properties were [TPS85, DRS90], with Berman-Garay-Perry [BGP92] the first to achieve the optimal $\min(t+1, f+2)$ early stopping in the information theoretic setting under $t < n/3$ with exponential message complexity, and [AD15] the first to achieve this with polynomial message complexity. Subsequent works also achieved near-optimal early stopping for $t < n/2$ [ELP25] and for $t < n$ (Byzantine broadcast) [DKLZ25]. A few works consider both round complexity notions. The first to do so [Zam96] achieves expected constant rounds and optimal early stopping for $t < n/8$ assuming a trusted dealer or otherwise $n > 80t+12$. In the honest-majority setting, [ELP24, EKL25] both achieve expected constant rounds and near-optimal early stopping assuming a PKI, the latter at the slightly sub-optimal threshold $t < (1-\alpha)n/2$ for a constant $\alpha > 0$. The PKI is central to their constructions, which rely on Correct-or-Detect Broadcast [FN09] and signed proofs of participation with no direct information-theoretic analogue; see the previous section for the detailed comparison with [ELP24, EKL25].

1.3 Technical overview

Expected constant round, early stopping BA. We construct compilers that takes an expected constant round BA π_{rand} (for $t < n/3$, or $t < n/2$ with PKI) and upgrades it to also satisfy near-optimal early stopping $O(f)$. Our compiler requires π_{rand} to belong to a special class of BA protocols that we call *injection-tolerant*. Informally, such protocol must satisfy 4 properties P1-P4 (we refer to (Def. 3.2) for the formal definitions of the properties). (P1) The protocol's rounds are organized into *phases*, where each phase has constant number of rounds, and where each party keeps track of a local/intermediate value (normally initialized with their input) that may change from one phase to another, until the party *commits* and thereby fixes its output, halting a constant number of rounds later. (P2) Commitment by all honest parties happens in constant number of phases if they unanimously hold the same local value, or (P3) within an expected constant number of phases, as when each phase leads to commitment with some constant probability p; in either case, after one honest party commits, all others do so within a constant number of phases (P4). The name reflects what the compiler does to π_{rand}. Our early-stopping mechanism overwrites, or *injects*, a party's local value with the default value ϕ at a phase boundary, and (P2)–(P4) are required to hold from any phase-start local values, so that such injections cannot derail the protocol.

Coming back to our compiler, the basic idea is to leverage property (P2): if all honest parties' inputs are the same, then (P2) guarantees commitment within a constant number of rounds. If *no* commitment happens within this time-frame, then it must be that not all honest parties have the same input (thus Validity does not apply). In this case, honest parties can replace their current local value with a fixed default value. If all honest parties do so, then (P2) will guarantee commitment in constant number of rounds. A challenge of this strategy is that since the protocol has eventual termination[6], a party can be unsure whether or not others have committed to terminate with a value: if someone has, we do not want to replace our local value with the default, otherwise (P4)'s quick commitment may not apply or worse, we may then commit to the wrong value. To prevent these problems, we add two rounds of so-called *crusader agreement (CA)* that lets parties tell each

[6]For simultaneous termination, a lower bound of $t+1$ rounds exists [DRS90].

other if they have committed to terminate with some value or not (via sending a ready or unready message). CA guarantees that every honest party either observes the same value as other honest parties, or detects the sender is corrupt (and thus can be immediately ignored). If an honest party sends a ready message then all honest parties continue running π_{rand} without injecting the default value, and are guaranteed to commit to the same output in a constant number of rounds by (P4). If on the other hand a corrupt party lies to an honest party about committing, then the honest party could detect and later ignore this party after noticing they themselves had not committed within the guaranteed (P4) number of rounds. This means that each malicious party can only delay termination by at most a constant number of rounds, before honest parties either commit by luck (P3) or by unanimously all taking on the default value as their local value (P2). The strategy is the same for $t < n/3$ and $t < n/2$; only the crusader agreement differs, needing no setup in the former and the standard PKI in the latter which is a standard assumption for BA when $t < n/2$.

Our compiler's modification to π_{rand} preserves the *expected constant round* nature of π_{rand}, except adding some constant rounds in expectation for calls to CA and to ensure that when (P2) or (P3) is triggered, we do not inject a value until parties have committed[7]. Since (P3) gives commitment within an expected constant number of phases from any local values, this holds even when some parties have replaced their local values with the default but others have not, provided no replacement falls inside the window over which (P3) delivers that commitment; we guarantee this by performing replacements only once every L phases, for a constant L that depends on π_{rand}. In addition, parties never replace their local value with the default once a party has actually committed, meaning (P4) will always apply for all parties to commit within constant rounds of each other.

Adaptive adversaries. If properties (P1) - (P4) (Def. 3.2) of π_{rand} and the guarantees of π_{pca} hold against a (strongly) adaptive adversary, then the compiled protocol is secure (with expected constant round and early stopping) against such an adversary as well. Informally, the compiled protocol deviates from π_{rand} only by uncommitted parties injecting a default local value, depending on whether they receive a ready message or not. If all honest parties start with the same input, (P2) makes them commit before any injection is allowed, satisfying Validity. If an honest party commits to an output value and remains honest through the first signaling round after its commitment, every honest party would commit soon after (by P4) to the same value, even if that party is corrupted afterwards, satisfying Agreement. Corrupting committers before they send ready does not break Agreement either. Such a corruption can leave an injection unblocked, but only at a phase preceding the first ready, and hence before the phases over which (P4) drives all honest parties to a common value. It therefore only changes the local values those phases start from, and (P4) makes no assumption on those values. For worst-case round complexity, an adversary can only prevent injection of the default value by falsely sending a ready message, and all honest parties disregard each lying party after the same constant number of phases. When all lying corrupt parties are disregarded, either some honest party has committed and (P4) applies, or all honest parties inject the default value and (P2) ensures *all* honest parties commit to the same default value output. For expected constant rounds, (P3) ensures that *all* honest parties commit soon after an expected constant number of blocks of L phases. Since (P3) holds against an adaptive adversary and asks *all* honest parties to commit, such an adversary cannot prevent expected constant round termination by corrupting some committers.

[7]When (P2) or (P3) is 'triggered' ((P2) by honest parties holding the same local value, or (P3) by luck, i.e., with probability $\geq p$), it takes some constant number of phases before honest parties commit. Thus, the protocol will ensure there are sufficient number of phases where we do not deviate from π_{rand} to let this commitment happen.

Parallel BA and Interactive Consistency (IC). We want to design a parallel BA/IC compiler that is both expected constant-round and early stopping. Parallel BA (Def. 2.4) refers to a protocol where parties are performing multiple BAs in parallel. Specifically, each party's input is a vector of N values, and output is an N-vector where the properties of BA (Validity and Agreement) holds individually for each of the N coordinates. For example, if all honest parties have the same value v for the first coordinate, then each honest party' output must be of the form $(v, \cdots)$ by Validity. Similarly, IC (Def. 2.5) refers to multiple BB's in parallel. Since BA implies BB for $t < n/2$, a parallel BA protocol implies an IC protocol in $t < n/2$ (and $t < n/3$).

A naive solution for parallel BA is to simply run multiple instances of a expected constant, early stopping BA in parallel. However, it is known that such a simple composition may achieve super-constant expected round complexity [BOEY03], as the composition's expected round complexity is the expectation of the *maximum* rounds of all the composed instances. Our parallel BA compiler follows ideas from Ben-Or-El-Yaniv [BOEY03], adding to it our early stopping property. At a high level, the compiler achieves expected constant rounds through a procedure that repeatedly runs the following steps:

1. Run in parallel m instances of the expected constant round protocol π_{rand} for each of the N coordinates, but only for a constant $r+\delta$ number of rounds. Parameters r, δ and m and chosen such that for *all* coordinates, it is likely at least *some* of the m instances would have terminated for someone with a value at round r and for all parties at round $r + \delta$. (Intuitively, if there are more coordinates, we need more instances so that at least one instance would terminate in a constant number of rounds; specifically, $m = \lceil \log(N) \rceil$.)

2. Detect if we are *very* lucky, i.e., an honest party observes that for every coordinate, all instances terminate, and all terminating instances for the same coordinate have the same output. This only requires a call to a single (non-parallel) BA protocol (if an honest party observes this very lucky event at round r, it means all honest parties must too at round $r + \delta$ due to the way δ is chosen in Step 1), and if we detect this lucky event, all honest parties can just terminate the parallel BA as there is no dispute in the output value.

3. Otherwise, we may still be somewhat lucky, i.e., for all coordinates, *some* instance(s) terminated for some honest party in this iteration. The issue is that, for the same coordinate, multiple instances may have terminated with different values, and moreover, some parties may have terminated whereas others may not, due to eventual termination of π_{rand}. To agree on the same vector as output, the parties send each other a candidate output vector, then pick a leader (via a constant-round *oblivious leader election OLE*, Def. 2.2) whose candidate the parties will try to use as output. After agreeing on the leader's candidate (a call to a single BA), parties check the candidate is indeed a legitimate output vector. Verification uses the property of π_{rand} that, if an honest leader claims an instance (in step 1) has terminated with a value at round r of that instance, all honest parties must also terminate in some constant δ rounds after r. Thus, an output vector from round r of the instances (from step 1, if such an output vector exists) is used as the candidate output vector in step 3, and the output at round $r + \delta$ is stored and used to verify the leader's claim.

4. Finally, the honest parties use a single BA call to agree on whether the leader's output is legitimate. If the BA outputs 1, they terminate with the leader's candidate vector; otherwise, they return to step 1.

Given each of the single-BA calls is expected $O(1)$ rounds and early stopping, each *iteration* of

the procedure has both properties too[8]. Then, since we set up each iteration to have a constant probability of successful termination (if some instance terminates for each coordinate and the chosen leader is honest, we terminate), the whole repeated procedure terminates in expected $O(1)$ rounds.

However, this does not mean the above repeated procedure is early stopping itself: e.g., in the worst case, we are never *very lucky* (in step 2) and always choose a bad leader who sends a bad candidate vector (in step 3). Thus, we augment the above with a parallel procedure that runs an early stopping BA protocol π_{es} for each of the coordinates, noting that if each π_{es} has some w worst-case round complexity, so does running multiple π_{es}'s in parallel. Once the π_{es} for a coordinate i terminates with a value v, parties will replace the input to the i-th coordinate π_{rand} calls with v (step 1), effectively 'injecting' the π_{es}-decided value to π_{rand}. Once all honest parties do so for all coordinates, it means that all honest parties have exactly the same input for all instances of π_{rand}. Thus, if π_{rand} guarantees (Def. 4.1)[9] constant-round termination if honest inputs are unanimous, all instances of the same coordinate would quickly output the same value (due to Validity) in step 1. This leads us to satisfy the 'very lucky' condition in step 2, allowing us to terminate in that step. In other words, once the early-stopping π_{es} protocols terminate, the repeated procedure terminates quickly thereafter, and our compiler also achieves early stopping.

As an aside, one may wonder why we need to run early stopping protocols π_{es} in addition to the expected constant procedure. Why not simply inject a default value when we fail to terminate in the first iteration and realise not all honest parties have the same input (like we do for the single/non-parallel BA)? This is because each iteration (steps 1 - 4 above) may fail to terminate, even if honest parties have unanimously the same input for certain coordinates, but no unanimous input for other coordinates. In this case, Validity still requires coordinates with unanimous honest input to output an honest party's value, thus making it incorrect to inject a default value.

Round-Optimal MPC. We are now only left to show how our IC protocols can replace broadcast hybrid-functionalities required in MPC protocols. As mentioned, we can prove that in the static security setting, property-based IC implies simulation-based IC. Intuitively, this holds because IC has no privacy requirement, so the ideal computation may hand the simulator all honest inputs up front, and with static corruptions the simulator can therefore run the honest parties' code on their true inputs from the first round, obtaining an internal execution *identically distributed* to a real one.

We need to prove that R *sequential* invocations of the IC protocol do not cause security issues. In particular, we need to prove that no honest party starts invocation $s+1$ before all have concluded invocation s, which, under probabilistic termination, can be problematic. A trivial solution would be to wait for the worst-case number of rounds, but this would defeat the entire premise of retaining expected constant termination for the MPC protocol. We instead apply the techniques of [LLR02, CCGZ16], keeping the parties within a constant slack, to make sure that they can synchronize. For more detail, we refer to Sec. 5.

2 Preliminaries and Model

Notation. Let κ denote the security parameter. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. A function $\mu : \mathbb{N} \to [0, 1]$ is *negligible* if $\mu(\kappa) < 1/p(\kappa)$ for every polynomial p and sufficiently large κ. We write

[8] Some nuance here (and in [BOEY03]) is that, since we invoke the single-BA calls sequentially, we have to take care of running BA calls *sequentially* when each call has eventual termination. We handle this similarly to previous work, and described more in Sec. 4.

[9] The requirements of Def. 4.1 are less stringent than properties (P1-P4) of Def. 3.2, e.g., not requiring the protocol to be phase-based.

PPT for probabilistic polynomial-time algorithms and use "$\approx_s$" and "$\approx_c$" to denote statistical and computational indistinguishability, respectively. For a distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denote sampling x from $\mathcal{D}$.

Communication model. We consider a synchronous network of n parties $P_1, \ldots, P_n$. Parties communicate over a complete network of pairwise secure point-to-point channels. These channels provide both privacy and authenticity.

Communication proceeds in rounds. In each round, parties first send their messages and then receive all messages sent to them in that round. Every message sent in round r is delivered before the beginning of round $r + 1$.

Adversarial model. We consider t-adversaries, who may corrupt up to t parties. A *rushing* adversary may, in each round, see the messages sent by honest parties before choosing the messages sent by corrupted parties in that same round. A *malicious* adversary may make corrupted parties deviate arbitrarily from the prescribed protocol. A *static* adversary fixes the set of corrupted parties before the execution begins, whereas an *adaptive* adversary may corrupt parties during the execution, subject to the bound t, and learns the internal state of a party upon corrupting it. A *strongly adaptive* adversary may, in addition, replace or drop the messages that a party sends in the round in which it is corrupted.

Unless otherwise specified, all our compilers are analyzed against rushing malicious adversaries and they match the corruption model of the building blocks they are instantiated with, so that instantiating a compiler with adaptively secure components yields an adaptively secure protocol, and likewise for strongly adaptively and statically secure ones. Underlying protocols and building blocks may also be assumed secure against weaker adversaries, such as non-rushing or semi-honest ones; in such cases the security of the resulting construction is interpreted with respect to the corresponding adversarial guarantees, and the exact assumptions are stated in the relevant theorem.

Property-based security. Most of our results use property-based security definitions. Each primitive is specified by explicit properties such as termination, agreement, and validity. These specifications are independent of the security flavor. We use the following convention throughout. A protocol satisfies a property-based definition

- *perfectly* if, against every possibly computationally unbounded adversary in the stated adversary class, the probability that an execution violates any qualitative defining property is zero, for every κ;
- *statistically* if, against every possibly computationally unbounded adversary in the stated adversary class, that probability is negligible in κ; and
- *computationally* if, against every PPT adversary in the stated adversary class, that probability is negligible in κ.

In every case the guarantee is required for every admissible input vector and initial adversarial state, and the probability is over the randomness of the parties, the adversary, and the setup, if any. We say that a protocol is a perfectly, statistically, or computationally secure implementation of a primitive according to which of the above conditions it satisfies. We use *information-theoretic security* to encompass the perfect and statistical flavors and specify which flavor is meant whenever the distinction matters. Our concrete constructions use perfect or computational security; we include the statistical flavor so that the security preserved by our generic transformations can be stated uniformly.

Some properties carry an intrinsic quantitative guarantee. In particular, a property requiring success probability at least p—such as p-fairness—requires probability at least p in the perfect setting and at least $p - \mathsf{negl}(\kappa)$ in the statistical and computational settings. Expected and worst-case round complexities are separate performance guarantees and are not altered by the security flavor unless explicitly stated.

2.1 Primitives and Building Blocks

Crusader agreement was introduced by Dolev [Dol82] as the relaxation of Byzantine broadcast in which the honest receivers that do not explicitly detect a faulty sender must agree on a common value, while those that do detect one may output a distinguished symbol. We use the n-sender parallel variant, in which every party acts as a sender once and all n instances run concurrently.

Definition 2.1 (Parallel Crusader Agreement)**.** *A protocol π_{pca} with message space $\mathcal{M}$ is a parallel crusader agreement protocol if, in every invocation, each party P_j has input $m_j \in \mathcal{M}$, each honest party P_i outputs a vector $\boldsymbol{v}_i = (v_{i,1}, \ldots, v_{i,n}) \in (\mathcal{M} \cup \{\perp\})^n$, and the following properties hold.*

- ***Termination.*** *Every honest party outputs.*
- ***Honest-sender delivery.*** *For every honest sender P_j with input m_j, every honest party P_i outputs $v_{i,j} = m_j$.*
- ***Crusader consistency.*** *For every sender index $j \in [n]$, if two honest parties P_i and $P_{i'}$ output non-$\perp$ values in coordinate j, namely $v_{i,j} \neq \perp$ and $v_{i',j} \neq \perp$, then $v_{i,j} = v_{i',j}$.*

The protocol π_{pca} we use for Parallel Crusader Agreement is a two-round protocol that uses no setup and tolerates $t < \frac{n}{3}$ corruptions. It is given, with its proof of correctness, in Appendix A (Proposition A.1).

The threshold is tight in the setup-free setting: Dolev shows that crusader agreement is unachievable once $t \geq \frac{n}{3}$ [Dol82, Lemma 4]. With a signature PKI the bound no longer applies, which is what the compiler of Section 3 with resilience $t < \frac{n}{2}$ exploits.

A leader-election protocol lets the parties choose a common leader, which is honest with constant probability.

Definition 2.2 (Oblivious Leader Election)**.** *An n-party protocol π_{ole} is a leader-election protocol with fairness p if the following conditions are satisfied:*

- ***Termination.*** *There exists an a-priori-known round R_{ole} such that every honest party terminates within R_{ole} rounds and outputs a leader identifier $\ell_i \in [n]$.*
- ***p-Fairness.*** *With probability at least p, there exists a party P_ℓ that is honest at the start of the leader-election invocation such that every honest party outputs $\ell_i = \ell$.*

Definition 2.3 (Byzantine Agreement)**.** *An n-party protocol π_{ba} with domain $\mathcal{M}$ is a Byzantine agreement protocol if each party P_i has input $x_i \in \mathcal{M}$, each honest party P_i outputs a value $o_i \in \mathcal{M}$, and the following properties hold.*

- ***Termination.*** *Every honest party eventually outputs.*
- ***Agreement.*** *For every two honest parties $P_i, P_{i'}$ that output, $o_i = o_{i'}$.*
- ***Validity.*** *If all honest parties input $v \in \mathcal{M}$, every honest output is $o_i = v$.*

Interactive consistency (IC, Def. 2.5) is the related task of N-parallel Byzantine broadcast (BB). We work internally with N-parallel BA, solving N BA instances at once. This implies N-parallel BB, and thus IC, for $t < n/2$.

Definition 2.4 (N-Parallel Byzantine Agreement)**.** *An n-party protocol π with domain $\mathcal{M}$ is an N-parallel Byzantine agreement protocol if each party P_i has input $(v_{i,1}, \ldots, v_{i,N}) \in \mathcal{M}^N$, each honest party P_i outputs a vector $(o_{i,1}, \ldots, o_{i,N}) \in \mathcal{M}^N$, and, for every coordinate $j \in [N]$, the following properties hold.*

- ***Termination.*** *Every honest party eventually outputs.*
- ***Agreement.*** *For every two honest parties $P_i, P_{i'}$ that output, $o_{i,j} = o_{i',j}$.*
- ***Validity.*** *If every honest party has the same input $v \in \mathcal{M}$ in coordinate j (i.e., $v_{i,j} = v$ for every honest P_i), then every honest output is $o_{i,j} = v$.*

Definition 2.5 (Interactive Consistency (IC))**.** *An n-party protocol π with message space $\mathcal{M}$ is an interactive consistency (equivalently, parallel broadcast) protocol if each party P_j has input $m_j \in \mathcal{M}$, each honest party P_i outputs a vector $(o_{i,1}, \ldots, o_{i,n}) \in (\mathcal{M} \cup \{\perp\})^n$, and the following properties hold.*

- ***Termination.*** *Every honest party eventually outputs.*
- ***Agreement.*** *For all honest $P_i, P_{i'}$ that output and all $j \in [n]$, $o_{i,j} = o_{i',j}$.*
- ***Validity.*** *For every honest sender P_j with input m_j, every honest party P_i that outputs has $o_{i,j} = m_j$.*

2.1.1 Cryptographic Building Blocks and Setups

Our cryptographic constructions require signatures, verifiable random functions, and their corresponding setups which are defined below.

Digital signatures. We rely on a secure digital signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$. On input 1^κ, the key-generation algorithm Gen outputs a signing key and verification key $(\mathsf{sk}, \mathsf{pk})$. Given a message m, the signing algorithm $\mathsf{Sign}_{\mathsf{sk}}(m)$ outputs a signature σ, while the verification algorithm $\mathsf{Ver}_{\mathsf{pk}}(m, \sigma)$ outputs 1 if σ is a valid signature on m with respect to pk, and outputs 0 otherwise.

We assume that Σ is existentially unforgeable under chosen-message attacks (EUF-CMA); i.e., no PPT adversary can produce, with non-negligible probability, a valid signature on a fresh message, even after adaptively obtaining signatures on messages of its choice.

Signature public-key infrastructure. Some of our feasibility results assume a public-key infrastructure for signatures (i.e. are in *Signature PKI model*). Each party has a registered verification key pk_i that is available to all protocol participants. For every honest party P_i, the key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ is generated honestly, and the signing key sk_i is known only to P_i. For corrupted parties, the adversary may choose and register their signature verification keys after observing the public keys of all honest parties. This last relaxation is specific to the signature PKI; the VRF-PKI defined next does not grant it.

Verifiable random functions. A verifiable random function is a triple $\mathsf{VRF} = (\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Ver})$. On input 1^κ, $\mathsf{VRF.Gen}$ outputs a key pair $(\mathsf{vsk}, \mathsf{vpk})$; on input a seed $x \in \{0,1\}^*$, $\mathsf{VRF.Eval}_{\mathsf{vsk}}(x)$ outputs a value $y \in \mathcal{Y}$ together with a proof ω; and $\mathsf{VRF.Ver}_{\mathsf{vpk}}(x, y, \omega)$ outputs 1 if ω certifies that y is the value of the function keyed by vpk at x. We require the standard properties:

- **Complete provability.** For honestly generated keys, an honestly computed (y, ω) always verifies.
- **Unique provability.** For every — possibly maliciously generated — vpk and every seed x, there is at most one y for which some proof verifies.

– **Residual pseudorandomness.** For keys output by $\mathsf{VRF.Gen}$, the value at any seed not yet queried is computationally indistinguishable from uniform on $\mathcal{Y}$, even given the values and proofs at seeds of the adversary's choosing.

We say the range is of *exponential size* if $|\mathcal{Y}| = 2^{\Omega(\kappa)}$, so that the probability that any two of the n parties obtain the same value at a given seed is negligible; ties are broken by party index in any case.

VRF public-key infrastructure. In the *VRF PKI model* the setup samples $(\mathsf{vsk}_i, \mathsf{vpk}_i) \leftarrow \mathsf{VRF.Gen}(1^\kappa)$ independently for every $i \in [n]$, publishes $(\mathsf{vpk}_1, \ldots, \mathsf{vpk}_n)$, and hands vsk_i to P_i. A corrupted party learns its own vsk_i, but neither chooses it nor selects among candidate key pairs. This is a strictly stronger setup assumption than the signature PKI above, and the strengthening is necessary rather than incidental: our leader election (Section 3) elects the party whose VRF value at a public seed is smallest, and if the adversary could register verification keys of its own choosing after seeing the seed, it could sample many candidate key pairs offline and register the one minimizing its value, biasing the election arbitrarily. Residual pseudorandomness is stated for honestly generated keys and gives no guarantee about the distribution of values under adversarially chosen ones.

2.1.2 Simulation-Based Security

Simulation-based security. Our constructions and their analyses are property-based. The application to MPC in Section 5, however, uses secure function evaluation and composes several invocations of interactive consistency inside a larger protocol, so it is naturally argued in the simulation paradigm. We therefore recall the ideal computation for secure function evaluation and translate our property-based interactive-consistency results into simulation-based ones. For simplicity, and following [Can00], we work in the stand-alone setting with static corruptions, in which the corrupted set is fixed before the protocol execution begins.

We parameterize the formulation by a security flavor $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. For two distribution ensembles X and Y, the relation $X \approx_\star Y$ means that X and Y are identically distributed when $\star = \mathsf{perf}$, that $X \approx_s Y$ when $\star = \mathsf{stat}$, and that $X \approx_c Y$ when $\star = \mathsf{comp}$. In the perfect and statistical cases the real-world adversary may be computationally unbounded, and the simulator is allowed running time polynomial in that of the adversary. In the computational case both the adversary and the simulator are PPT.

Real-world execution. An n-party protocol $\pi = (P_1, \ldots, P_n)$ is an n-tuple of interactive PPT Turing machines. Each party P_i starts with an input $x_i \in \{0,1\}^*$ and uniformly distributed random coins; without loss of generality, the input length is bounded by the security parameter κ. An adversary $\mathcal{A}$ is another interactive machine describing the behavior of the corrupted parties and belongs to the computational class specified by the security flavor above. It starts the execution with the identities of the corrupted parties, their private inputs, and an auxiliary input. The parties execute the protocol in the synchronous, rushing model described above: honest parties follow the protocol, whereas corrupted parties act according to $\mathcal{A}$ and may deviate arbitrarily. At the conclusion of the execution, each honest party outputs its prescribed output, while the adversary outputs its view of the computation.

Definition 2.6 (Real-world execution)**.** *Let $\pi = (P_1, \ldots, P_n)$ be an n-party protocol and let $I \subseteq [n]$ denote the set of parties corrupted by $\mathcal{A}$. The real-world execution of π with adversary $\mathcal{A}$ on input*

vector $\boldsymbol{x} = (x_1, \ldots, x_n)$*, auxiliary input* z*, and security parameter* κ*, denoted by* $\text{REAL}_{\pi,I,\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$*, is the joint output distribution of* $P_1, \ldots, P_n$ *and* $\mathcal{A}(z)$ *resulting from the above protocol interaction.*

Ideal computation for secure function evaluation. We use the standard ideal computation with guaranteed output delivery. Let $g : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function, and fix a set $I \subseteq [n]$ of corrupted parties with $|I| \leq t$, chosen before the execution begins. On input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z, and security parameter κ, the ideal computation proceeds as follows.

- **Sending inputs to the trusted party.** Each honest party P_i, for $i \notin I$, sends its input $x_i \in \{0,1\}^*$ to the trusted party. The simulator $\mathcal{S}$ sends an arbitrary input $x'_i \in \{0,1\}^*$ on behalf of each corrupted party P_i, $i \in I$. Let $x'_i := x_i$ for every $i \notin I$.
- **Trusted party computes the outputs.** The trusted party computes $(y_1, \ldots, y_n) := g(x'_1, \ldots, x'_n)$, sends y_i to $\mathcal{S}$ for every $i \in I$, and sends y_i to P_i for every $i \notin I$.
- **Outputs.** Each honest party outputs the value received from the trusted party, and the simulator outputs an arbitrary function of its view.

We denote the resulting joint output distribution by $\text{IDEAL}^{g}_{I,\mathcal{S}(z)}(\boldsymbol{x}, \kappa)$.

Definition 2.7 (Secure function evaluation)**.** *Fix* $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$*. A protocol* π *is* $\star$*-securely computes* g *with guaranteed output delivery against static* t*-adversaries if, for every real-world adversary* $\mathcal{A}$ *in the class associated with* $\star$*, there exists an ideal-world simulator* $\mathcal{S}$ *in the corresponding class such that, for every* $I \subseteq [n]$ *of size at most* t*,*

$$\left\{\text{REAL}_{\pi,I,\mathcal{A}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\, \kappa\in\mathbb{N}} \approx_\star \left\{\text{IDEAL}^{g}_{I,\mathcal{S}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\, \kappa\in\mathbb{N}}.$$

Equivalently, we call π *a perfectly, statistically, or computationally* t*-secure protocol for* g *according as* $\star = \mathsf{perf}$*,* stat*, or* comp*.*

Ideal computation for interactive consistency. All our agreement primitives guarantee output delivery, so we define a single ideal computation with no abort options for the adversary. Interactive consistency has no privacy requirement, and accordingly the trusted party hands the simulator all honest inputs. The only power the ideal-model adversary retains is to choose the messages of the corrupted senders, which is precisely the slack that the validity property of Definition 2.5 leaves to a real-world adversary. Fix a set $I \subseteq [n]$ of corrupted parties with $|I| \leq t$, chosen before the execution begins.

- **Sending inputs to the trusted party.** Each honest party P_i, for $i \notin I$, sends its input $m_i \in \mathcal{M}$ to the trusted party, who forwards m_i to the simulator $\mathcal{S}$. The simulator sends arbitrary inputs $m'_i \in \mathcal{M} \cup \{\perp\}$ on behalf of the corrupted parties P_i, $i \in I$.
- **Trusted party answers parties.** The trusted party sets $y_j := m_j$ for every $j \notin I$ and $y_j := m'_j$ for every $j \in I$, and sends $\boldsymbol{y} = (y_1, \ldots, y_n)$ to P_i for every $i \notin I$.
- **Outputs.** Each honest party outputs the vector it received from the trusted party, and the simulator outputs an arbitrary function of its view.

We denote this ideal computation by $\text{IDEAL}^{\mathsf{ic}}_{I,\mathcal{S}(z)}(\boldsymbol{m}, \kappa)$.

Definition 2.8 (Simulation-based security)**.** *Fix* $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$*. A protocol* π *is* $\star$*-securely realizes the ideal interactive-consistency computation against static* t*-adversaries if, for every real-world adversary* $\mathcal{A}$ *in the class associated with* $\star$*, there exists an ideal-world simulator* $\mathcal{S}$ *in the corresponding class such that, for every* $I \subseteq [n]$ *of size at most* t*,*

$$\left\{\text{REAL}_{\pi,I,\mathcal{A}(z)}(\boldsymbol{m}, \kappa)\right\}_{(\boldsymbol{m},z)\in(\{0,1\}^*)^{n+1},\, \kappa\in\mathbb{N}} \approx_\star \left\{\text{IDEAL}^{\mathsf{ic}}_{I,\mathcal{S}(z)}(\boldsymbol{m}, \kappa)\right\}_{(\boldsymbol{m},z)\in(\{0,1\}^*)^{n+1},\, \kappa\in\mathbb{N}}.$$

Equivalently, we call π a perfectly, statistically, or computationally t-secure interactive-consistency protocol according as $\star = \mathsf{perf}$, stat, or comp.

The honest senders' coordinates are fixed to their true inputs, matching the validity clause of Definition 2.5, while the corrupted senders' coordinates are chosen freely by $\mathcal{S}$. For either simulation-based definition, if π assumes a setup, both worlds are understood to be augmented with the corresponding setup sampled from its prescribed distribution. In the ideal world the simulator emulates the registration phase, sampling what the setup samples and letting the adversary register what the adversary registers. In particular, the signature PKI lets the adversary register the corrupted parties' verification keys, whereas the VRF PKI does not.

Reactive functionality for interactive consistency. An MPC protocol in the broadcast model does not consume a single invocation of interactive consistency, but R *sequential* ones, with the inputs to invocation s computed from the outputs of invocation $s-1$. We therefore also define a reactive functionality supporting sequential invocations, in which the trusted party maintains state across invocations, following [CL17, CCF+26]. This approach both permits reuse of the same setup across the sessions and separates the treatment of probabilistic termination from the final replacement of the broadcast channel, which becomes a single application of the composition theorem of [Can00].

Each invocation proceeds exactly as in the ideal computation above: each honest P_i sends its s-th input $m_i^s \in \mathcal{M}$, which the trusted party forwards to $\mathcal{S}$; the simulator sends inputs $m_j'^s$ on behalf of the corrupted senders P_j, $j \in I$; and the trusted party delivers $\boldsymbol{y}^s$, where $y_j^s = m_j^s$ for $j \notin I$ and $y_j^s = m_j'^s$ for $j \in I$. Invocation $s+1$ begins only after every honest party has received $\boldsymbol{y}^s$. We denote this ideal computation by $\textsc{ideal}^{\mathsf{ic}\text{-}R}_{I,\mathcal{S}(z)}(\boldsymbol{m}, \kappa)$. For each security flavor $\star$, secure realization of reactive interactive consistency is defined as in Definition 2.8, replacing $\textsc{ideal}^{\mathsf{ic}}$ by $\textsc{ideal}^{\mathsf{ic}\text{-}R}$. Since our agreement primitives guarantee output delivery, no state in fact propagates between invocations and the trusted party evaluates each one from scratch—in contrast to an aborting functionality, which must remember an abort and return $\perp$ in every later invocation. When instantiating the reactive functionality, the protocol may reuse its existing setup across the sequential invocations.

3 Randomized Early-Stopping BA

Goldreich and Petrank [GP90] consider the problem of eliminating the possibility of non-terminating executions in fast randomized Byzantine agreement protocols, such as Feldman-Micali [FM97], which achieve constant expected running time but may have (rare) infinite executions. Their goal is to obtain a protocol that preserves the constant expected complexity of the randomized protocol while guaranteeing termination in all executions. To this end, they assume a randomized BA protocol that proceeds in phases and satisfies two key properties: (i) once an honest party decides in a phase, all honest parties are aligned on that value by the end of the phase, and (ii) if all honest parties enter a phase with the same value, they all decide during that phase. It is worth noting that not all randomized BA protocols satisfy these properties; for example, the protocol of [KK06] does not. Thus, to capture a broader class of protocols, the required properties must be made more permissive. In particular, rather than requiring termination in the same phase in which the relevant event occurs, one should allow termination to occur within a bounded number of subsequent phases. Aside from this limitation, we show below that the compiler cannot be used to obtain a protocol that simultaneously achieves constant expected round complexity and $O(f)$ worst-case round complexity, where f denotes the actual number of corruptions, which is precisely the combination of guarantees targeted in this work.

A first attempt at achieving an expected constant round, early stopping protocol is to start from the compiler of Goldreich and Petrank [GP90]. Their compiler combines a fast randomized protocol with any deterministic BA protocol. The parties execute the randomized protocol for a fixed number of phases k. If agreement is reached during these phases, the protocol terminates as usual. Otherwise, after k phases, all parties invoke a deterministic BA protocol using their current values as inputs. This yields a protocol with expected running time close to that of the randomized protocol, while guaranteeing termination within k phases plus the cost of the deterministic protocol. This yields a protocol with constant expected round complexity and worst-case round complexity $t + O(\log t)$, where $t \in O(n)$.

However, we show that even if their deterministic fallback is replaced by an early-stopping $O(\min(f+2, t+1))$-round protocol, e.g., [AD15], there is no choice of the parameter k that yields a protocol with both constant expected round complexity and $O(f)$ worst-case round complexity when $t = \Theta(n)$.

Lemma 3.1. *Fix any security flavor $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. Consider the Goldreich–Petrank compiler with its deterministic fallback replaced by a $\star$-secure early-stopping Byzantine agreement protocol of round complexity $O(\min\{f+2, t+1\})$, and suppose the compiled randomized protocol is $\star$-secure against up to $t = \Theta(n)$ corruptions. Then no choice of k yields both constant expected round complexity and worst-case round complexity $O(f)$.*

Proof. We distinguish two cases.

Case 1: k is constant. This is where the hypothesis $t = \Theta(n)$ is used. Karlin and Yao [KY84], and independently Chor, Merritt and Shmoys [CMS89], show that when the number of corruptions is linear in the number of parties, any ρ-round randomized agreement protocol fails to reach agreement with probability at least $(c\rho)^{-\rho}$ for a constant c. A phase consists of a constant number of rounds, so k phases occupy $\rho = O(k)$ rounds, and for constant k that failure probability is a constant $q > 0$ independent of n. With probability at least q, therefore, the randomized protocol has not terminated within the k phases of the prefix and the fallback is invoked. In executions with $f = t$ the fallback may take $\Theta(t)$ rounds, so the expected round complexity is at least $q \cdot \Theta(t) = \Omega(t)$, which by $t = \Theta(n)$ is not constant.

The hypothesis is what makes this quantitative rather than merely plausible. Constant expected round complexity on its own does not lower-bound the probability of a long execution (a protocol that always halts in one phase has constant expectation and never overruns) and without $t = \Theta(n)$ the probability q of overrunning could be small enough that $q \cdot t$ stays bounded, leaving the expectation constant after all.

Case 2: k is not constant. Then the protocol spends $\Omega(k)$ rounds in the randomized prefix before invoking the fallback. For executions with constant f, the fallback takes $O(1)$ rounds, but the total running time is still $\Omega(k)$, which is not $O(f)$. Hence, the protocol is not early-stopping.

Combining the two cases, no choice of k achieves both properties. □

3.1 Randomized Early-Stopping BA Compilers

Both compilers below apply to randomized BA protocols with the following properties, which many existing protocols satisfy, including [FM97, KK06, KK09, Mic17].

Definition 3.2 (Injection-tolerant BA)**.** *Consider a phase-based randomized Byzantine agreement protocol π_{rand} in which each party maintains a local value, a monotone* commit *flag, and a local output. An* injection *overwrites a party's local value at a phase boundary, and a window of consecutive phases is* undisturbed *if no injection takes place at a boundary interior to it. The local*

values at the first phase of a window are always unrestricted. We call π_{rand} injection-tolerant*, with constants* $c, p, \Delta_0, \Delta_1, \Delta_2, \Delta_3$*, if it satisfies the following properties.*

- ***(P1) Phase structure.*** *Each phase consists of a constant* c *number of rounds. Each party maintains a local value* b_i *and a monotone, locally computable flag* $\mathsf{committed}_i$*; upon setting* $\mathsf{committed}_i$*, a party fixes its local output, continues to execute* π_{rand} *for a further constant* Δ_0 *number of its rounds, and then halts. An injection into a party that has already set* $\mathsf{committed}_i$ *has no effect. The* c *rounds are those in which* π_{rand} *carries out the phase itself;* π_{rand} *may in addition run sub-protocols in the background, spanning the compiler's signaling rounds and earlier phases, provided they never read the local value and their schedule is a fixed function of the round index.*
- ***(P2) Unanimity convergence.*** *If all honest parties start a phase with the same local value* v *(independent of previous phases) and no honest party has committed, then they keep their local values equal to* v *thereafter, and all honest parties commit to* v *by the end of the phase that is a constant* Δ_1 *phases later.*
- ***(P3) Expected commitment.*** *There exists a constant* $\Delta_2 \geq 0$ *such that, for any number* m *of disjoint windows of* $\Delta_2 + 1$ *phases fixed in advance and left undisturbed, with* m *polynomial in* κ *when security is computational, the expected index of the first window that begins with some honest party committed or ends with all honest parties committed to the same value, counted as* m *if there is none, is at most* $1/p$ *for a constant value* p*.*[10]
- ***(P4) Commit propagation.*** *From any phase-start local values at a phase in which no honest party has committed, once some honest party commits, all remaining honest parties commit to the same value within a constant* Δ_3 *additional phases, provided that window is undisturbed.*

Property (P3) holds in particular when each window that begins with no honest party committed ends with all honest parties committed to the same value with probability at least p given the history up to its start, since the index of the first window that does so is then dominated by a geometric random variable with parameter p. We state it over a sequence of windows because such a bound given the history can fail when the windows share a setup, as with a leader election that uses one VRF key per party across all phases.

Setting $\Delta_2 = 0$ in Property (P3) recovers the literal reading that all honest parties commit within the very phase in which the underlying randomness is used, in which case the compiler may inject at every phase boundary; this is the case, e.g., for the Feldman–Micali protocol [FM97] instantiated below. Protocols whose confirmation step is only completed in a later phase (e.g., the protocol of [Mic17] under its natural presentation, or the protocol of [KK06] when a single iteration is taken to be a phase) are captured by taking $\Delta_2 > 0$ and injecting only once every $L > \Delta_2$ phases. The two instantiations below take the two available routes: rotating the round sequence of [Mic17] brings its confirmation inside a single phase, giving $\Delta_2 = 0$ and $L = 1$, whereas for [KK06] no rotation does so and we keep its own iteration as the phase, with $\Delta_2 = 1$ and $L = 2$. Similarly, $\Delta_0 = 0$ recovers the reading in which a party halts as soon as it commits, whereas $\Delta_0 > 0$ accommodates protocols, such as that of [KK06], whose output value is irrevocably fixed a constant number of rounds before the party may stop participating.

Phase alignment is a modeling choice. Property (P1) fixes only the length of a phase (c rounds), not where a phase begins. Given a candidate protocol π_{rand} whose behavior repeats with

[10] Against an adaptive adversary, k guaranteed commitments let it void up to t/k windows by corrupting the committers before they signal ready: $\Theta(f)$ extra blocks for $k = 1$, a constant number when k is a constant fraction of t, none for $k = t + 1$. We ask for all honest parties because existing protocols provide it, with better constants.

period c, a phase is any block of c consecutive rounds, and any fixed cyclic rotation of its round sequence may be used to choose which round opens it, provided the same rotation is used throughout; the execution begins at that round, so no round falls outside a phase. Different rotations can yield different – and generally incomparable – constants $p, \Delta_0, \Delta_1, \Delta_2, \Delta_3$; in particular, starting a phase immediately after the protocol's own randomized step, rather than wherever its original presentation happens to place that step, can collapse Δ_2 to 0 even for a protocol whose native presentation does not decide-and-confirm within a single stated phase. We exploit this when instantiating π_{rand} with the protocol of [Mic17] in Corollary 3.7 below, where Properties (P1)–(P4) are proved directly for the rotated start. A few rounds may still precede the first phase when a sub-protocol runs in the background, as in Corollaries 3.5 and C.1; that prefix does not come from the alignment and is charged on top.

3.1.1 Compiler without Setup

The compiler $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{pca}})$ follows the approach described in Section 1.3. Parties run π_{rand} phase by phase and, once every L phases, signal through π_{pca} whether they have committed. From phase $\Delta_1 + 1$ on, a party that has not committed and trusts no ready signal resets its local value to a fixed default value ϕ, and a party whose ready is not followed by commitment within Δ_3 phases is marked faulty and ignored.

Compiler $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{pca}})$

Below is the code for party P_i.
Parameters: the constants $\Delta_0, \Delta_1, \Delta_3$ of Properties (P1)–(P4), an injection period $L > \Delta_2$, and a fixed default value ϕ of the agreement domain.

On input $(\texttt{input}, x)$, initialize $b := x$, $output := \perp$, $S := \emptyset$, $BAD := \emptyset$, and $w_j := 0$ for all $j \in [n]$.
Adjusted execution of the randomized π_{rand}
Repeat the following for phase $r = 1, 2, \ldots$:

1. Execute one full phase of π_{rand} starting from local value b, continuing from the previous internal state except for the local value b which may be externally updated, and let b denote the updated local value. If π_{rand} has set its commit flag, let $output$ denote its local output.
2. If $r \not\equiv 0 \pmod{L}$, skip this step and the next, proceeding directly to the output step: in a phase at whose end no injection can occur, neither the signaling round nor the bookkeeping it feeds is executed. Otherwise, if $output \neq \perp$, send input ready to π_{pca}; and if $output = \perp$, update $w_j := w_j + 1$ for every party $P_j \in S$ and send input unready to π_{pca}.
3. After receiving output $y = (y^1, \ldots, y^n)$ from π_{pca}:
   - For every party P_j such that $y^j = \perp$, update $BAD := BAD \cup \{P_j\}$;
   - For every party $P_j \notin BAD$ such that $y^j = \mathsf{ready}$, update $S := S \cup \{P_j\}$;
   - For every party $P_j \in S$ such that $w_j \geq \lceil \Delta_3 / L \rceil$, update $BAD := BAD \cup \{P_j\}$ and $S := S \setminus \{P_j\}$;
   - If $r > \Delta_1$, $output = \perp$, and $S = \emptyset$, then reset the local value to the default value $b := \phi$.
4. If $output \neq \perp$ and $(\texttt{output}, output)$ has not been output already, output $(\texttt{output}, output)$. Halt at the end of the first phase at (or after) the one in which $output$ was set whose end carries a signaling round, or once Δ_0 further rounds of π_{rand} have been executed, whichever is later; otherwise continue.

Figure 1: Randomized early-stopping BA compiler without any setup

Theorem 3.3. *Fix a security flavor $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. Let π_{rand} be a $\star$-secure injection-tolerant randomized Byzantine agreement protocol (Definition 3.2), and let π_{pca} be a $\star$-secure parallel crusader agreement protocol against the same adversary class, tolerating fewer than t_{rand} and*

t_{pca} *corruptions, respectively. Let* $L > \Delta_2$. *Then* $\pi = \mathsf{Comp}^{\mathsf{No\text{-}Setup}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{pca}})$ *is a* $\star$*-secure Byzantine agreement protocol against up to* t *corruptions and the same adversary class, for every* $t < \min\{t_{\mathsf{rand}}, t_{\mathsf{pca}}\}$. *Its expected round complexity is*

$$\left(c + \tfrac{c_{\mathsf{pca}}}{L}\right)\left(\frac{L}{p} + L\left(\lceil \tfrac{\Delta_3}{L} \rceil - 1\right)\right) + \Delta_0^*,$$

and its worst-case round complexity is

$$\left(c + \tfrac{c_{\mathsf{pca}}}{L}\right)\left(fL\lceil \tfrac{\Delta_3}{L} \rceil + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3) + 2(L-1)\right) + \Delta_0^*,$$

where f *is the actual number of corruptions,* $c, p, \Delta_0, \Delta_1, \Delta_2, \Delta_3$ *are the constants of* π_{rand} *in Definition 3.2,* c_{pca} *is the worst-case round complexity of* π_{pca}, *and* $\Delta_0^* := \Delta_0 + c_{\mathsf{pca}}\lceil \Delta_0/(Lc) \rceil$.

Proof. We prove validity, agreement, termination, and round complexity separately. Consider executions with at most $t < \min\{t_{\mathsf{rand}}, t_{\mathsf{pca}}\}$ corruptions so that both π_{rand} and π_{pca} remain secure. A phase of π_{rand} takes c rounds by Property (P1), and the compiler adds one invocation of π_{pca}, costing c_{pca} rounds, after every L-th phase and nowhere else; so a run of M phases occupies at most $M(c + \frac{c_{\mathsf{pca}}}{L})$ rounds — the signaling round falls $\lfloor M/L \rfloor \le M/L$ times — and it suffices to count phases.

The argument covers adaptive adversaries once "honest" is read at the right time. Corruption is irrevocable, so a party honest at some point was honest throughout until then, and validity and agreement concern the parties that are never corrupted. Against an adaptive adversary, we take the guarantees of π_{pca} to hold conditioned on the history so far, whatever corruptions it contains, while Property (P3) already speaks of a whole sequence of windows. What remains is to check that the compiler's own steps, i.e. signaling, the sets S and BAD, and injections, give such an adversary no extra leverage, and we note each place where corrupting a party mid-execution matters. The same reading covers a strongly adaptive adversary, which can also alter the messages a party sends in the round in which it is corrupted. Every step below that relies on a message being delivered concerns a party honest throughout that round, and a party corrupted mid-round is simply a corrupted party.

Validity. Assume all honest parties start with the same input value v. Then all honest parties start the first phase of π_{rand} with local value v and none has committed, so by Property (P2) they keep their local values equal to v and all commit to v by the end of phase $\Delta_1 + 1$. The compiler attempts an injection only at the end of a phase $r > \Delta_1$, and only at a party with $\mathit{output} = \perp$; since every honest party has $\mathit{output} \neq \perp$ by the end of phase $\Delta_1 + 1$, no honest party ever injects in such an execution. Hence, by Property (P1), every honest output is v.

Agreement. Let P_g be the first party to commit and then signal while still honest, that is, while honest throughout its next invocation of π_{pca}, say committing to v in phase r, and let r^* be the first signaling phase at or after r. No honest party has halted by the invocation of π_{pca} in r^*, since a party halts only after committing and signaling, and one that had done both while honest would contradict the choice of P_g. So π_{pca} has all its properties there, and honest-sender delivery carries P_g's ready to every honest party. As P_g sent unready in every earlier signaling round and, being honest, was never observed as $\perp$, it is not in BAD, and every honest party adds it to S. It stays there until its counter reaches $\lceil \Delta_3/L \rceil$, even if it halts or is corrupted, and only uncommitted parties advance that counter. So no honest party injects during the Δ_3 phases after r. Before r^* there is no signaling phase at which to do so, and from r^* on each honest party either has P_g in S or has committed. The window of Property (P4) is therefore undisturbed, all honest parties commit to v within Δ_3 phases, and by Property (P1) they all output v.

Two further points. After r^* honest parties start halting and π_{pca} may lose its guarantees, but nothing depends on it any more, since P_g leaves S only through its counter and a halted party

observed as $\perp$ merely ends up in BAD, which only suppresses signals that are no longer needed. And commitments made before P_g's, by parties corrupted before or during their signaling, change nothing. The injections they left unblocked fell on uncommitted parties, which Property (P4) tolerates since it holds from any phase-start local values, and one made after the previous signaling phase lies in the same undisturbed window, so its value is v as well.

Termination and round complexity. For the expected round complexity, group the phases into consecutive blocks of L. Injections happen only at the end of a block, so the first $\Delta_2 + 1$ phases of every block form an undisturbed window. Now take any block that begins before all honest parties have committed. If no honest party has committed by its start, Property (P3) applies to its window. Otherwise some honest party had already committed before the block began; it was honest through the first signaling round after its commitment and signalled there, so by the agreement argument all honest parties commit within Δ_3 phases of that commitment, which is by the end of the current block when $\Delta_3 \leq L$. So a block ends with all honest parties committed whenever its window begins with some honest party committed or ends with all of them committed. By Property (P3), applied to the windows of the first m blocks with m beyond the worst-case number of blocks bounded below, which is polynomial, so that one of them qualifies, the expected index of the first such block is at most $1/p$, and all honest parties have committed within L/p phases in expectation. That block ends at a signaling phase, so each party has by then also passed the first signaling round after its commitment, which the halting rule waits for. For $\Delta_3 > L$ the second case may run past the end of its block, and the last parties to commit then wait for the next signaling phase, up to $L(\lceil \Delta_3/L \rceil - 1)$ phases past the block, which gives $L/p + L(\lceil \Delta_3/L \rceil - 1)$ in general. By Property (P1) each party halts once Δ_0 further rounds of π_{rand} have also passed, which the compiled execution spans in Δ_0^* rounds. Multiplying the phase count by $c + \frac{c_{\mathsf{pca}}}{L}$ and adding these Δ_0^* rounds gives the claimed expectation.

We next analyze the worst-case round complexity. If all honest parties start with the same value, then by Property (P2) they commit by the end of phase $\Delta_1 + 1$. Otherwise, from phase $\Delta_1 + 1$ onward an honest party attempts the default-value injection at every phase divisible by L, provided no non-bad party has signalled ready; the first such phase is at most $\Delta_1 + L$.

The only obstruction to this injection is a fake ready signal by a corrupted party. Due to the use of π_{pca}, such misbehavior is consistently observed by all honest parties, in the sense that for each sender either all honest parties observe the same value, or some observe $\perp$ and immediately transfer that party to BAD. Once a corrupted party enters the set S of some honest party, its counter there is increased in every subsequent signaling round in which that honest party remains uncommitted, and since the protocol moves P_j to BAD as soon as $w_j \geq \lceil \Delta_3/L \rceil$, each corrupted party obstructs at most $\lceil \Delta_3/L \rceil$ injection attempts. Injection attempts are L phases apart, so with at most f corrupted parties the obstructions last at most $fL\lceil \Delta_3/L \rceil$ phases, and since they are counted in whole attempts, the next attempt after them is again at a phase divisible by L.

At that phase, either some honest party has already committed, and then no honest party injects and, by the agreement argument, all honest parties commit within Δ_3 additional phases; or none has, and then every honest party injects, all of them begin the next phase holding ϕ, and by Property (P2) they keep ϕ, which makes later injections no-ops, and commit within $\Delta_1 + 1$ further phases, one to reach the phase in which they are unanimous and Δ_1 more.

The second case needs the injection to be simultaneous, since a split one would leave honest parties at different values and Property (P2) would not apply. The sets S are not common to all honest parties, as crusader consistency lets a corrupted sender, or one corrupted during the invocation, be seen as ready by some honest parties and as $\perp$ by others, who place it in BAD instead. But a party enters honest parties' sets S, if at all, in a single round, and leaves them only through the counter, which all uncommitted honest parties advance together. So the phase reached

above, at which no corrupted party is in any honest party's S, is the same for all of them, and all of them inject there. Earlier split injections only hit uncommitted parties and are overwritten by this one.

Collecting the four contributions — reaching the first injection phase, clearing the obstructions, converging once the injection goes through, and reaching the signaling round after the last commitment, which the halting rule waits for —

$$\underbrace{(\Delta_1 + L)}_{\text{first injection}} + \underbrace{fL\lceil \Delta_3/L \rceil}_{\text{obstructions}} + \underbrace{\max(\Delta_1 + 1, \Delta_3)}_{\text{convergence}} + \underbrace{(L-1)}_{\text{halting round}}$$
$$= fL\lceil \tfrac{\Delta_3}{L} \rceil + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3) + 2(L-1)$$

phases. Multiplying the phase count by $c + \frac{c_{\mathsf{pca}}}{L}$ and adding the Δ_0^* trailing rounds that Property (P1) requires after committing gives the claimed bound. □

We can instantiate π_{rand} using Feldman–Micali [FM97] and the two-round π_{pca} of Proposition A.1 (both are setup-free and work for $t < \frac{n}{3}$), taking Property (P3) with $\Delta_2 = 0$, since Feldman–Micali's coin and its confirmation step already lie within a single phase under its own natural presentation. A phase of that protocol takes $c = 40$ rounds when its own common coin is used, so it achieves an expected round complexity of $\frac{40}{0.35} \leq 115$, but no worst-case guarantee at all. By contrast, applying our compiler to π_{rand} yields the following guarantees:

Corollary 3.4. *For $t < \frac{n}{3}$, there exists a perfectly secure Byzantine agreement protocol against up to t corruptions, using no setup, with expected round complexity at most 123 and worst-case round complexity at most $42f + 87$, where $f \leq t$ is the actual number of corruptions.*

Proof. We instantiate π_{rand} with the protocol of [FM97], denoted FM. In its native presentation, FM repeatedly executes, in order, the randomized subprotocol P_r followed by the two confirmation subprotocols P_0 and P_1. We take one execution of (P_r, P_0, P_1) to constitute one phase, identify the compiler's local value b_i with FM's variable of the same name, and say that a party commits when it executes the output instruction in P_0 or P_1. We take π_{pca} to be the two-round protocol of Proposition A.1, so $c_{\mathsf{pca}} = 2$.

We use the terminology of [FM97], where the network is said to be *in agreement on b* if every honest party is either already terminated with output b or still active with local value b. The proof of Theorem 4 there establishes the following, which is all we use:

(i) agreement on b is preserved by each of P_r, P_0, P_1 (Claim T4-2);

(ii) whenever an honest party outputs b, the network is already in agreement on b (Claim T4-3);

(iii) if the network is not in agreement at the beginning of P_r, then with probability greater than .35 it is in agreement by the end of P_r (Claim T4-4); and

(iv) if the network is in agreement on 0 at the beginning of P_0, or on 1 at the beginning of P_1, then every honest party outputs that value and terminates one round later (Claim T4-5).

For Property (P1), the proof of Theorem 4 of [FM97] accounts for P_r, P_0 and P_1 as 36, 2 and 2 rounds, so one phase has length $c = 40$.[11] A committed party is dead in FM's sense and its local value is never read again, so an injection into it is vacuous, as (P1) requires. By (iv) a party that outputs sends its decided bit for one further round before halting, so $\Delta_0 = 1$; that round may fall beyond the phase boundary, which the compiled execution reaches only after its c_{pca} signaling rounds, giving $\Delta_0^* = 3$.

[11] The remarks in [FM97] instead describe P_r as "a quite complex 34-round protocol"; we quote the figure used in the proof of its Theorem 4, which is the conservative one. A cheaper common coin lowers c, and with it every constant below, since all of them are of the form $(c+2)\cdot(\,\cdot\,)+3$.

For Property (P2), suppose all honest parties enter a phase with the same local value v and none has committed. Then the network is in agreement on v at the beginning of P_r, and by (i) it remains so throughout the phase. If $v = 0$, applying (iv) to that phase's P_0 makes every honest party commit to 0; if $v = 1$, agreement is preserved through P_0 and applying (iv) to P_1 makes every honest party commit to 1. In both cases all honest parties commit within that same phase, so $\Delta_1 = 0$.

For Property (P3), consider a phase entered with no honest party committed. If the network is already in agreement, the argument for (P2) gives commitment within that phase. Otherwise, by (iii), with probability greater than .35 the network is in agreement by the end of P_r, and (iv) then makes every honest party commit during that phase's P_0 or P_1. As P_r draws a fresh coin in every phase, this bound holds given the history up to the phase, so Property (P3) follows as noted after Definition 3.2. Hence $p = .35$ and $\Delta_2 = 0$, and $L = 1$ is admissible.

For Property (P4), let g be the first honest party to commit, to some v. By (ii) the network is already in agreement on v at that moment, and by (i) it stays so. Hence, no later than the confirmation subprotocol P_v of the following phase, (iv) makes every remaining honest party commit to v. Thus $\Delta_3 = 1$.

Validity follows from the theorem, since local values start at the actual inputs and (P2) holds uniformly. Substituting $p = .35$, $\Delta_0 = 1$, $\Delta_1 = 0$, $\Delta_2 = 0$, $\Delta_3 = 1$ and $L = 1$ into Theorem 3.3 gives expected round complexity at most $\frac{c+2}{.35}+3$ and worst-case round complexity at most $(c+2)(f+2)+3$; with $c = 40$ these are at most 123 and $42f + 87$. □

The same threshold admits a second, considerably cheaper instantiation. The $t < \frac{n}{3}$ protocol of Thm. 11 of [KK09] (denoted KK_3) is also information-theoretic, and unlike Feldman–Micali it is built to have its randomized step amortized away, which is exactly the structure Property (P1) licenses us to keep.

Corollary 3.5. *For $t < \frac{n}{3}$, there exists a perfectly secure Byzantine agreement protocol against up to t corruptions, using no setup, with expected round complexity at most 24 and worst-case round complexity at most $14f + 31$, where $f \le t$ is the actual number of corruptions.*

Proof. We instantiate π_{rand} with the binary Byzantine agreement protocol of Theorem 11 of [KK09], denoted KK_3, and π_{pca} with the two-round protocol of Proposition A.1, so $c_{\mathsf{pca}} = 2$; both are setup-free and unconditional at $t < \frac{n}{3}$. An iteration of KK_3 consists of six steps: Steps 1–5 are single-round all-to-all exchanges of the local bit, and Step 6 runs an oblivious leader election and has each party with $\mathsf{update}_i = \mathsf{true}$ adopt the elected leader's bit, terminating if $\mathsf{exitBA}_i = \mathsf{true}$. The leader election is the unauthenticated protocol of Theorem 8 of [KK09], whose fairness is $\frac{n-t}{n} - \frac{1}{n^2}$ and hence at least $\frac{2}{3}$ for $t < \frac{n}{3}$ (Corollary 9 there), and whose first phase takes 8 rounds and second phase a single round.

The first phase of that leader election samples and shares fresh values and never reads the local bit, so by Property (P1) we may run it in the background: each iteration then costs only Steps 1–5 and the one round of the second phase, that is, 6 rounds. We schedule one background execution per iteration, timed so that its first phase ends at the Step-5 round of the iteration that consumes it; two executions are then in flight at any time, and the earliest one starts $8 - 5 = 3$ rounds before the protocol proper, which we account for as an additive prefix.[12]

[12] We stagger the background executions of the leader-election protocol so that its first phase completes in the iteration in which the elected leader is used. This alignment is needed because Definition 6 of [KK09] guarantees that the elected leader is honest at the end of that first phase. The executions can be staggered without increasing the amortized round complexity, while retaining the three-round initialization prefix. We use the same scheduling idea, with a different alignment, for the leader-election invocation in $\mathsf{Comp}^{\mathsf{IC}}$ (Corollary C.1).

We take the local value to be KK_3's bit b_i, and say that a party commits when it outputs and terminates in Step 6. Its output is fixed at that instant, and nothing is owed to the others afterwards: as [KK09] observes, a party that terminates in an iteration has already sent its bit in Step 5 and "can be viewed as continuing to send" it thereafter, which is precisely the convention Steps 1–5 use for missing messages. Hence $\Delta_0 = 0$. Finally, we take *two* consecutive iterations to be one phase, so $c = 12$; as with the protocol of Corollary 3.8, this is what drives Δ_2 to 0.

We use three facts established in the proof of Theorem 11 of [KK09]: (a) if all honest parties enter an iteration with the same bit b, then by the end of Step 5 they all hold b with $\mathsf{exitBA} = \mathsf{true}$, and every active honest party outputs b and terminates in that iteration; (b) if an honest party outputs in some iteration, then all honest parties output the same bit and terminate by the end of the next; and (c) if an honest leader is elected in Step 6 of some iteration, then all honest parties terminate by the end of the next.[13]

Property (P2) is fact (a): unanimity at a phase start yields commitment within the phase's first iteration, so $\Delta_1 = 0$. Property (P4) is fact (b), and the following iteration lies in the same phase or the next, so $\Delta_3 = 1$. For Property (P3), an honest leader is elected in the first iteration of a phase with probability at least $\frac{2}{3}$, and fact (c) then has every honest party commit by the end of the second iteration, that is, within the same phase; hence $p = \frac{2}{3}$ and $\Delta_2 = 0$, so $L = 1$ is admissible and no gating is needed. The election samples fresh values in every phase, so this bound holds given the history up to the phase, and Property (P3) follows as noted after Definition 3.2.

Substituting $c = 12$, $c_{\mathsf{pca}} = 2$, $p = \frac{2}{3}$, $\Delta_0 = 0$, $\Delta_1 = 0$, $\Delta_3 = 1$, $\Delta_2 = 0$ and $L = 1$ gives expected round complexity $14 \cdot \frac{1}{1-\frac{1}{3}} = 21$ and worst-case round complexity $14(f+2) = 14f + 28$; adding the 3-round prefix gives 24 and $14f + 31$. □

Both instantiations are setup-free, unconditional and optimal $t < n/3$, so Cor. 3.5 strictly dominates Cor. 3.4: roughly 5 times better in expectation and 3 times better in the coefficient of f. We retain both because they exercise the properties differently and show broad applicability of our compiler – Feldman–Micali has $\Delta_2 = 0$ under its own phase structure and no background computation, whereas KK_3 needs both the phase coarsening and the allowance Property (P1) makes for sub-protocols run in the background, which is what lets its leader election be pipelined – but it is the latter that we carry forward in Section 4.

3.1.2 Compiler in Signature PKI Model

The next protocol refines the coordination mechanism by replacing the π_{pca} with a PKI-based approach. Instead of relying on a π_{pca} to aggregate ready signals and detect equivocation, parties use digital signatures to disseminate and verify readiness messages. Each party signs its ready/unready signal and sends it to all others, while valid messages and any equivocation evidence are forwarded in the background so that all honest parties eventually obtain a consistent view. Every signature is bound to the session identifier of its invocation as well as to the phase index. The phase index alone would not do, because a single PKI may serve many invocations of the compiled protocol — $\mathsf{Comp}^{\mathsf{IC}}$ of Section 4 invokes it three times per iteration, in every iteration — and an unlabelled readiness signature would be a bearer token, which a corrupted party could harvest from one invocation and replay into another to block the default-value injection there.

This reduces the signaling overhead from two rounds to one and removes the resilience bound imposed by π_{pca}. It also lets a party signal and halt without waiting for a signaling round. A ready is now a signed message forwarded in the background, so a party sends it as soon as it commits

[13] The first is item (i) of that proof; the other two are the two unnumbered paragraphs that follow it. We relabel to avoid a clash, since the proof's own items (ii) and (iii) are different statements.

and halts once it has sent it and Property (P1) allows, and a party that commits in the middle of a phase need not sit out the rest of it. The rest of the compiler is unchanged.

Compiler $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand}})$

Below is the code for party P_i. Assume a signature PKI. Let pk_i and sk_i be party P_i's has signing and verification keys, respectively. Also, consider domain separation for signature using sid.
Parameters: the constants $\Delta_0, \Delta_1, \Delta_3$ of Properties (P1)–(P4), an injection period $L > \Delta_2$, and a fixed default value ϕ of the agreement domain.

On input $(\mathtt{input}, \mathsf{sid}, x)$, initialize $b := x$, $output := \perp$, $S := \emptyset$, $BAD := \emptyset$, $E := \emptyset$, and $\rho_j := \perp$ for all $j \in [n]$.

Main execution of the adjusted randomized π_{rand}

Repeat the following for phase $r = 1, 2, \ldots$:

1. Execute one full phase of π_{rand} starting from local value b, continuing from the previous internal state except for the local value b which may be externally updated, and let b denote the updated local value. If π_{rand} sets its commit flag during the phase, let $output$ denote its local output, output $(\mathtt{output}, output)$ at once, and in the next round send $(\mathsf{sid}, r, \mathsf{ready}, \sigma_i^r)$ with $\sigma_i^r := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, r, \mathsf{ready})$ to all parties. Halt at the end of that round or once Δ_0 further rounds of π_{rand} have been executed, whichever is later.
2. If $r \not\equiv 0 \pmod{L}$, skip this and the next step and proceed directly to the next phase; when no injection can occur at phase end, neither signaling nor its bookkeeping is needed. Otherwise, send $(\mathsf{sid}, r, \mathsf{ready}, \sigma_i^r)$ to all parties with $\sigma_i^r := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, r, \mathsf{ready})$ if $output \neq \perp$, and $(\mathsf{sid}, r, \mathsf{unready}, \sigma_i^r)$ with $\sigma_i^r := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, r, \mathsf{unready})$ if $output = \perp$.
3. Do the following:
   - For every $P_j \notin BAD$ such that E contains a valid message $(r', \mathsf{ready}, \sigma)$ from P_j, update $S := S \cup \{P_j\}$;
   - For each $P_j \in S$ with $r - \rho_j \geq \Delta_3$, set $BAD := BAD \cup P_j$ and $S := S \setminus P_j$;
   - If $r > \Delta_1$, $output = \perp$, and $S = \emptyset$, reset $b := \phi$.

Background evidence dissemination

1. In parallel with the main execution, upon receiving a valid signed message $(\mathsf{sid}, r', m, \sigma)$ from P_j with $r' \leq r$, $m \in \mathsf{ready}, \mathsf{unready}$, and $\mathsf{Ver}_{\mathsf{pk}_j}((\mathsf{sid}, r', m), \sigma) = 1$, add it to E; if $m = \mathsf{ready}$, $P_j \notin S$, and $\rho_j = \perp$ or $r' < \rho_j$, set $\rho_j := r'$; and forward the message to all parties if not already forwarded.
2. If E contains valid $(r', \mathsf{ready}, \sigma)$ and $(r', \mathsf{unready}, \sigma')$ messages from the same P_j, forward the pair as equivocation evidence and set $BAD := BAD \cup P_j$.

Figure 2: Randomized early-stopping BA compiler assuming signature PKI.

Theorem 3.6. *Assume EUF-CMA signature schemes exist. Let π_{rand} be a computationally secure injection-tolerant randomized Byzantine agreement protocol (Definition 3.2) tolerating fewer than t_{rand} corruptions, and let $L > \Delta_2$. Then $\pi := \mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand}})$ is a computationally secure Byzantine agreement protocol against up to t corruptions and the same adversary class as π_{rand} in the signature-PKI model, for every $t < t_{\mathsf{rand}}$. Its expected round complexity is*

$$\left(c + \tfrac{1}{L}\right)\left(\frac{L}{p} + L\left(\lceil \tfrac{\Delta_3}{L} \rceil - 1\right)\right) + \Delta_0^*,$$

and its worst-case round complexity is

$$\left(c + \tfrac{1}{L}\right)\left(fL\lceil \tfrac{\Delta_3}{L} \rceil + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3) + (L - 1)\right) + \Delta_0^*,$$

where f is the actual number of corruptions, $c, p, \Delta_0, \Delta_1, \Delta_2, \Delta_3$ are the constants of π_{rand} in Definition 3.2, and $\Delta_0^* := \max(\Delta_0, 1) + \lceil \Delta_0/(Lc) \rceil$.

Proof. Again, we prove each property separately, reading "honest" as in the proof for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$.
Validity. The validity argument is the same as in the proof for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$.
Agreement. Let P_g be the first party to commit while honest through the end of the first signaling round after its commitment, say committing to v in phase r, and let r^* be the first signaling phase at or after r. In the round after it commits, P_g signs $(\mathsf{sid}, r, \mathsf{ready})$ and sends it to all parties. Injections occur only at signaling phases, so none occurs between r and r^*. Having signed only $\mathsf{unready}$ before, P_g is not yet in S, and while honest it cannot be caught equivocating, since its signatures cannot be forged. So it is not in BAD at phase r^*, and its ready reaches every honest party, enters E, and puts P_g in every honest S.

This blocks the default-value injection, which is allowed only when $S = \emptyset$, and it does so for long enough. Indeed, $\rho_g = r$ at every honest party: P_g signed ready for no earlier phase while honest, its signature cannot be forged, and ρ_g is frozen once P_g is in S, so a backdated ready signed after a later corruption leaves it unchanged. And P_g is removed from S only once $r' - \rho_g \geq \Delta_3$ in a later phase r', since equivocation evidence, which such a corruption could also produce, only keeps a party out of S. As this test is performed after the corresponding phase of π_{rand} has been executed, the readiness signal remains active throughout the Δ_3 phases in which Property (P4) guarantees that all honest parties commit to v. Any injection into a party that has itself committed is a no-op by Property (P1). Hence the window of Property (P4) is undisturbed and, by Property (P1), all honest parties output v. Commitments before P_g's, by parties corrupted before the end of the first signaling round after their commitment, are handled exactly as for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$.
Termination and round complexity. The expected-round argument is the block-by-block one given for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$. Its only new ingredient is in the case of a block that begins with some honest party committed. There the party that signalled while honest stays in every honest S for Δ_3 phases whatever happens to it afterwards, as the agreement argument has just shown. So again all honest parties have committed within $\frac{L}{p} + L(\lceil \Delta_3/L \rceil - 1)$ phases in expectation, and they halt Δ_0^* rounds later. A phase of π_{rand} takes c rounds, and one signaling round follows every L-th phase, the background forwarding running in parallel without adding a round. So a run of M phases takes at most $M(c + \frac{1}{L})$ rounds, which gives the claimed expectation.

The worst-case argument follows the one for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$, except that the sets S, BAD, E and the values ρ_j are now local, and honest parties may hold different ones at the same phase, since a signed message reaches a party either directly or after a round of forwarding. We write $S^{(i)}$, $BAD^{(i)}$, $E^{(i)}$ and $\rho_j^{(i)}$ for the copies held by P_i. Here an honest party is one that is never corrupted, so that everything it sends or forwards is delivered unaltered. Two facts take the place of crusader consistency.

First, a corrupted party P_j enters $S^{(i)}$ at a signaling phase at which P_i holds a valid ready from it and no equivocation evidence against it. The value $\rho_j^{(i)}$ is then frozen at the phase of the earliest valid ready held so far, which is at most that phase, and P_j is removed at the first signaling phase r with $r - \rho_j^{(i)} \geq \Delta_3$. Neither later nor backdated ready messages can extend its stay, and this holds equally for a party that signalled ready while honest and was corrupted afterwards. Second, every honest party forwards each valid signed message and each equivocation pair the first time it sees it, so whatever one honest party holds at the end of a phase all of them hold one phase later. Hence, once P_j has entered some honest party's S, every honest party holds a ready from it whose phase is no later than that entry, and P_j is out of every honest $S^{(i)}$ by the first signaling phase at least Δ_3 phases after its first entry. Across all honest parties it therefore obstructs at most $\lceil \Delta_3/L \rceil$ injection attempts.

So with at most f corrupted parties all fake readiness signals together delay the reset for at most $fL\lceil \Delta_3/L \rceil$ phases; adding the at most $\Delta_1 + L$ phases before the first injection phase, we reach

a phase divisible by L at which no corrupted party remains in any honest $S^{(i)}$, the obstructions being counted in whole attempts. At that phase, either some honest P_i has an honest party in $S^{(i)}$, in which case that party has committed and signalled while honest, and by the agreement argument all honest parties commit within Δ_3 additional phases; or $S^{(i)} = \emptyset$ at every honest P_i, in which case every honest party that has not committed resets its local value to ϕ and, by Property (P2), all honest parties commit within $\Delta_1 + 1$ additional phases—the honest local values remaining equal to ϕ, so that later injections are no-ops. The two cases are exhaustive, and the one that applies is the same at all honest parties, since an honest party sends its signed ready to all of them directly and so enters all their sets S in the same round.

The tally is the one made for $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}$ without its last contribution, as a party no longer waits for a signaling round to halt:

$$\underbrace{(\Delta_1 + L)}_{\text{first injection}} + \underbrace{fL\lceil \Delta_3/L \rceil}_{\text{obstructions}} + \underbrace{\max(\Delta_1 + 1, \Delta_3)}_{\text{convergence}}$$
$$= \; fL\lceil \tfrac{\Delta_3}{L} \rceil + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3) + (L - 1)$$

phases. Multiplying the phase count by $c + \frac{1}{L}$ and adding the Δ_0^* rounds after the last commitment, in which a party sends its ready and executes the further rounds of π_{rand} that Property (P1) requires, gives the claimed bound. For $L = 1$ this is $f\Delta_3 + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3)$. □

We can instantiate π_{rand} with the protocol of [Mic17], which also operates in the signature-PKI model for $t < \frac{n}{3}$. Although it terminates in an expected 9 rounds, it provides no worst-case round bound. Applying our framework instead yields the following guarantees:

Corollary 3.7. *Assume unique-signature EUF-CMA schemes exist in the random-oracle model. For $t < \frac{n}{3}$, there exists a computationally secure Byzantine agreement protocol against up to t corruptions in the signature-PKI model, with expected round complexity at most 14 and worst-case round complexity at most $4f + 10$, where $f \leq t$ is the actual number of corruptions.*

Proof. We instantiate π_{rand} with the protocol of [Mic17], denoted $BBA^\star$, whose natural presentation is a three-step loop consisting of a Coin-Fixed-To-0 step, a Coin-Fixed-To-1 step and a Coin-Genuinely-Flipped step, in that order, each costing one round. Following the phase-alignment discussion above, we take one phase to consist instead of a Coin-Genuinely-Flipped step immediately followed by the next Coin-Fixed-To-0 and Coin-Fixed-To-1 steps; equivalently, $BBA^\star$'s execution is taken to begin with a Coin-Genuinely-Flipped step, using the actual input as the local value entering it. A party commits when it halts, which happens only in the two Coin-Fixed steps and fixes its output at that moment.

We use the following, established in [Mic17]:

(i) if all honest parties hold b entering a Coin-Fixed-To-b step, every one of them halts with output b in that step (Property 2);

(ii) with probability at least $1/3$, all honest parties hold the same value at the end of a Coin-Genuinely-Flipped step (Claim A);

(iii) once all honest parties hold b, they continue to hold it (Claim B); and

(iv) if an honest party halts with output b in a Coin-Fixed-To-b step, then all honest parties hold b by the end of that step (Claim C).

For Property (P1), a phase is three rounds, so $c = 3$. By $BBA^\star$'s halting rule a party that halts propagates the special value 0^* (respectively 1^*) for one further round, after which every receiver reuses that bit for it in all later steps, so $\Delta_0 = 1$; reusing the last bit heard would not do, since

a party may halt with output b having propagated $1-b$ at the start of the very step in which it halts. For a Coin-Fixed-To-1 halter that trailing round falls in the next phase, which the compiled execution reaches only after its signaling round, so $\Delta_0^* = 2$.

For Property (P2), suppose all honest parties enter a phase with the same local value v and none has halted. Unanimity carries them through the phase's Coin-Genuinely-Flipped step by that step's counting rules rather than its coin substep, so they still hold v entering the Coin-Fixed-To-0 step. If $v = 0$, (i) makes every honest party halt with output 0 there; if $v = 1$, none halts there and (i) applied to the following Coin-Fixed-To-1 step makes every honest party halt with output 1. In either case all honest parties commit within the same phase and their local values remain v, so $\Delta_1 = 0$.

For Property (P3), suppose no honest party has halted entering a phase. If the honest parties already agree, the argument for (P2) applies verbatim. Otherwise (ii) gives, with probability at least $1/3$, unanimity at the end of that phase's Coin-Genuinely-Flipped step, and the argument for (P2), resumed at the Coin-Fixed-To-0 step, has every honest party commit before the phase ends. Hence $p = 1/3$ and $\Delta_2 = 0$, so $L = 1$ is admissible. This is what the rotation buys: under $BBA^\star$'s own presentation the coin ends the phase, no party halts in it, and commitment would fall into the next, giving $\Delta_2 = 1$. The coin is built from each party's signature over the public random string and the loop counter, which we identify with the compiler's phase index so that honest parties never sign different counters. Across phases we rely on the expected round complexity claimed in [Mic17], which applies Claim A at the start of every iteration of its loop, whatever values the honest parties then hold, so the first window that does not fail comes within an expected 3 windows, which is Property (P3) with $p = 1/3$. The corollary assumes unique-signature EUF-CMA in the random oracle model because the fairness bound needs that coin unbiasable: with several valid signatures per message, a corrupted signer could bias its own coin by choosing which to reveal.

For Property (P4), a party halts during a Coin-Fixed-To-b step. By (iv) all honest parties hold b by the end of that step, and by (iii) they continue to hold it, so the stragglers enter the following phase unanimous on b and commit there by the argument for (P2). Thus $\Delta_3 = 1$.

Validity follows from the theorem, since local values start at the actual inputs and (P2) holds uniformly under this rotation. Finally, $BBA^\star$ tolerates $t < \frac{n}{3}$ corruptions and terminates in an expected 9 rounds, consistent with $c = 3$ and $p = 1/3$. Substituting $c = 3$, $p = 1/3$, $\Delta_0 = 1$, $\Delta_1 = 0$, $\Delta_2 = 0$, $\Delta_3 = 1$ and $L = 1$ into Theorem 3.6 gives expected round complexity at most $(c+1)\cdot\frac{1}{p}+2 = 4\cdot 3+2 = 14$ and worst-case round complexity at most $(c+1)(f+2)+2 = 4f+10$. □

We can also instantiate π_{rand} using the point-to-point, signature-based protocol of Appendix B of [KK06] (underlying Lemma 18 there), which operates in the signature PKI model and tolerates $t < \frac{n}{2}$ corruptions, together with an oblivious leader election protocol π_{ole} achieving fairness $p \geq \frac{1}{2}$ against $t < \frac{n}{2}$ corruptions. The protocol's own scheduling of π_{ole} calls across iterations is moot for the instantiation we give below, whose leader election takes a single round, so we use the unpipelined per-iteration structure here; Corollary 3.5 shows how that scheduling is exploited when it does pay.

For π_{ole} we do not use the leader election of [KK06] itself, whose round complexity dominates everything else, but a VRF-based sortition protocol: each party evaluates a VRF on the session identifier and the current iteration number and, in a single round, broadcasts its output together with a proof of correct evaluation; every party then elects, among the valid outputs it received, the party achieving the (canonically ordered) smallest VRF output as leader. Against $t < \frac{n}{2}$ corruptions this has fairness $p \geq (n-t)/n > \frac{1}{2}$ and round complexity $R_{\mathsf{ole}} = 1$, assuming VRF schemes with exponential-size range in the VRF-PKI model of Section 2.1. Two features of that model carry the bound. Keys are sampled by the setup rather than chosen by the adversary, so

residual pseudorandomness applies to all n values at the seed and the smallest of them belongs to a party honest at the start of the invocation with probability at least $(n-t)/n$; and an honest party's output reaches every party within the single round, so no honest party can be made to overlook the minimum, which is what supplies the unanimity that Definition 2.2 requires. Withholding a corrupted party's output only removes a candidate and cannot displace an honest minimum. All invocations use the same VRF keys instead of separate setups, each evaluating the VRF at its own seed. The seeds are distinct and fixed before the setup, as the session identifier is chosen before it. Then the first k elections all fail to elect a party honest at their start with probability at most $(1-p)^k$, up to a negligible error. For uniform values, the smallest value of a corrupted party at a seed is a bar that an honest value must beat. A corruption may lower it, but in expectation it costs the honest parties only the corrupted party's own share of their chance, whatever the order of corruptions and elections. Pseudorandomness then lets us replace the VRF values by uniform ones, one corruption or election at a time. Each reduction stops right after its step, so it never has to hand the adversary the key it is challenged on. Hence an iteration of the protocol of [KK06] takes $c = 6 + R_{\mathsf{ole}} = 7$ rounds. It is this improvement of [KK06] that we compile. See Corollary 3.8.

Corollary 3.8. *Assume EUF-CMA signature schemes and VRF schemes with exponential-size ranges exist. For $t < \frac{n}{2}$, there exists a computationally secure Byzantine agreement protocol against up to t corruptions in the signature-PKI and VRF-PKI model, with expected round complexity at most 37 and worst-case round complexity at most $15f + 29$, where $f \leq t$ is the actual corruption count.*

Proof. We instantiate π_{rand} with the protocol of Appendix B of [KK06] (underlying Lemma 18 there), denoted KK, under its own presentation, whose iterations consist of six single-round all-to-all exchanges (of signatures in Steps 1 and 3, of certificates in Steps 2, 4 and 5, and of a bare value in Step 6) followed by a seventh step that invokes π_{ole} and then applies a value-adoption rule and a termination check, so that an iteration takes $c = 6 + R_{\mathsf{ole}}$ rounds. One iteration is one phase of π_{rand}, of c rounds; as will be clear from (P3) below, the confirmation completes only in the iteration after the election, so $\Delta_2 = 1$ and the injection period is $L = 2$. The local value is KK's variable v_i, and a party commits when it sets $\mathsf{lock}_i := 1$ in Step 4; as in the analogous step of Theorem 13 of [KK06], we read that assignment as guarded by $\mathsf{lock}_i = \infty$. The preamble's convention that a locked v_i never changes is exactly (P1)'s requirement that an injection into a committed party has no effect, and it fixes the party's output at the moment it commits. We take the compiler's default value to be KK's own default $\phi \in V$, so that an injected value always lies in the protocol's domain; Step 7 leaves no honest party at $\perp$, so phase-start local values always lie in V too.

We use three facts established in the proof of Lemma 18 of [KK06]: (a) if all honest parties hold a common value v immediately before some iteration and none has yet terminated, then all of them terminate with output v by the end of the following iteration; (b) in the first iteration in which an honest party sets $\mathsf{lock} := 1$, every honest party holds that party's value by the end of the iteration and none has terminated by then; and (c) if an honest leader is elected in some iteration, then all honest parties hold a common value by the end of that iteration.[14] It is convenient to read (a) through the termination rule of Step 7, which lets a party terminate only with $\mathsf{lock} = 0$ and turns lock to 0 only at the Step 7 following the Step 4 that set it to 1: a party terminating at the end of an iteration locked in Step 4 of the previous one, so (a) says that every honest party has locked, on v, by Step 4 of the iteration they entered unanimously.

For (P2), a party that has not committed has not terminated, so the hypothesis of (a) is met whenever all honest parties enter a phase holding the same $v \in V$ with none committed, and the reading above places all of their commitments in that very phase. Hence $\Delta_1 = 0$.

[14] All three are stated as unnumbered prose there; the labels are ours.

For (P3), with probability at least p the leader elected in Step 7 is honest, and (c) then makes all honest values equal at the end of that iteration. Since locking happens in Step 4, three rounds before the election, commitment follows only in the next iteration: no honest party had committed entering the electing iteration, so none terminates in it, and the argument for (P2) applies to the one that follows. So with probability at least p all honest parties commit by the end of the phase following the one in which the election falls, giving $\Delta_2 = 1$ and hence $L = 2$: injections are attempted at every second phase, and the one boundary they could disturb is the one between the electing phase and the locking phase, which $L = 2$ leaves alone. The fairness bound applies as stated because π_{ole} guarantees a leader that is honest at the start of its invocation, which here is the start of Step 7, and the argument behind (c) needs the elected leader honest only through Step 6, where it relays its value. Across windows, take any windows fixed in advance. A window whose election some honest party skips does not fail: that party has halted, so it committed before the window, as no honest party commits in an electing iteration entered with none committed. We may therefore apply the bound for the sortition above as if halted parties kept running the elections. The first k windows then all fail with probability at most $(1-p)^k$, so the expected index of the first window that does not fail is at most $\sum_{k\geq 0}(1-p)^k = 1/p$, which is Property (P3).

For (P4), fact (b) leaves every honest party at the end of the first locking iteration holding the locker's value with none yet terminated, so the reading of (a), applied to the iteration that follows, has them all locked by its Step 4. That iteration is the next phase, so $\Delta_3 = 1$.

It remains to determine Δ_0. A party that locks in Step 4 of iteration I has its output fixed but must keep signing: with $n = 2t + 1$ the honest parties are a bare majority, so its Step-3 signature in iteration $I + 1$ is needed for the stragglers' Step-4 certificate there. Nothing is needed of it afterwards, since by that point every honest party is locked. Counting from Step 4 of I, this covers Steps 5–7 of I and Steps 1–3 of $I + 1$, that is, $\Delta_0 = 2 + R_{\mathsf{ole}} + 3 = c - 1$ rounds of KK; these span fewer than Lc rounds, so they meet at most one signaling round and the compiled execution covers them in $\Delta_0^* = (c-1) + 1 = c$ rounds. This is why we designate "commits" as the locking event rather than KK's own termination event: halting a party the instant it locks would drop it from the signer pool before its signature is used, whereas the compiler lets it output at once and halt Δ_0 rounds later.

Validity follows exactly as in the theorem's proof, since local values are initialized to the actual inputs and (P2) holds uniformly. Substituting $p = \frac{1}{2}$, $\Delta_0 = c - 1$, $\Delta_1 = 0$, $\Delta_3 = 1$, $\Delta_2 = 1$ and $L = 2$ gives $L(\lceil \Delta_3/L \rceil - 1) = 0$, so the expected phase count is $L/p = 4$ and the worst-case phase count is $fL\lceil \Delta_3/L \rceil + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3) + (L - 1) = 2f + 3$. With $c + \frac{1}{L} = c + \frac{1}{2}$ these give $4(c + \frac{1}{2}) + c$ and $(2f + 3)(c + \frac{1}{2}) + c$; with $c = 7$ they are $30 + 7 = 37$ and $15f + 22.5 + 7$, that is, at most $15f + 29$, a round count being an integer. □

The injection period $L = 2$ is what $BBA^\star$ avoids by rotation, and no rotation avoids it here. Unlike $BBA^\star$, whose native presentation merely mislabels where a phase begins – a gap phase-rotation closes – KK's locking step and its leader election sit three rounds apart in a fixed order, so any rotation that places the election and the ensuing lock inside a single iteration would have to cut the round sequence between Steps 4 and 7; but an injection there is either erased by Step 5, which resets an uncertified value to $\perp$, or survives into Step 7 and destroys the invariant that every honest value lies in $\{v^*, \perp\}$ for the unique certified v^*, on which Lemma 18's honest-leader argument rests. Declaring two consecutive iterations to be one phase would give $\Delta_2 = 0$ and $L = 1$ instead, and gains nothing, as the injections fall at the same points and the signaling rounds are equally many, since the compiler signals once per injection period either way, and Theorem 3.6 even gives it the weaker worst-case bound $15f + 37$, as it counts the last phase whole.

4 Randomized Early-Stopping IC

Since interactive consistency follows from parallel Byzantine agreement with one additional round (Lemma A.2 in Appendix A), we focus on parallel BA.

4.1 Transformation for Randomized Early-Stopping Parallel BA

Section 1.3 outlines our construction of N-parallel Byzantine agreement (Definition 2.4). It combines four components, namely truncated executions of a randomized BA protocol π_{rand}, $m = \lceil \log N \rceil$ of them per coordinate as in [BOEY03]; an early-stopping BA protocol π_{es} per coordinate, run in parallel as a fallback whose outputs reset those executions; a randomized early-stopping BA protocol π_{res} that coordinates the parties, for instance the output of either compiler of Section 3 on an injection-tolerant BA (Definition 3.2); and a leader election π_{ole}. The truncated executions require the following properties of π_{rand}.

Definition 4.1 (Unanimity-responsive BA)**.** π_{rand} *is* unanimity-responsive*, with constants e and c, if it satisfies:*

- ***(Q1) Expected-constant-round.*** *π_{rand} has constant expected round complexity e.*
- ***(Q2) 1-shift.*** *All honest parties terminate within one round of each other.*[15]
- ***(Q3) Unanimity convergence.*** *If all honest parties start with the same input v, then they all output v within a constant c number of rounds.*

We note that our transformation below can be easily extended to randomized BA protocols satisfying a relaxed version of property (Q2), in which only a δ-shift is guaranteed for some constant $\delta \geq 1$ (i.e., all honest parties terminate within δ rounds of each other), rather than the 1-shift stated above. However, any such δ-shift protocol can be converted into a genuinely 1-shift one by a lightweight termination-synchronization layer run on top of it, at the cost of $O(1)$ additional rounds folded into Properties (Q1) and (Q3) (leaving Property (Q2) itself at $\delta = 1$). For $t < \frac{n}{3}$ with no setup, the approach of [Bra87], in the form used by [CCGZ16], gives such a reduction. Each party echoes a value once it is ready to output it or upon receiving $t + 1$ matching echoes, and outputs upon receiving $n - t$ matching echoes, sending its own echo before it halts. One round suffices as every honest party echoes by the round in which the last of them becomes ready, so every honest party holds $n - t$ matching echoes one round later; and the layer is 1-shift, since a party that collects $n - t$ echoes has collected at least $n - 2t \geq t + 1$ honest ones, which reach every honest party in the same round and trigger its echo. For $t < \frac{n}{2}$ in the PKI model, the standard analogue is signature-based. Writing λ for the label of the instance the layer wraps, each party signs and sends $(\lambda, \mathsf{exit}, v)$ for its (once-decided) value v, forwards any valid signature on $(\lambda, \mathsf{exit}, v)$ it has not yet forwarded before halting, and outputs v as soon as it holds $t + 1$ valid signatures on $(\lambda, \mathsf{exit}, v)$ under distinct verification keys; the same argument gives one additional round and a 1-shift. This remains safe at the weaker threshold $t < \frac{n}{2}$ precisely because signatures cannot be equivocated, so any $t + 1$ signatures on $(\lambda, \mathsf{exit}, v)$ necessarily include one from a party that signed it *in this instance* and was therefore honestly ready to output v here – unlike Bracha's plain-message echo, whose safety instead relies on quorum intersection and hence needs $t < \frac{n}{3}$. For simplicity, we focus on the 1-shift case, with explanations in the compiler and Thm. 4.2 proof how it adapts to δ-shift.

Fast sequential composition. Both $\mathsf{Comp}^{\mathsf{IC}}$ and the protocols it invokes have probabilistic termination, and honest parties therefore may not leave an invocation in the same round, which makes

[15] Our Thm. 4.2 proof also handles δ-shift; also see below for how to turn it into 1-shift.

sequential composition delicate. The difficulty and its resolution are well understood in our setting, due to Lindell, Lysyanskaya and Rabin [LLR02], and were later recast in the simulation-based framework by Cohen et al. [CCGZ16]. One keeps the parties within a known constant gap rather than in lockstep, expands each round into a window wide enough to absorb that gap, and restores the gap after every probabilistic-termination invocation with a termination-synchronization layer. This overhead is not an artifact of our construction; the parallel-broadcast protocol of [BOEY03] that we build on incurs it too. Concretely, in $\mathsf{Comp}^{\mathsf{IC}}$ the gap is one round, so every round of its Procedure 2 is executed as a window of $2 \cdot 1 + 1 = 3$ rounds, a party listening throughout and sending in the middle round, and each of the three invocations of π_{res} (the only sub-protocols with probabilistic termination) is followed by one plain round of the termination-synchronization layer.[16] Procedure 1 is left unexpanded, since a single invocation starts all honest parties in the same round; Section 5.2 expands it as well when the protocol is invoked sequentially.

Compiler $\mathsf{Comp}^{\mathsf{IC}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{es}}, \pi_{\mathsf{res}}, \pi_{\mathsf{ole}}, N, r)$

Below is the code for party P_i. The sequential composition technique must be applied to Procedure 2 (Section 5.2 extends it to the Procedure 1). To support δ-shift in place of Property (Q2), replace $r+1$ by $r+\delta$ in Step 2.
Message labelling: every message of the copy of π_{rand} indexed (l, j) in iteration ι carries the label $\lambda := (\mathsf{sid}, \iota, l, j)$, and every message of the invocation of π_{res} at Step $s \in \{3, 8, 10\}$ of iteration ι carries $\lambda := (\mathsf{sid}, \iota, s)$. Each label also serves as the session identifier of its copy or invocation, so that the signatures and VRF evaluations made in it are bound to it.

Let $m = \lceil \log N \rceil$, and for input $(\mathtt{input}, \mathsf{sid}, (v_1, \ldots, v_N))$, initialize $u_{l,j} := v_j$ for all $l \in [m], j \in [N]$, and set $\mathit{ES\text{-}terminated}_j := 0$ and $\mathit{ES\text{-}output}_j := \perp$ for all $j \in [N]$.

Procedure 1: Execution of the early-stopping fallback π_{es}

1. For every $j \in [N]$, send input v_j to a fresh copy of π_{es}.
2. Upon receiving output v from the jth copy of π_{es}, set $\mathit{ES\text{-}terminated}_j := 1$ and $\mathit{ES\text{-}output}_j := v$.

Procedure 2: Adjusted Execution of the modified Ben-Or and El-Yaniv

Repeat the following for iteration $\iota = 1, 2, \ldots$:

1. If $\mathit{ES\text{-}terminated}_j = 1$ for every $j \in [N]$, then for every $l \in [m]$ and $j \in [N]$, reset $u_{l,j} := \mathit{ES\text{-}output}_j$.
2. For every $l \in [m]$ and $j \in [N]$, run π_{rand} on input $u_{l,j}$ for exactly $r+1$ rounds:
   - Let $c_{l,j}$ denote the output after r rounds, or $\perp^{\mathsf{run}}$ if that copy has not produced an output by then.
   - Let $\hat{c}_{l,j}$ be the output after $r+1$ rounds, or $\perp^{\mathsf{run}}$ if none exists by then.
   - For every $j \in [N]$
     - define $S_j := \{c_{l,j}\}_{l=1}^{m} \setminus \{\perp^{\mathsf{run}}\}$ and $\hat{S}_j := \{\hat{c}_{l,j}\}_{l=1}^{m} \setminus \{\perp^{\mathsf{run}}\}$, the sets of values actually decided by round r and by round $r+1$.
     - if there exists a value v such that $c_{l,j} = v$ for all $l \in [m]$ — in particular no copy returned $\perp^{\mathsf{run}}$ — then set $c_j := v$; otherwise, set $c_j := \perp^{\mathsf{agr}}$.
3. If $c_j \neq \perp^{\mathsf{agr}}$ for all $j \in [N]$, send input 1 to π_{res}; otherwise, send input 0.
4. If π_{res} outputs 1, for every $j \in [N]$ choose some $o_j \in \hat{S}_j$ and output $(\mathtt{output}, \mathsf{sid}, (o_1, \ldots, o_N))$ and terminate. (The proof shows that in this case $\hat{S}_j$ is a singleton at every honest party, so the choice is forced.)
5. If $S_j \neq \emptyset$ for every $j \in [N]$, then for every $j \in [N]$ choose $d_j \in S_j$ and send $(d_1, \ldots, d_N)$ to all parties; otherwise, send $\perp^{\mathsf{msg}}$ to all parties.

[16] The blowup factor for a slack of c rounds is $2c+1$ here, compared with $3c+1$ in [CCGZ16]. The extra c rounds there arise from their use of canonical synchronous functionalities, which require a round for parties to submit their inputs before the message-exchange window begins. Our accounting avoids this step, so a window of $2c+1$ rounds suffices.

6. Call π_{ole} to choose a leader.
7. Let ℓ be the output of π_{ole} and P_ℓ the leader for this iteration, and let D be the value received from P_ℓ in the previous step. If no proper message is received, set $D := \perp^{\mathsf{msg}}$.
8. Send input D to π_{res}, whose domain for this invocation is $(\mathcal{M} \cup \{\perp\})^N \cup \{\perp^{\mathsf{msg}}\}$.
9. Let E be the output of π_{res}. If E is of the form $(e_1, \ldots, e_N)$ and $e_j \in \hat{S}_j$ for every $j \in [N]$, set $\mathsf{term} := 1$; otherwise, set $\mathsf{term} := 0$.
10. Send input term to π_{res}.
11. Let $\mathsf{terminate}$ denote the output of π_{res}. If $\mathsf{terminate} = 1$, for every $j \in [N]$ set $o_j := e_j$ and output $(\mathtt{output}, \mathsf{sid}, (o_1, \ldots, o_N))$ and terminate.

Figure 3: Compiler for expected-constant-round parallel BA with early stopping, from a unanimity-responsive randomized Byzantine agreement protocol π_{rand}, an early-stopping Byzantine agreement protocol π_{es}, a randomized early-stopping Byzantine agreement protocol π_{res}, and a leader-election protocol π_{ole}.

Theorem 4.2. *Fix a security flavor $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. Let π_{es} be a $\star$-secure early-stopping Byzantine agreement protocol tolerating up to t_{es} corruptions and terminating in $c_1 f + c_2$ rounds in the worst case, where c_1 and c_2 are constants and f is the actual number of corruptions. Let π_{rand} be a $\star$-secure unanimity-responsive randomized Byzantine agreement protocol (Definition 4.1) tolerating up to t_{rand} corruptions. Let π_{res} be a $\star$-secure randomized early-stopping Byzantine agreement protocol tolerating up to t_{res} corruptions, with expected round complexity e_{res} and worst-case round complexity $c_{1,\mathsf{res}} f + c_{2,\mathsf{res}}$, where $c_{1,\mathsf{res}}$ and $c_{2,\mathsf{res}}$ are constants. Let π_{ole} be a $\star$-secure leader-election protocol with fairness p and round complexity R_{ole}, tolerating up to t_{ole} corruptions, against the same adversary class as the other components, and set $t := \min\{t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}}\}$. Assume moreover that either $t < \frac{n}{3}$, or that $\star = \mathsf{comp}$, $t < \frac{n}{2}$, and an EUF-CMA signature scheme and a signature PKI are available for the implicit termination-synchronization rounds after each invocation of π_{res}.*

Then $\pi_{\mathsf{pba}}^{N,r} := \mathsf{Comp}^{\mathsf{IC}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{es}}, \pi_{\mathsf{res}}, \pi_{\mathsf{ole}}, N, r)$, with $r \geq \max(c, 2e)$, and c, e are the constants of π_{rand} in Def. 4.1, is a $\star$-secure N-parallel BA protocol against $\leq t$ corruptions and the same adversary class. Its expected and worst-case round complexities are $\frac{4}{p}(3(r + 2 + R_{\mathsf{ole}} + 3e_{\mathsf{res}}) + 3)$ and $c_1 f + c_2 + 3(2r + 2 + R_{\mathsf{ole}} + 4(c_{1,\mathsf{res}} f + c_{2,\mathsf{res}})) + 6$, respectively where $f \leq t$ is the actual corruption count.

Proof. We prove validity, agreement, termination and round complexity separately. Consider executions with at most $t = \min\{t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}}\}$ corruptions. We read "honest" as in the proof of Theorem 3.3, which also covers strongly adaptive adversaries. Beyond its components and the termination-synchronization layer, the compiler sends only the candidate vectors of Step 5, and only the leader's matters.

One observation is used twice below. Suppose the invocation of π_{res} in Step 3 outputs 1. By its validity some honest party P_h had input 1, so for every coordinate $j \in [N]$ there is a value w_j with $c_{l,j} = w_j$ for all $l \in [m]$ at P_h, and in particular no copy of coordinate j returned $\perp^{\mathsf{run}}$ there. By Property (Q2) every honest party obtains an output from each of these copies by round $r + 1$, and by the agreement of π_{rand} that output is again w_j. Hence $\hat{S}_j = \{w_j\}$ at every honest party, $\perp^{\mathsf{run}}$ being excluded from $\hat{S}_j$ by definition, and all honest parties output the same vector $(w_1, \ldots, w_N)$ in Step 4.

Validity. We prove validity coordinate-wise. Fix any coordinate $k \in [N]$, and assume that all honest parties start with the same input value v_k in coordinate k. The inputs in the other coordinates may be arbitrary.

We first observe that throughout the execution, every invocation of π_{rand} corresponding to

coordinate k is started by all honest parties from the same value v_k. Initially, this holds because $u_{l,k} = v_k$ for every $l \in [m]$. Later, if all copies of the fallback procedure π_{es} terminate, then Step 1 resets $u_{l,k} := \mathit{ES\text{-}output}_k$ for every $l \in [m]$. By the validity of π_{es}, since all honest parties started the kth copy with input v_k, we have $\mathit{ES\text{-}output}_k = v_k$. Hence the invariant holds.

Now consider any iteration of Procedure 2. Since all honest parties start every invocation of π_{rand} for coordinate k with the same value v_k, Property (Q3) implies that every such invocation outputs v_k within c rounds. As $r \geq c$, for every honest party and every $l \in [m]$, $c_{l,k} = v_k$ and $\hat{c}_{l,k} = v_k$. Thus $S_k = \hat{S}_k = \{v_k\}$ at every honest party.

Outputting in Step 4. By the observation above, all honest parties output the same vector $(w_1, \ldots, w_N)$ with $\hat{S}_j = \{w_j\}$, and since $\hat{S}_k = \{v_k\}$ we have $w_k = v_k$. Thus validity holds if the protocol terminates in Step 4.

Outputting in Step 11. Now suppose the protocol terminates in Step 11. Let $E = (e_1, \ldots, e_N)$ be the output of the multi-valued invocation of π_{res} in Step 8. Since $\mathsf{terminate} = 1$, by the validity of π_{res}, there exists at least one honest party P that sets its term to one in Step 9. By definition of term, at that honest party, for every $j \in [N]$, $e_j \in \hat{S}_j$. Now for index k, as discussed above, we know that $\hat{S}_k = \{v_k\}$ at every honest party, so the kth output is v_k, as required.

Since k was arbitrary, the protocol satisfies coordinate-wise validity.

Agreement. The final output is generated by Procedure 2. Any such output occurs either in Step 4 or in Step 11. Moreover, since the decision to output in either of these steps is determined via an invocation of π_{res}, the agreement of π_{res} ensures that all honest parties output in the same step.

Outputting in Step 4. By the observation above, all honest parties output the same vector $(w_1, \ldots, w_N)$.

Outputting in Step 11. In this case, the output vector is $(e_1, \ldots, e_N)$, which is the output of the multi-valued invocation of π_{res}. By the agreement property of π_{res}, all honest parties obtain the same vector $(e_1, \ldots, e_N)$ and therefore output the same value.

Therefore, in both cases, all honest parties that output produce the same vector, and the agreement property holds.

Termination and round complexity. We first bound the expected running time of Procedure 2. As in the analysis of Ben-Or and El-Yaniv [BOEY03], we show that each iteration ends the procedure with constant probability and has constant expected length.

Fix an iteration. Let E_1 be the event that for every coordinate $j \in [N]$ some invocation of π_{rand} outputs by round r, that is, every set S_j is nonempty after Step 2. Let E_2 be the event that the leader elected in Step 6 is honest at the start of the leader-election invocation.

If both events occur, Procedure 2 terminates in that iteration. The leader, honest at the start of Step 6, was honest throughout Step 5 as well, corruption being irrevocable, and by E_1 it sent there a vector $(c_1, \ldots, c_N)$ with each $c_j \in S_j$. Every honest party therefore takes this vector as D in Step 7, whatever the adversary does to the leader afterwards. Each c_j was output by round r by some copy of π_{rand} at an honest party, so by Property (Q2) and the agreement of π_{rand} it lies in $\hat{S}_j$ at every honest party. As all honest parties input D to the invocation of π_{res} in Step 8, its validity gives them all $E = D$, so $e_j = c_j \in \hat{S}_j$ for every j. All of them then set $\mathsf{term} := 1$ in Step 9, and the invocation of π_{res} in Step 10 makes them all terminate.

Both events are likely. By Property (Q1) and Markov's inequality, a single invocation of π_{rand} terminates within $2e \leq r$ rounds with probability at least $\frac{1}{2}$. The m invocations of a coordinate run on separate randomness and are kept apart by the message labels of Figure 3, so all of them

miss that deadline with probability at most 2^{-m}, that is, $\Pr[S_j = \emptyset] \le 2^{-m} \le \frac{1}{N}$. Over the N coordinates this gives $\Pr[E_1] \ge (1 - \frac{1}{N})^N \ge \frac{1}{4}$ for $N \ge 2$. The event E_1 is determined before the leader election begins, and its bound holds whatever the adversary does in Step 2, even knowing something about the election to come. That knowledge only lets it make E_1 likelier when the election is bad, which does not help it, so an iteration ends the procedure with probability at least $\alpha := p(1 - \frac{1}{N})^N \ge \frac{p}{4}$. For the same reason the bound on E_1 combines with the fairness of the separate calls to π_{ole}, and k iterations all fail with probability at most $(1 - \alpha)^k$.

So the number of iterations is dominated by a geometric random variable with parameter α and has expectation at most $\frac{1}{\alpha} \le \frac{4}{p}$. Every round of Procedure 2 is executed as a window of three rounds and each invocation of π_{res} is followed by one plain layer round, so an iteration costs in expectation

$$\underbrace{3(r+1)}_{\text{Step 2}} + \underbrace{3}_{\text{Step 5}} + \underbrace{3R_{\mathsf{ole}}}_{\text{Step 6}} + \underbrace{3 \cdot 3e_{\mathsf{res}} + 3}_{\text{Steps 3, 8, 10}} = 3(r + 2 + R_{\mathsf{ole}} + 3e_{\mathsf{res}}) + 3$$

rounds, an invocation of π_{res} having expected round complexity at most e_{res} whatever its domain. Procedure 2 therefore terminates within $\frac{4}{p}(3(r + 2 + R_{\mathsf{ole}} + 3e_{\mathsf{res}}) + 3)$ rounds in expectation, and $\pi^{N,r}_{\mathsf{pba}}$ outputs as soon as it does, which is the claimed expectation.

Now we move on to the worst-case analysis. Let $c_{\mathsf{res}}(f) := c_{1,\mathsf{res}} f + c_{2,\mathsf{res}}$ be the worst-case round complexity of an invocation of π_{res}. The per-iteration count above, with e_{res} replaced by $c_{\mathsf{res}}(f)$, shows that an iteration of Procedure 2 takes at most $C_{\mathrm{iter}} := 3(r + 2 + R_{\mathsf{ole}} + 3c_{\mathsf{res}}(f)) + 3$ rounds.

Since the N copies of π_{es} in Procedure 1 run in parallel, and each copy terminates within $c_1 f + c_2$ rounds in the worst case, by round $c_1 f + c_2$ every honest party has $\mathit{ES\text{-}terminated}_j = 1$ and holds the same fallback output value $\mathit{ES\text{-}output}_j$ for every $j \in [N]$.

Every honest party performs the next execution of Step 1 of Procedure 2 within at most $C_{\mathrm{iter}} - 1$ rounds from this point, and resets $u_{l,j} := \mathit{ES\text{-}output}_j$ for all $l \in [m]$, $j \in [N]$ there. The honest parties need not be in the same iteration when they do so, since the layer keeps them only within one round of each other and an iteration boundary may fall between them. This costs nothing, because the guard of Step 1 is re-evaluated in every iteration and holds from here on, so once an honest party has reset it enters every later iteration with $u_{l,j} = \mathit{ES\text{-}output}_j$. Let ι^* be the largest of the honest parties' first post-reset iteration indices. Every honest party executes iteration ι^*, and every honest party enters its Step 2 with $u_{l,j} = \mathit{ES\text{-}output}_j$: the last to reset does so in ι^* itself, and the others reset earlier and keep the value.

Hence, for every coordinate $j \in [N]$, all m invocations of π_{rand} in Step 2 start from the same value $\mathit{ES\text{-}output}_j$. By Property (Q3), each such invocation outputs $\mathit{ES\text{-}output}_j$ by round r, and therefore for every $l \in [m]$ and $j \in [N]$ we have $c_{l,j} = \mathit{ES\text{-}output}_j$; in particular no copy returns $\perp^{\mathsf{run}}$. Consequently $c_j = \mathit{ES\text{-}output}_j \neq \perp^{\mathsf{agr}}$ for every $j \in [N]$. Here it matters that the sentinel $\perp^{\mathsf{agr}}$ is distinct from the domain element $\perp$, which $\mathit{ES\text{-}output}_j$ may be, since the reduction of Section 4.2 feeds $\perp$ into the coordinate of a silent sender; otherwise an honest party would read a unanimous coordinate as a failure and send 0 in Step 3.

Thus, in Step 3 every honest party sends input 1 to the (binary) invocation of π_{res}. By the validity of π_{res}, the output in Step 4 is 1 for all honest parties, and by the observation at the start of the proof they all output the same vector in Step 4, namely $(\mathit{ES\text{-}output}_1, \ldots, \mathit{ES\text{-}output}_N)$.

It follows that, after all copies of π_{es} have terminated, the protocol requires at most $(C_{\mathrm{iter}} - 1) + (3(r+1+c_{\mathsf{res}}(f))+1)$ additional rounds before termination, both counted in expanded rounds: the wait for the next Step 1 is bounded by one full iteration less one round, while the tail consists of the $3(r + 1)$ rounds of Step 2, the $3c_{\mathsf{res}}(f)$ rounds of the Step-3 invocation of π_{res}, and the one

plain round of the termination-synchronization layer that follows it. This sums up to

$$\begin{aligned} & c_1 f + c_2 + 3\big(2r + 2 + R_{\mathsf{ole}} + 4c_{\mathsf{res}}(f)\big) + 6 \\ & \quad = \; c_1 f + c_2 + 3\big(2r + 2 + R_{\mathsf{ole}} + 4(c_{1,\mathsf{res}} f + c_{2,\mathsf{res}})\big) + 6. \end{aligned}$$

The argument uses Property (Q2) only to carry an output that one honest party holds by round r to every honest party by the round at which $\hat{S}_j$ is read. Hence, if π_{rand} only guarantees that honest parties terminate within some constant $\delta \geq 1$ rounds of each other, the same proof goes through with $r+1$ replaced by $r+\delta$ in the compiler and in the round counts above, which gives expected round complexity

$$\frac{4}{p}\big(3(r + \delta + 1 + R_{\mathsf{ole}} + 3e_{\mathsf{res}}) + 3\big)$$

and worst-case round complexity

$$c_1 f + c_2 + 3\big(2r + 2\delta + R_{\mathsf{ole}} + 4(c_{1,\mathsf{res}} f + c_{2,\mathsf{res}})\big) + 6.$$

□

Corollary 4.3. *There exist N-parallel Byzantine agreement protocols with expected round complexity $O(1)$ and worst-case round complexity $O(f)$, where $f \leq t$ is the actual number of corruptions, with the following security guarantees:*

- *perfect security against up to t corruptions, using no setup, for $t < \frac{n}{3}$; and*
- *computational security against up to t corruptions, in the signature-PKI and VRF-PKI model, for $t < \frac{n}{2}$, assuming EUF-CMA signature schemes and VRF schemes with exponential-size range.*

Proof. See Corollaries C.1 and C.3 in Appendix C. □

At $t < \frac{n}{3}$ the constants improve considerably if one is willing to pay for a setup, by instantiating the same theorem with the signature-based protocol of [Mic17] in place of the setup-free components; see Corollary C.2 in Appendix C.

4.2 From Parallel BA to Interactive Consistency

Interactive consistency (Definition 2.5), which a broadcast-hybrid MPC protocol consumes, follows from parallel Byzantine agreement with one additional round, in which every party sends its input to all parties before they run the parallel protocol on the received values (Lemma A.2 in Appendix A). Applied to the protocols of Corollary 4.3, this gives the following.

Corollary 4.4. *There exist interactive-consistency protocols for any domain $\mathcal{M}$ with expected round complexity $O(1)$ and worst-case round complexity $O(f)$, where $f \leq t$ is the actual number of corruptions, with the following guarantees:*

- *perfect security against up to t corruptions, using no setup, for $t < \frac{n}{3}$; and*
- *computational security against up to t corruptions, in the signature-PKI and VRF-PKI model, for $t < \frac{n}{2}$, assuming EUF-CMA signature schemes and VRF schemes with exponential-size range.*

Proof. Apply Lemma A.2 to the protocols of Corollary 4.3; the single prepended round leaves both bounds in the same classes. □

5 Randomized Early-Stopping MPC

5.1 From Property-Based to Simulation-Based IC

The guarantees of Corollary 4.4 are the property-based ones of Definition 2.5, whereas MPC protocols are analyzed in the broadcast-hybrid (equivalently, IC-hybrid) model, where the hybrid is an *ideal computation* and security is argued by simulation. Bridging the two is essentially free: any protocol meeting Definition 2.5 also securely realizes the ideal IC computation of Def. 2.8 (described at a high level in Sec. 2.1.2), in the stand-alone model against static adversaries. Against adaptive adversaries the two come apart [HZ10, GKKZ11], and it remains an open question to show whether our protocol is secure against adaptive corruptions.

Lemma 5.1 (Property-based IC implies simulation-based IC)**.** *Fix $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. Let π be a $\star$-secure IC protocol over $\mathcal{M}$ under the property-based Definition 2.5, against static, rushing, malicious adversaries corrupting up to t parties. Assume that π has an a-priori worst-case round bound, polynomial in κ when $\star = \mathsf{comp}$. Then π $\star$-securely realizes the ideal interactive-consistency computation of Definition 2.8 against the same adversary class and corruption threshold. The simulator is straight-line and black-box in the adversary.*

Proof sketch. Fix a static adversary $\mathcal{A}$, a corruption set $I \subseteq [n]$ with $|I| \leq t$, an input vector $\boldsymbol{x}$, and an auxiliary input z. We describe the simulator $\mathcal{S}$, which runs $\mathcal{A}$ in a straight-line, black-box manner.

The simulator.

1. *Initialization.* $\mathcal{S}$ receives from the trusted party the inputs m_i of every honest party P_i, $i \notin I$. It then emulates the setup of π (if any) exactly as the real registration phase does, which is not the same for the two setups this paper uses. For the signature PKI it samples $(\mathsf{sk}_i, \mathsf{pk}_i)$ for the honest parties only, hands $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ restricted to the honest indices to $\mathcal{A}(z)$, and lets $\mathcal{A}$ choose and register the corrupted parties' verification keys, as Section 2.1 permits it to do. For the VRF-PKI it samples every key pair itself, since there the setup samples all of them, and hands each corrupted party its own vsk_i along with the public $\mathsf{vpk}_1, \ldots, \mathsf{vpk}_n$. It then invokes $\mathcal{A}$ on that setup together with the public information $\mathcal{A}$ would see at the onset of a real execution. Emulating the signature PKI the way the VRF-PKI is emulated would be wrong: it would hand $\mathcal{A}$ keys it did not choose, and the claim of identical distribution below would fail precisely for the PKI-based instantiations for which this lemma is needed.

2. *Internal emulation.* $\mathcal{S}$ runs a complete internal execution of π: it plays each honest party P_i, $i \notin I$, by running π's code honestly on the true input m_i with fresh randomness, while $\mathcal{A}$ controls the parties in I. Rounds are emulated exactly as in the real execution; in particular, in each round $\mathcal{S}$ first hands $\mathcal{A}$ the messages sent by the honest parties in that round and only then collects the messages of the corrupted parties, so a rushing $\mathcal{A}$ is emulated faithfully.

3. *Output extraction.* If every internally emulated honest party produces an output, let $\boldsymbol{o} = (o_1, \ldots, o_n)$ denote the output vector of the honest party with the smallest index; if the emulation instead reaches the a-priori worst-case round bound of π without all of them having produced one, let $\boldsymbol{o}$ be the all-$\perp$ vector. In either case $\mathcal{S}$ sends $m'_j := o_j$ to the trusted party on behalf of every corrupted sender P_j, $j \in I$. The second branch is what keeps $\mathcal{S}$ well defined when termination fails; it is taken only on the event $\neg E$ below, whose probability the analysis already charges.

4. *Output.* $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs in the internal execution and halts.

Analysis. The crucial observation is that, because the ideal computation leaks all honest inputs to $\mathcal{S}$ and because I is fixed before the execution begins, $\mathcal{S}$ is in a position to run the honest parties' code *on their real inputs, from the first round on.* Consequently, the internal execution in Step 2 is not merely indistinguishable from, but is *identically distributed to*, a real execution $\text{REAL}_{\pi,I,\mathcal{A}(z)}(\boldsymbol{m}, \kappa)$ of π with adversary $\mathcal{A}$, corruption set I, inputs $\boldsymbol{m}$, and auxiliary input z: the honest parties' messages are produced by the same code on the same inputs with identically distributed randomness, and $\mathcal{A}$'s interface is identical in the two experiments. In particular, the view of $\mathcal{A}$, and hence $\mathcal{S}$'s output, is distributed exactly as $\mathcal{A}$'s output in the real execution.

It remains to argue that the honest parties' *outputs* match as well. Let E be the event that the internal execution satisfies the three properties of Definition 2.5. Since that execution is distributed as a real execution of π against a static, rushing, malicious t-adversary, $\Pr[\neg E]$ is exactly the failure probability $\varepsilon(\kappa)$ named in the statement. Condition on E and consider the three properties in turn.

- *Termination* guarantees that every internally emulated honest party eventually outputs, so Step 3 is reached and $\boldsymbol{o}$ is well defined.
- *Agreement* guarantees that all internally emulated honest parties output the same vector. Hence $\boldsymbol{o}$ does not depend on the (arbitrary) choice of the smallest-index honest party in Step 3, and $\boldsymbol{o}$ equals the common output vector of the honest parties in the corresponding real execution.
- *Validity* guarantees that $o_j = m_j$ for every honest sender P_j, $j \notin I$.

Now compare $\boldsymbol{o}$ with the vector $\boldsymbol{y}$ that the trusted party delivers in the ideal execution. For an honest sender P_j the trusted party sets $y_j = m_j$, and validity gives $o_j = m_j$, so $y_j = o_j$. For a corrupted sender P_j the trusted party sets $y_j = m'_j = o_j$ by the simulator's choice in Step 3. Hence $\boldsymbol{y} = \boldsymbol{o}$, i.e., the honest parties in the ideal execution output exactly the vector that the honest parties output in the emulated real execution.

Therefore, conditioned on E, the joint distribution of ($\mathcal{A}$'s output, honest parties' outputs) is the same in the real and in the ideal execution, and the two ensembles differ by at most $\Pr[\neg E] \leq \varepsilon(\kappa)$. This yields perfect emulation when $\varepsilon = 0$, statistical closeness when ε is negligible against unbounded adversaries, and computational indistinguishability when the properties hold only against PPT adversaries (as for our PKI-based instantiations), where $\mathcal{S}$ is PPT whenever $\mathcal{A}$ is. □

Instantiating Lemma 5.1 with the protocols of Corollary 4.4 gives the following.

Corollary 5.2. *The interactive-consistency protocols of Corollary 4.4 securely realize the ideal interactive-consistency computation in the stand-alone model against static, rushing, malicious adversaries, with the following guarantees: the setup-free protocol is perfectly t-secure for $t < \frac{n}{3}$, whereas the signature-PKI and VRF-PKI protocol is computationally t-secure for $t < \frac{n}{2}$, assuming EUF-CMA signature schemes and VRF schemes with exponential-size range.*

Remark 5.3 (Running time of the simulator)**.** *Definition 2.5 only requires* eventual *termination, so in general the simulator's internal emulation need not terminate within a strictly polynomial number of rounds. This is a non-issue for our protocols since by Corollary 4.4 they have a worst-case round bound that is linear in $f \leq t < n$, so $\mathcal{S}$ is strictly PPT.*

5.2 Reactive IC and Sequential Composition

A single invocation of IC is not what an MPC protocol consumes. An R-round protocol in the broadcast model performs R *sequential* rounds of broadcast, each of which is one IC invocation, with the inputs to invocation s computed from the outputs of invocation $s-1$. The object to realize is therefore the reactive ideal computation of Section 2.1, rather than R independent copies of the

single-shot one. This lets us handle PKI reuse across invocations and the nuances of sequential composition separately from the incorporation into the MPC protocol.

Sequential composition of IC invocations. Canetti's modular composition theorem [Can00] realizes such a reactive IC by sequential invocations of an IC protocol, but requires them to be strictly sequential, with no honest party starting invocation $s+1$ before every honest party has concluded invocation s. With probabilistic termination there is no a-priori round by which this has happened.

We therefore apply the technique of Section 4 one level up [LLR02, CCGZ16]. The randomized procedure (Procedure 2) of the protocol of Corollary 4.4 is already expanded internally, and it already ends with one round of the termination-synchronization layer, after the last invocation of π_{res}.

Two other parts of that protocol are not expanded: the N parallel copies of the early-stopping fallback π_{es} in Procedure 1, and the single round that Lemma A.2 prepends to turn parallel Byzantine agreement into interactive consistency. Both are analyzed under a simultaneous start, which is exactly what a standalone invocation gives them and what the second and later invocations here do not. We therefore expand them here rather than in Section 4, so that the bounds proved there, where the protocol is invoked once, are not inflated by a cost that only sequential composition incurs. Each round of π_{es} and the prepended round are executed as windows of three rounds in the same way, which replaces the $c_1 f + c_2$ term of Theorem 4.2 by $3(c_1 f + c_2)$ and the prepended round by three.

With every round of an invocation now expanded, adding the termination-synchronization layer between invocations[17] lets us use [LLR02, CCGZ16] off the shelf. These two modifications turn the IC protocol of Corollary 4.4 into a reactive IC protocol for any polynomial number R of sequential sessions. Note that since all our instantiations for IC uses at most signature and VRF, using standard domain separation techniques we can use the same setup for all R invocations.

Write E_{ic} and $W_{\mathsf{ic}}(f)$ for the expected and worst-case round complexities of the underlying IC protocol, and $c_1 f + c_2$ for the worst-case round complexity of its early-stopping fallback. Then each session of the reactive IC costs

$$E_{\mathsf{ric}} = E_{\mathsf{ic}} + 3, \qquad W_{\mathsf{ric}}(f) = W_{\mathsf{ic}}(f) + 2(c_1 f + c_2) + 3. \tag{5.1}$$

Corollary 5.4. *For any polynomial number R of sequential sessions, there exist reactive interactive-consistency protocols in the stand-alone model against static, rushing, malicious adversaries, each session of which has expected round complexity $O(1)$ and worst-case round complexity $O(f)$, where $f \le t$ is the actual number of corruptions, with the following guarantees:*

- *a perfectly t-secure protocol, using no setup, for $t < \frac{n}{3}$; and*
- *a computationally t-secure protocol, in the signature-PKI and VRF-PKI model, for $t < \frac{n}{2}$, assuming EUF-CMA signature schemes and VRF schemes with exponential-size range.*

Proof. By (5.1), applied to the protocols of Corollary 4.4, both per-invocation bounds stay in the same classes, since $c_1 f + c_2$ is itself linear in f. Corollary C.4 in Appendix C performs the substitution and records the resulting constants. □

[17] Expanding rounds is not by itself enough, because it only lets an invocation *absorb* the slack it starts with; it does not control the slack it leaves behind. This causes the expansion factor to grow exponentially in the number of sessions.

5.3 From Reactive IC to MPC over Point-to-Point Channels

An R-round MPC protocol in the broadcast model is nothing but a protocol in the reactive-IC-hybrid model. Its s-th broadcast round is the s-th session of the reactive ideal computation of Section 2.1. Canetti's modular composition theorem [Can00] therefore lets us replace that hybrid by any protocol realizing it, and Corollary C.4 supplies such protocols over point-to-point channels. This recipe has already been used in the literature to achieve MPC over point-to-point channels for different goals or settings [CCGZ17, CCF+26]. All that is left is to say which broadcast-model MPC protocols we may start from.

Rushing suffices. The one point that requires care is that our ideal computation for interactive consistency is not the "megaphone" functionality: the trusted party hands the simulator the messages of all honest senders *before* the simulator fixes the messages of the corrupted ones. It thus offers no input independence, and a broadcast-model protocol whose analysis relies on the corrupted parties committing to their round-s broadcast before seeing the honest ones is not covered. This is no restriction in practice: exactly the same power is enjoyed by a rushing adversary against a broadcast round, and MPC protocols are routinely analyzed against rushing adversaries.

Corollary 5.5. *Fix $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$, and g an n-party functionality. Let π_{bc} be an R-round protocol that $\star$-securely computes g in the broadcast model, in the stand-alone setting against static, rushing, malicious adversaries corrupting up to t parties. Let π_{ric} $\star$-securely realize the reactive ideal interactive-consistency computation against the same adversary class and corruption threshold, with expected round complexity E and worst-case round complexity $W(f)$ per invocation when $f \le t$ parties are corrupted. Then there exists a protocol π_{p2p} that $\star$-securely computes g over point-to-point channels alone, in the same setting and against the same adversary class and corruption threshold, with expected round complexity at most $R \cdot E$ and worst-case round complexity at most $R \cdot W(f)$.*

Proof sketch. By the discussion above, π_{bc} is a protocol in the reactive-IC hybrid model making R sequential calls to the ideal computation of Section 2.1, so replacing those calls by π_{ric} preserves security by [Can00]. As for the rounds, the R invocations are performed one after the other and each costs at most E rounds in expectation and $W(f)$ in the worst case, so the bounds add up. □

Combining Corollary 5.4 with a constant-round base MPC protocol gives our final result. In the computational setting, constant-round MPC with guaranteed output delivery in the broadcast model is known for both $t < \frac{n}{3}$ [DI05] and $t < \frac{n}{2}$ [BMR90].

Corollary 5.6. *Let g be an n-party functionality computable with guaranteed output delivery by a constant-round broadcast-model protocol, computationally secure in the stand-alone setting against static, rushing, malicious adversaries corrupting up to t parties. Then g is computable over point-to-point channels alone, with the same security, adversary class and threshold, in $O(1)$ rounds in expectation and $O(f)$ in the worst case for $f \le t$ corruptions, with no additional setup for $t < \frac{n}{3}$, and for $t < \frac{n}{2}$ in the signature-PKI and VRF-PKI model assuming EUF-CMA signatures and VRFs with exponential-size range. The base protocols cited above additionally assume a pseudorandom generator.*

Bibliography

[ACD+19] Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the*

2019 ACM Symposium on Principles of Distributed Computing, PODC '19, page 317–326, New York, NY, USA, 2019. Association for Computing Machinery.

[AD15] Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 605–614, 2015.

[ADD+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected o (1) rounds, expected communication, and optimal resilience. In *International Conference on Financial Cryptography and Data Security*, pages 320–334. Springer, 2019.

[AKP20] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st FOCS*, pages 1277–1284, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press.

[AKP23] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of statistical MPC with optimal resiliency. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 1527–1536, Orlando, FL, USA, June 20–23, 2023. ACM Press.

[BG93] Piotr Berman and Juan A Garay. Cloture votes: n/4-resilient distributed consensus in t+ 1 rounds. *Mathematical systems theory*, 26(1):3–19, 1993.

[BGP92] Piotr Berman, Juan A Garay, and Kenneth J Perry. Optimal early stopping in distributed consensus. In *International Workshop on Distributed Algorithms*, pages 221–237. Springer, 1992.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BOEY03] Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[Bra87] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and computation*, 75(2):130–143, 1987.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[CCF+26] Ran Cohen, Daniel Collins, Pouyan Forghani, Juan Garay, and Vassilis Zikas. On the round complexity of dishonest-majority mpc. *Cryptology ePrint Archive*, 2026.

[CCGZ16] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[CCGZ17] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP 2017*, volume 80 of *LIPIcs*, pages 37:1–37:15, Warsaw, Poland, July 10–14, 2017. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

[CDG+25] Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. Repeated agreement is cheap! on weak accountability and multishot byzantine agreement. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 15–27, 2025.

[CDR+25] Michele Ciampi, Ivan Damgård, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Information-theoretic broadcast-optimal MPC. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part I*, volume 16268 of *LNCS*, pages 152–181, Aarhus, Denmark, December 1–5, 2025. Springer, Cham, Switzerland.

[CDvdG88] David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 87–119, Santa Barbara, CA, USA, August 16–20, 1988. Springer Berlin Heidelberg, Germany.

[CEG26] Pierre Civit, Grigorii Emdin, and Rachid Guerraoui. Constant-round interactive consistency in dishonest majority, or with quadratic communication in honest majority. *Cryptology ePrint Archive*, 2026.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 39–53, Taormina, Sicily, Italy, March 19–21, 2012. Springer Berlin Heidelberg, Germany.

[CL17] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, October 2017.

[CMS89] Benny Chor, Michael Merritt, and David B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *Journal of the ACM*, 36(3):591–614, 1989.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394, Santa Barbara, CA, USA, August 14–18, 2005. Springer Berlin Heidelberg, Germany.

[DKLZ25] Giovanni Deligios, Ivana Klasovita, and Chen-Da Liu-Zhang. Asymptotically optimal early termination for dishonest majority broadcast. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 116–144. Springer, 2025.

[DMR+21] Ivan Damgård, Bernardo Magri, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with an honest majority. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 155–184, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[Dol82] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[DRS90] Danny Dolev, Ruediger Reischuk, and H Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM (JACM)*, 37(4):720–741, 1990.

[EGPS22] Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. TurboPack: Honest majority MPC with constant online communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 951–964, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

[EKL25] Fatima Elsheimy, Simon Holmgaard Kamp, and Julian Loss. Early stopping is cheap. *Cryptology ePrint Archive*, 2025.

[ELP24] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Early stopping byzantine agreement (1+epsilon)f rounds. In *Advances in Cryptology – ASIACRYPT 2024: 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9–13, 2024, Proceedings, Part VI*, page 398–424, Berlin, Heidelberg, 2024. Springer-Verlag.

[ELP25] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Towards optimal early stopping agreement protocols. *Cryptology ePrint Archive*, 2025.

[FLZL21] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. A new way to achieve round-efficient byzantine agreement. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 355–362, Virtual Event, Italy, July 26–30, 2021. ACM.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[FN09] Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In *International Symposium on Distributed Computing*, pages 449–463. Springer, 2009.

[GGLZ22] Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. Round-optimal byzantine agreement. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 96–119, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589, Crete, Greece, July 6–8, 2001. ACM Press.

[GKKO07] Juan A Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 658–668. IEEE, 2007.

[GKKZ11] Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In Cyril Gavoille and Pierre Fraigniaud, editors, *30th ACM PODC*, pages 179–186, San Jose, CA, USA, June 6–8, 2011. ACM.

[GM98] Juan A Garay and Yoram Moses. Fully polynomial byzantine agreement for n> 3 t processors in t+ 1 rounds. *SIAM Journal on Computing*, 27(1):247–290, 1998.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[GP90] Oded Goldreich and Erez Petrank. The best of both worlds: Guaranteeing termination in fast randomized byzantine agreement protocols. *Information Processing Letters*, 36(1):45–49, 1990.

[HOSS18] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 3–33, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 466–485, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

[KF99] Axel W Krings and Thomas Feyer. The byzantine agreement problem: optimal early stopping. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, pages 12–pp. IEEE, 1999.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462, Santa Barbara, CA, USA, August 20–24, 2006. Springer Berlin Heidelberg, Germany.

[KK09] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for Byzantine agreement. *Journal of Computer and System Sciences*, 75(2):91–112, 2009.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595, Warsaw, Poland, May 4–8, 2003. Springer Berlin Heidelberg, Germany.

[KY84] Anna R. Karlin and Andrew C. Yao. Probabilistic lower bounds for Byzantine agreement and clock synchronization. Unpublished manuscript, 1984.

[LLR02] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In Aleta Ricciardi, editor, *21st ACM PODC*, pages 203–212, Monterey, CA, USA, July 21–24, 2002. ACM.

[LN24] Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–488. Springer, 2024.

[LS22] Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 60–69, 2022.

[Mic17] Silvio Micali. Byzantine agreement , made trivial. 2017.

[OB08] Michael Okun and Amnon Barak. Efficient algorithms for anonymous byzantine agreement. *Theory of Computing Systems*, 42(2):222–238, 2008.

[SLM+23] Shravan Srinivasan, Julian Loss, Giulio Malavolta, Kartik Nayak, Charalampos Papamanthou, and Sri AravindaKrishnan Thyagarajan. Transparent batchable time-lock puzzles and applications to byzantine consensus. In *IACR International Conference on Public-Key Cryptography*, pages 554–584. Springer, 2023.

[TC84] Russell Turpin and Brian A Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[TPS85] Sam Toueg, Kenneth J Perry, and TK Srikanth. Fast distributed agreement (preliminary version). In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 87–101, 1985.

[WXSD20] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In *Theory of Cryptography Conference*, pages 381–411. Springer, 2020.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[Zam96] Arkady Zamsky. An randomized byzantine agreement protocol with constant expected time and guaranteed termination in optimal (deterministic) time. In *Proceedings of the fifteenth annual ACM symposium on principles of distributed computing*, pages 201–208, 1996.

[ZHDZ25] Xufeng Zhang, Baohan Huang, Sisi Duan, and Haibin Zhang. Randomized vs. deterministic? practical randomized synchronous bft in expected constant time. In *2025 44th International Symposium on Reliable Distributed Systems (SRDS)*, pages 24–35. IEEE, 2025.

A Constructions for Standard Building Blocks

This appendix collects the constructions we use for primitives that are standard or immediate, together with their proofs, so that they do not interrupt Section 2.

Parallel crusader agreement. Dolev's Algorithm CR [Dol82] is stated for general networks, where the sender's value travels along $2t+1$ disjoint paths and each receiver runs a purifying subroutine on the copies it collects. Over point-to-point channels between all pairs of parties both devices collapse, and running the n single-sender instances concurrently costs no more than running one. The protocol below is that specialization, and is the π_{pca} of Section 2.1.

Proposition A.1. *For $t < \frac{n}{3}$, the following two-round protocol is a perfectly secure parallel crusader agreement protocol over any message space $\mathcal{M}$, using no setup.*

- Round 1. *Each party P_j sends its input m_j to all parties.*
- Round 2. *Each party P_i sends to all parties the vector $(m_{1,i}, \ldots, m_{n,i})$, where $m_{j,i}$ is the value it received from P_j in round 1, or $\perp$ if it received none.*
- Output. *For each $j \in [n]$, party P_i sets $v_{i,j} := v$ if at least $n-t$ parties reported the same value $v \in \mathcal{M}$ as P_j's, and $v_{i,j} := \perp$ otherwise.*

Proof. Termination is immediate, and the output rule is well defined: two sets of at least $n-t$ reports intersect in at least $n-2t > t$ parties, hence in an honest one, and an honest party reports a single value for each sender, so no two distinct values can both reach the threshold in one party's view.

For honest-sender delivery, let P_j be honest. All $n-t$ honest parties receive m_j in round 1 and report it in round 2, so every honest P_i sees at least $n-t$ reports of m_j and, by the previous paragraph, of nothing else; hence $v_{i,j} = m_j$.

For crusader consistency, suppose honest parties P_i and $P_{i'}$ output $v \neq \perp$ and $v' \neq \perp$ in coordinate j. Then at least $n-t$ parties reported v to P_i and at least $n-t$ reported v' to $P_{i'}$; the two sets intersect in more than t parties, hence in an honest one, which sent the same report to both. Therefore $v = v'$. □

From parallel BA to interactive consistency. Interactive consistency (Definition 2.5), in which each party is a *sender* rather than the holder of one coordinate of a shared input vector, reduces to parallel Byzantine agreement with one additional round.

Lemma A.2. *Fix a security flavor $\star \in \{\mathsf{perf}, \mathsf{stat}, \mathsf{comp}\}$. Let π be a $\star$-secure N-parallel Byzantine agreement protocol with $N = n$ over the domain $\mathcal{M} \cup \{\perp\}$, against up to t corruptions, with expected round complexity E and worst-case round complexity $W(f)$, where $f \leq t$ is the actual number of corruptions. Then there is a $\star$-secure interactive-consistency protocol over $\mathcal{M}$, against up to t corruptions and the same adversary class, with expected round complexity $E+1$ and worst-case round complexity $W(f)+1$.*

Proof. In the first round, every party P_j sends its input m_j to all parties. The parties then invoke π with $N = n$, where P_i uses as its input in coordinate j the value it received from P_j in the first round, or $\perp$ if no valid value was received. Each party outputs the vector produced by π.

Termination and agreement are inherited coordinate-wise from π. For validity, let P_j be an honest sender. Then every honest party receives m_j in the first round and enters coordinate j of π with input m_j, so by the validity of π every honest party outputs m_j in that coordinate. The transformation adds exactly one round, and since that round is prepended, precedes every probabilistic-termination invocation and is executed by all honest parties simultaneously, it needs no expansion and affects neither bound beyond the additive 1, nor the termination gap of π. □

B A Binary-to-Multivalued Transformation for $t < \frac{n}{2}$

Section 4 needs to lift a binary Byzantine agreement protocol to a multivalued one at $t < \frac{n}{2}$. Both [FLZL21] and [GGLZ22] refer to such a transformation, at a cost of three rounds, citing the reduction of Turpin and Coan [TC84]; that reduction is given for $t < \frac{n}{3}$, and its consistency argument uses the threshold. We assemble the $t < \frac{n}{2}$ case here out of a building block of [GGLZ22], at the same cost of three rounds.

Conditional graded broadcast. The building block is a graded relaxation of broadcast, in which a sender distributes a value and each party attaches a grade to what it receives. We use it in the conditional form of [GGLZ22], where each party carries an additional bit controlling whether it speaks.

Definition B.1 (Conditional graded broadcast [GGLZ22, Definition 3])**.** *A protocol in which a designated sender P_s holds a value x, each party P_i holds a bit b_i, and each party outputs a pair (y_i, g_i) with $g_i \in \{0, 1, 2\}$, is a conditional graded broadcast protocol tolerating t corruptions if the following hold.*

- ***Conditional validity.*** *If P_s is honest and every honest party has $b_i = 1$, then every honest party outputs $(x, 2)$.*
- ***Conditional graded consistency.*** *For any two honest parties P_i and P_j we have $|g_i - g_j| \leq 1$, and if $g_i > 0$ and $g_j > 0$ then $y_i = y_j$.*
- ***No honest participation.*** *If every honest party has $b_i = 0$, then every honest party outputs $(\perp, 0)$.*

Theorem 2 of [GGLZ22] gives such a protocol for $t < \frac{n}{2}$ in three rounds, assuming a signature PKI and nothing further: the sender signs and sends its value, each party appends its own signature and echoes the pair, and each party then forwards the valid tuples it has received. We set $b_i = 1$ at every party, which makes the third property vacuous and the first an ordinary validity clause.

The transformation. Fix a default value $\phi \in \mathcal{M}$ known to all parties. Each party P_i holds an input $v_i \in \mathcal{M}$ and proceeds as follows.

- *Distribute.* Run n instances of conditional graded broadcast in parallel, the jth instance having P_j as sender with input v_j, and every party setting $b_i = 1$. Write $(u_{j,i}, g_{j,i})$ for P_i's output in instance j. The instances are independent and concurrent, so this costs three rounds.
- *Aggregate* (local computation). If some value v has $g_{j,i} = 2$ and $u_{j,i} = v$ in more than $\frac{n}{2}$ of the instances, set $(w_i, h_i) := (v, 2)$. Otherwise, if some v has $g_{j,i} > 0$ and $u_{j,i} = v$ in more than $\frac{n}{2}$ of them, set $(w_i, h_i) := (v, 1)$. Otherwise set $h_i := 0$ and $w_i := \phi$.
- *Agree.* Run the binary protocol on the bit $\beta_i := 1$ if $h_i = 2$ and $\beta_i := 0$ otherwise, and let β be its output. Output w_i if $\beta = 1$ and ϕ if $\beta = 0$.

The majority in the aggregation step is taken over instances rather than over parties, which is what makes the argument below work at $t < \frac{n}{2}$: two honest parties can disagree about a party, but not about an instance.

Lemma B.2. *Let $t < \frac{n}{2}$, assume an EUF-CMA signature scheme and a signature PKI, and let π_{bin} be a computationally secure binary Byzantine agreement protocol against up to t corruptions. Then the protocol above is a computationally secure Byzantine agreement protocol over any domain $\mathcal{M}$, against up to t corruptions and the same adversary class, whose round complexity is that of π_{bin} plus three.*

Proof. The round count is immediate: the three graded-broadcast rounds precede π_{bin} and are the same in every execution. The aggregation is well defined too, since two values each holding positive grade at P_i in more than $\frac{n}{2}$ instances would have index sets sharing an instance j, whereas P_i outputs a single pair in instance j; so at most one value qualifies, at either grade.

Everything else rests on one observation: if some honest P_i has $h_i = 2$ and $w_i = v$, then $w_k = v$ at every honest P_k. Let S be the set of instances in which P_i output $(v, 2)$, so $|S| > \frac{n}{2}$, and fix $j \in S$ and an honest P_k. Graded consistency gives $|g_{j,i} - g_{j,k}| \leq 1$, hence $g_{j,k} \geq 1$, and then $u_{j,k} = u_{j,i} = v$ since both grades are positive. So v has positive grade at P_k throughout S, the aggregation rule fires there, and by well-definedness the value it fires on is v.

Validity follows. If every honest party inputs v then, more than $\frac{n}{2}$ of the senders being honest, conditional validity gives $(v, 2)$ at every honest party in each of their instances; so every honest party sets $h_k = 2$ and $\beta_k = 1$, and the validity of π_{bin} returns $\beta = 1$ and output v. For agreement, all honest parties obtain the same β: if $\beta = 0$ they output ϕ, and if $\beta = 1$ then binary validity applied to the input 0 gives contrapositively that some honest party had $h_k = 2$, so by the observation they all hold the same w. Termination is immediate, the three rounds being fixed and π_{bin} terminating by assumption. □

C Explicit Round Complexities

Corollary 4.3 rests on instantiating Theorem 4.2. We do so three ways here via Corollaries C.1, C.2, and C.3, and compute the constants in each: setup-free at $t < \frac{n}{3}$, in the signature-PKI and VRF-PKI model at $t < \frac{n}{3}$, and in the same model at $t < \frac{n}{2}$. A fourth corollary carries these figures through the reactive transformation of Section 5.2.

Corollary C.1. *For $t < \frac{n}{3}$, there exists a perfectly secure N-parallel Byzantine agreement protocol against up to t corruptions, using no setup, with expected round complexity at most 2232 and worst-case round complexity at most $169f + 665$, where $f \leq t$ is the actual number of corruptions.*

Proof. We instantiate Theorem 4.2 with setup-free building blocks throughout.

We take π_{rand} to be the $t < \frac{n}{3}$ protocol of Theorem 11 of [KK09], denoted KK_3 as in the proof of Corollary 3.5 (no setup, $t_{\mathsf{rand}} < \frac{n}{3}$), with its leader election pipelined as described there, so that an iteration costs 6 rounds after a one-time 3-round prefix. On unanimous input all honest parties terminate within that iteration, so 9 rounds suffice; and an honest leader is elected with probability at least $\frac{2}{3}$ in each iteration, after which all honest parties terminate by the end of the next, so the expected round complexity is at most $6 \cdot \frac{5}{2} + 3 = 18$. Since KK_3 is binary while $\mathsf{Comp}^{\mathsf{IC}}$ runs π_{rand} on the coordinate values, we lift it to the multi-valued domain by the reduction of Turpin and Coan [TC84], at the price of 2 rounds and at the same threshold $t < \frac{n}{3}$; this leaves the shift unchanged and gives 20 and 11 in place of 18 and 9. Finally, KK_3 is 6-shift rather than 1-shift, since a straggler finishes one full iteration behind the first party to terminate, so we run

the termination-synchronization layer of [Bra87] on top of it, which is setup-free and safe at $t < \frac{n}{3}$; this recovers a genuine 1-shift (Property (Q2)) at the price of one additional round, giving $e = 21$ (Property (Q1)) and $c = 12$ (Property (Q3)).

We take π_{es} to be the Abraham–Dolev protocol [AD15] ($t_{\mathsf{es}} < \frac{n}{3}$, no setup), which is early-stopping with optimal worst-case round complexity $f + 2$, i.e., $c_1 = 1$ and $c_2 = 2$.

We take π_{res} to be the protocol of Corollary 3.5, namely $\mathsf{Comp}^{\mathsf{No\text{-}Setup}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{pca}})$ instantiated with KK_3 and the two-round parallel crusader agreement, for the binary invocations of π_{res} (Steps 3 and 10 of $\mathsf{Comp}^{\mathsf{IC}}$), and its Turpin–Coan-lifted multi-valued extension for the invocation over candidate output vectors (Step 8). Both are setup-free and share $t_{\mathsf{res}} < \frac{n}{3}$, which is also the threshold required by the Turpin–Coan reduction. The binary protocol has expected round complexity 24 and worst-case round complexity $14f + 31$; the multi-valued extension costs 2 additional rounds in both cases. We therefore instantiate Theorem 4.2 with the (uniformly valid) upper bounds $e_{\mathsf{res}} = 26$, $c_{1,\mathsf{res}} = 14$, and $c_{2,\mathsf{res}} = 33$.

We take π_{ole} to be the unauthenticated oblivious leader-election protocol of Katz and Koo [KK09], which is setup-free and information-theoretic: each party's tally is generated by moderated verifiable secret sharing, itself built from gradecast, and the party with the smallest tally is elected. Theorem 8 of [KK09] gives fairness $\frac{n-t}{n} - \frac{1}{n^2}$, which for $t < \frac{n}{3}$ is at least $\frac{2}{3}$ (Corollary 9 there), so $p = \frac{2}{3}$ and $\frac{4}{p} = 6$. Its first phase takes 8 rounds and its second a single round.[18] As in the proof of Corollary 3.5, the first phase only samples and shares fresh values and reads nothing the protocol computes, so we run it in the background and charge the invocation only its second phase, $R_{\mathsf{ole}} = 1$. There is ample room: Step 2 occupies $3(r + 1) = 129$ rounds against the 8 the phase needs.

The staggering matters here for the same reason it does there, and the alignment is different. We schedule the background execution consumed by an iteration so that its first phase *ends* at that iteration's Step-5 round, not earlier. The good-event argument needs the elected leader to have been honest when it sent its candidate vector in Step 5, and Definition 6 of [KK09] certifies the elected leader honest at the end of the first phase — exactly the Step-5 round under this schedule, which is what the argument consumes. Starting the phase earlier than that would move the start of the invocation before Step 5, and against an adaptive adversary honesty at that earlier point no longer implies honesty at Step 5. With the phase ending at Step 5 the certification is a fortiori honesty at the start of the invocation as well, so Definition 2.2 is met and $p = \frac{2}{3}$ is unaffected.

Putting it together. Since $t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}} < \frac{n}{3}$, Theorem 4.2 applies for $t = \min\{t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}}\} < \frac{n}{3}$. We take $r = \max(c, 2e) = \max(12, 42) = 42$. Then the resulting protocol $\pi_{\mathsf{pba}}^{N,r} := \mathsf{Comp}^{\mathsf{IC}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{es}}, \pi_{\mathsf{res}}, \pi_{\mathsf{ole}}, N, r)$ realizes N-parallel Byzantine agreement against up to $t < \frac{n}{3}$ corruptions, with no setup and with information-theoretic security.

Substituting $r = 42$, $R_{\mathsf{ole}} = 1$, and $e_{\mathsf{res}} = 26$ into the expected round complexity of Theorem 4.2 gives

$$\frac{4}{p}\big(3(42 + 2 + 1 + 3 \cdot 26) + 3\big) = \frac{4}{p} \cdot 372 = 6 \cdot 372 = 2232,$$

using $p = \frac{2}{3}$.

Substituting $r = 42$, $R_{\mathsf{ole}} = 1$, $c_1 = 1$, $c_2 = 2$, $c_{1,\mathsf{res}} = 14$, and $c_{2,\mathsf{res}} = 33$ into the worst-case round complexity of Theorem 4.2 gives

$$f + 2 + 3\big(2 \cdot 42 + 2 + 1 + 4(14f + 33)\big) + 6 = f + 2 + 3(56f + 219) + 6 = 169f + 665.$$

□

[18] The figure 8 is from the journal version [KK09]; the earlier conference version [KK06] obtains 13 for the same phase.

Corollary C.2. *Assume unique-signature EUF-CMA schemes exist in the random-oracle model and VRF schemes with exponential-size range exist. For $t < \frac{n}{3}$, there exists a computationally secure N-parallel Byzantine agreement protocol in the signature-PKI and VRF-PKI model against up to t corruptions, with expected round complexity at most 1368 and worst-case round complexity at most $49f + 305$, where $f \leq t$ is the actual number of corruptions.*

Proof. We instantiate Theorem 4.2 as follows.

We take π_{rand} to be the protocol of [Mic17] itself (signature PKI, $t_{\mathsf{rand}} < \frac{n}{3}$), which has expected round complexity 9 and, on unanimous input, terminates within 2 rounds. It is, however, only a 3-shift protocol: a party halts in a Coin-Fixed-To-b step, and by Claims B and C of [Mic17] the remaining honest parties then hold b and keep it, but they can halt only at the next Coin-Fixed-To-b step, three rounds later. As discussed above, we therefore run the signature-based termination-synchronization layer on top of it, which is sound here since $t_{\mathsf{rand}} < \frac{n}{3} < \frac{n}{2}$, at the price of one additional round (Property (Q2)). Being binary, it must in addition be lifted to the multi-valued domain by the Turpin–Coan reduction, at the price of 2 further rounds and at the threshold $t < \frac{n}{3}$ it already assumes. Together these give $e = 12$ (Property (Q1)) and $c = 5$ (Property (Q3)).

We take π_{es} to be the Abraham–Dolev protocol [AD15] ($t_{\mathsf{es}} < \frac{n}{3}$, no setup), which is early-stopping with optimal worst-case round complexity $f + 2$, i.e., $c_1 = 1$ and $c_2 = 2$.

We take π_{res} to be the protocol of Corollary 3.7, namely $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand}})$ instantiated with the protocol of [Mic17], for the binary invocations of π_{res} (Steps 3 and 10 of $\mathsf{Comp}^{\mathsf{IC}}$), and its Turpin–Coan-lifted multi-valued extension for the invocation over candidate output vectors (Step 8). Both share $t_{\mathsf{res}} < \frac{n}{3}$. The binary protocol has expected round complexity 14 and worst-case round complexity $4f+10$; the multi-valued extension costs 2 additional rounds in both cases, i.e., expected round complexity 16 and worst-case round complexity $4f + 12$. Since Theorem 4.2 uses a single pair of constants to bound every invocation of π_{res}, we instantiate it with the (uniformly valid) upper bounds $e_{\mathsf{res}} = 16$, $c_{1,\mathsf{res}} = 4$, and $c_{2,\mathsf{res}} = 12$.

We take π_{ole} to be the standard VRF-based sortition leader-election protocol: each party evaluates a VRF on the session identifier and the current iteration number as a public seed and, in a single round, broadcasts its output together with a proof of correct evaluation; every party then locally elects, among the valid outputs it received, the party achieving the (canonically ordered) smallest VRF output as the leader. In the VRF-PKI model of Section 2.1 every key pair is sampled by the setup, so residual pseudorandomness applies to all n values at the seed and the smallest of them belongs to a party honest at the start of the invocation with probability at least $(n - t)/n$; and an honest party's output reaches every party within the single round, so all honest parties see that minimum and elect the same leader, as Definition 2.2 requires. A rushing adversary can withhold its own outputs, which only removes candidates and cannot displace an honest minimum. This is a leader-election protocol against $t < \frac{n}{2}$ corruptions with fairness $p \geq (n - t)/n$ and round complexity $R_{\mathsf{ole}} = 1$, assuming VRF schemes with exponential-size range. These calls, like those of the protocols run inside the copies and invocations of Figure 3, use the same keys instead of separate setups, but each one evaluates or signs under its own label, so their values are independent. Seeing its corrupted parties' values early does not help the adversary, since corrupting all of them up front is its best strategy and the bound on E_1 holds whatever it knows, so the bounds in the proof of Theorem 4.2 still hold.

Since $t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}} < \frac{n}{3}$ and the leader-election protocol tolerates the (weaker) bound $t_{\mathsf{ole}} < \frac{n}{2}$, Theorem 4.2 applies for $t = \min\{t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}}\} < \frac{n}{3}$. We take $r = \max(c, 2e) = \max(5, 24) = 24$. Then the resulting protocol $\pi^{N,r}_{\mathsf{pba}} := \mathsf{Comp}^{\mathsf{IC}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{es}}, \pi_{\mathsf{res}}, \pi_{\mathsf{ole}}, N, r)$ realizes N-parallel Byzantine agreement against up to t corruptions.

Substituting $r = 24$, $R_{\mathsf{ole}} = 1$, and $e_{\mathsf{res}} = 16$ into the expected round complexity of Theorem 4.2

gives

$$\frac{4}{p}(3(24+2+1+3\cdot 16)+3) = \frac{4}{p}\cdot 228.$$

Since $t < \frac{n}{3}$, we have $p = (n-t)/n > \frac{2}{3}$, and hence $\frac{4}{p} < 6$. Therefore the expected round complexity is at most $6 \cdot 228 = 1368$.

Substituting $r = 24$, $c_1 = 1$, $c_2 = 2$, $c_{1,\mathsf{res}} = 4$, and $c_{2,\mathsf{res}} = 12$ into the worst-case round complexity of Theorem 4.2 gives

$$f + 2 + 3(2 \cdot 24 + 2 + 1 + 4(4f + 12)) + 6 = f + 2 + 3(16f + 99) + 6 = 49f + 305.$$

□

Corollary C.3. *Assume EUF-CMA signature schemes and VRF schemes with exponential-size range exist. For $t < \frac{n}{2}$, there exists a computationally secure N-parallel Byzantine agreement protocol against up to t corruptions in the signature-PKI and VRF-PKI model, with expected round complexity at most* 4152 *and worst-case round complexity at most* $184f + 729$*, where $f \le t$ is the actual number of corruptions.*

Proof. We instantiate Theorem 4.2 as follows.

Instantiating π_{rand}. We take π_{rand} to be the protocol of Appendix B of [KK06] (KK), as analysed in the proof of Corollary 3.8, whose phases are its iterations: $c = 6 + R_{\mathsf{ole}} = 7$ rounds per iteration, with the same VRF-based sortition π_{ole} ($R_{\mathsf{ole}} = 1$), and an honest leader is elected in each iteration with probability at least $p = \frac{1}{2}$.

For Property (Q3), if all honest parties enter an iteration holding the same value v then, by the argument for (P2) in the proof of Corollary 3.8, they all lock on v in Step 4 of that iteration; each then sets $\mathsf{lock} := 0$ at that iteration's Step 7 and outputs v at the next one, i.e., within $2c = 14$ rounds.

For Property (Q1), note that in each iteration an honest leader is elected with probability at least $p = \frac{1}{2}$, whatever the local values, and that, as in the proof of Corollary 3.8, none of k iterations elects one with probability at most 2^{-k}. So the index of the first such iteration is dominated by a geometric random variable with success probability $\frac{1}{2}$ and has expectation at most 2. In that iteration all honest values become equal, by Lemma 18 of [KK06]; they all lock in the next one and output one iteration after that. Hence the expected number of iterations is at most $2 + 2 = 4$ and the expected round complexity is at most $4c = 28$.

KK under this presentation is only a 7-shift protocol, not a 1-shift one: once the first honest party locks in some iteration I, all remaining honest parties do so in $I + 1$, and each outputs one iteration after it locks, so the stragglers terminate at the end of iteration $I + 2$ – a gap of one full iteration, $c = 7$ rounds, behind the first decider. As mentioned above, we can use the signature-and-echo termination-synchronization layer to recover a genuine 1-shift at the price of one additional round, giving expected round complexity $e = 29$ (Property (Q1)) and unanimity-convergence constant $c = 15$ (Property (Q3)), with $t_{\mathsf{rand}} < \frac{n}{2}$.

Instantiating π_{es}. We take π_{es} to be the protocol of Theorem 2 of [ELP25] ($t_{\mathsf{es}} < \frac{n}{2}$, signature PKI), with worst-case round complexity $f + 6\lceil\sqrt{f}\rceil + 6$. That protocol is binary (its Definition 1 fixes both the inputs and the outputs to $\{0, 1\}$), whereas Procedure 1 of $\mathsf{Comp}^{\mathsf{IC}}$ runs π_{es} on the coordinate values themselves, so it must be lifted to the multi-valued domain. The Turpin–Coan reduction used earlier in this section is unavailable here, as it needs $t < \frac{n}{3}$; at $t < \frac{n}{2}$ we instead use the transformation of Lemma B.2 in Appendix B, which costs 3 additional rounds, assumes no more than the signature PKI already in force, and preserves early stopping, the added rounds being a fixed prefix. Hence π_{es} has worst-case round complexity $f + 6\lceil\sqrt{f}\rceil + 9$. Since $(\sqrt{f} - 1)^2 \ge 0$

for all $f \geq 0$, we have $\sqrt{f} \leq \frac{f+1}{2}$, so $6\lceil\sqrt{f}\rceil \leq 6(\sqrt{f}+1) \leq 6\left(\frac{f+1}{2}+1\right) = 3f+9$, and hence $f + 6\lceil\sqrt{f}\rceil + 9 \leq f + 3f + 9 + 9 = 4f + 18$. We instantiate Theorem 4.2 with $c_1 = 4$, $c_2 = 18$.

Instantiating π_{res}. We take π_{res} to be the protocol of Corollary 3.8, namely $\mathsf{Comp}^{\mathsf{PKI}}(KK)$ as in that corollary, for *all three* invocations of π_{res} (Steps 3, 8, and 10 of $\mathsf{Comp}^{\mathsf{IC}}$): since the protocol of Corollary 3.8 is natively multi-valued (its certificates carry an arbitrary value v directly), no binary-to-multivalued lift is required for the Step-8 invocation, unlike the Turpin–Coan-lifted instantiation used in the corollary earlier in this section. This gives $t_{\mathsf{res}} < \frac{n}{2}$, expected round complexity $e_{\mathsf{res}} = 2(2c+1) + c = 37$, and worst-case round complexity $c_{1,\mathsf{res}} f + c_{2,\mathsf{res}} = 15f + 29$, as in that corollary, i.e., $c_{1,\mathsf{res}} = 15$, $c_{2,\mathsf{res}} = 29$.

Instantiating π_{ole}. We take π_{ole} to be the same VRF-based sortition leader-election protocol used in Section 3: each party evaluates a VRF on the session identifier and the current iteration number and, in a single round, broadcasts its output together with a proof of correct evaluation; every party then elects, among the valid outputs it received, the party achieving the (canonically ordered) smallest VRF output as leader. As above, this is a leader-election protocol against $t < \frac{n}{2}$ corruptions with fairness $p \geq (n-t)/n$ and round complexity $R_{\mathsf{ole}} = 1$, assuming VRF schemes with exponential-size range in the VRF-PKI model of Section 2.1.

Putting it together. Since $t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}} < \frac{n}{2}$, Theorem 4.2 applies for $t = \min\{t_{\mathsf{es}}, t_{\mathsf{rand}}, t_{\mathsf{res}}, t_{\mathsf{ole}}\} < \frac{n}{2}$. We take $r = \max(c, 2e) = \max(15, 58) = 58$. Then the resulting protocol $\pi_{\mathsf{pba}}^{N,r} := \mathsf{Comp}^{\mathsf{IC}}(\pi_{\mathsf{rand}}, \pi_{\mathsf{es}}, \pi_{\mathsf{res}}, \pi_{\mathsf{ole}}, N, r)$ realizes N-parallel Byzantine agreement against up to $t < \frac{n}{2}$ corruptions.

Substituting $r = 58$, $R_{\mathsf{ole}} = 1$, and $e_{\mathsf{res}} = 37$ into the expected round complexity of Theorem 4.2 gives

$$\frac{4}{p}\big(3(58 + 2 + 1 + 3 \cdot 37) + 3\big) = \frac{4}{p} \cdot 519.$$

Since $t < \frac{n}{2}$, we have $p = (n-t)/n > \frac{1}{2}$, and hence $\frac{4}{p} < 8$. Therefore the expected round complexity is at most $8 \cdot 519 = 4152$.

Substituting $r = 58$, $c_1 = 4$, $c_2 = 18$, $c_{1,\mathsf{res}} = 15$, and $c_{2,\mathsf{res}} = 29$ into the worst-case round complexity of Theorem 4.2 gives

$$4f + 18 + 3\big(2 \cdot 58 + 2 + 1 + 4(15f + 29)\big) + 6 = 4f + 18 + 3(60f + 235) + 6 = 184f + 729.$$

□

This corollary trades constants for resilience against the one above: it achieves the full threshold $t < \frac{n}{2}$ rather than $t < \frac{n}{3}$, at the cost of larger constants (4152 vs. 1368 expected, $184f + 729$ vs. $49f + 305$ worst-case), driven mainly by KK's heavier per-phase structure relative to $BBA^\star$.

Corollary C.4. *For any polynomial number R of sequential invocations, there exist reactive interactive-consistency protocols in the stand-alone model against static, rushing, malicious adversaries, each invocation of which has the following security and complexity guarantees, where $f \leq t$ is the actual number of corruptions:*

- *a perfectly t-secure protocol, using no setup, for $t < \frac{n}{3}$, with expected round complexity at most* 2236 *and worst-case round complexity at most* $171f + 673$*; and*
- *a computationally t-secure protocol, in the signature-PKI and VRF-PKI model, for $t < \frac{n}{2}$, assuming EUF-CMA signature schemes and VRF schemes with exponential-size range, with expected round complexity at most* 4156 *and worst-case round complexity at most* $192f + 769$*.*

Proof. Both cases substitute into (5.1).

For the setup-free protocol the fallback is the Abraham–Dolev protocol [AD15], so $c_1 f + c_2 = f + 2$. Corollary C.1 gives expected 2232 and worst-case $169f + 665$, and Lemma A.2 adds one round to each, so $E_{\mathsf{ic}} = 2233$ and $W_{\mathsf{ic}}(f) = 169f + 666$. Hence

$$E_{\mathsf{ric}} = 2233 + 3 = 2236, \qquad W_{\mathsf{ric}}(f) = 169f + 666 + 2(f + 2) + 3 = 171f + 673.$$

For the $t < \frac{n}{2}$ protocol the fallback is the lifted protocol of Theorem 2 of [ELP25], so $c_1 f + c_2 = 4f + 18$. Corollary C.3 gives expected 4152 and worst-case $184f + 729$, so $E_{\mathsf{ic}} = 4153$ and $W_{\mathsf{ic}}(f) = 184f + 730$. Hence

$$E_{\mathsf{ric}} = 4153 + 3 = 4156, \qquad W_{\mathsf{ric}}(f) = 184f + 730 + 2(4f + 18) + 3 = 192f + 769.$$

□

D Related Work

Below, we give a more comprehensive overview of works related to our main contributions, as a supplement to those introduced in the introduction.

Expected constant-time Byzantine agreement and broadcast in private point-to-point, synchronous networks (without early stopping). Feldman and Micali constructed the first optimally resilient, information-theoretic BA protocol terminating in an expected constant rounds [FM97]. Goldreich and Petrank subsequently constructed an expected constant time protocol with a worst-case running time of $t+O(\log t)$ [GP90]. Katz and Koo gave concrete expected-round bounds of 18 rounds for (t<n/3) and 56 rounds for (t<n/2), the latter using signatures and PKI [KK09]. In the authenticated setting, [ADD+19] reduced the expected complexity for $t < n/2$ to 10 rounds against static corruptions and 26 rounds against adaptive corruptions, while [ACD+19] obtained expected (O(1))-round BA for $t < (1 - \epsilon)n/2$ with improved communication. Recent work also investigates the gap between asymptotic and concrete performance: Zhang et al. develop practical synchronous randomized BFT protocols based on expected-constant-time techniques [ZHDZ25].

For Byzantine broadcast (BB), $t > n/2$ resilience is possible using PKI. Garay et al. obtained $O(k^2)$-round protocols where $t < n/2 + k$ [GKKO07]. Wan et al. constructed weakly adaptively secure BB tolerating $t < n - 1$, with expected $O((n/(n - t))^2$ rounds and hence expected constant complexity whenever the honest fraction is constant [WXSD20]. Srinivasan et al. compiled weakly adaptive BB to a strongly adaptive one, preserving expected $O(1)$-rounds for $t < n$, using time-lock puzzles and a random oracle [SLM+23]. Civit, Emdin, and Guerraoui further obtain expected constant-round interactive consistency (IC) for $t < n$ under PKI [CEG26].

Early-stopping BA and BB (without expected constant rounds). When the actual number of corruptions f is lower than the corruption threshold t, some BA/BB protocols may be able to terminate sooner (with number of rounds as a function of f)—these are referred to as *early-stopping* protocols. In our information-theoretic setting, the study of early-stopping protocols began with Toueg, Perry, and Srikanth who gave a $\min(2f+4, 2t+2)$ round protocol for $t < n/3$ [TPS85]. Dolev, Reischuk, and Strong obtained the optimal $\min(f+2, t+1)$ rounds under $n > \max(4t, 2(t+(t-1)^2))$ [DRS90]. Berman, Garay, and Perry achieved the first information-theoretic protocol with optimal $t < n/3$ and $\min(f + 2, t + 1)$, but their exponential information-gathering protocol requires exponential communication [BGP92]. Subsequently, optimal early stopping for $t < n/4$ with a polynomial message protocol was achieved [BG93], and Krings and Feyer gave a alternative

optimal early stopping, EIG-based protocol for $t < n/3$ [KF99]. Garay and Moses gave the first polynomial protocol for $t < n/3$ with almost-optimal early-stopping $\min(t+1, f+3)$ [GM98]. Okun and Barak later achieved an early-stopping construction whose round complexity depends on n, t, and f, although it is not optimal [OB08]. Abraham and Dolev finally combined optimal resilience, polynomial complexity, and optimal $\min(f+2, t+1)$ early stopping [AD15]. Lenzen and Sheikholeslami reduced the communication to $O(n^2)$ using a recursive Phase-King construction, at the cost of $6(f+1)$ rounds [LS22].

With setup assumptions (generally PKI), early-stopping BA/BB with $t < n/2$ or even $t < n$. For $t < n/2$, Elsheimy, Loss, and Papamanthou construct a near-optimal early stopping BA polynomial protocol terminating in $f + 6\lceil\sqrt{f}\rceil + 6$ rounds assuming PKI, as well as a BB protocol with an almost-optimal $\min(f+3, t+1)$ but exponential communication [ELP25]. Assuming PKI, Loss and Nielsen tolerate $t < (1-\epsilon)n$, $\epsilon > 0$ and achieve $O(\min(f^2, t+1))$ rounds [LN24]. Deligios, Klasovitá, and Liu-Zhang extend this to arbitrary $t < n$, obtaining $O(\min(f^2, t))$ rounds in general and $O(f)$ when $t < (1-\epsilon)n$ [DKLZ25].

Expected Constant and Early-Stopping Protocols A few previous works have constructed BA/BB protocols that combine both early-stopping and expected constant rounds. In the realm of $t = \Omega(n)$ (with assumption of a trusted dealer, $t < n/8$), Zamsky [Zam96] improved on the worst case round complexity of [GP90] to the optimal $\min(f+2, t+1)$. With assumptions of signatures/PKI, recent work have presented protocols for $t < n/2$, with $(d+9) \times (\lfloor f/d \rfloor + 1) + 2$ rounds (which is $(1+\epsilon)f$, $\epsilon \in (0,1)$, for a large enough d and large enough f with respect to d) [ELP24], and for $t < (1-\alpha)n/2$ ($\alpha \in (0,1)$), $(1+\epsilon)f$ ($\epsilon \in (0,1)$) rounds [EKL25].

A natural question is whether the recent results under $t < n/2$ can trivially imply our results in the $t < n/3$, information-theoretic setting—unfortunately, this is not the case. There exists essential components in these works relying on signatures, which do not easily translate to an equivalent protocol in our setting. First, [ELP24, EKL25] rely on a subprotocol called Correct-or-Detect Broadcast [FN09], where parties detect a common subset of corrupted parties. It is unclear how to substitute this protocol with an information-theoretic one. While our protocol parties indeed also detect misbehaving parties, the type of misbehaviour detected is when corrupt parties falsely claim to have terminated. Another difference is that in [ELP24, EKL25], parties must present proofs-of-participation (signatures from other parties endorsing their non-corrupted-ness) in order to continue participating in the protocol. In contrast, parties in our protocol *locally* decide to ignore detected corrupt parties, and do not convince others who the bad parties are.

Composition of BA/BB protocols in synchronous, point-to-point networks. Composition requires more nuance for probabilistically terminating protocols: N parallel instance of an expected constant round protocol can require $\Theta(\log N)$ expected rounds, since termination occurs only when the slowest instance finishes. Ben-Or and El-Yaniv addressed this by constructing expected constant round parallel BA and interactive consistency (IC) protocols [BOEY03]. Their construction executes multiple truncated BA instances and generate common randomness in order to agree on the output. Cohen, Coretti, Garay, and Zikas [CCGZ16] give a general treatment in the UC framework, proving composition theorems for replacing deterministic-termination ideal functionalities with expected round protocols, treating both the issue of parallel and sequential composition.

Sequential composition presents another challenge as parties in expected constant round BA/BB protocols generally do not have simultaneous termination. Thus, parties running such a BA or BB instance inside a protocol may not know when every has terminated this instance, and could start

the next protocol round. Lindell, Lysyanskaya, and Rabin initiated the formal study of sequential composition without simultaneous termination [LLR02], and Katz-Koo gave a 'staggering' mechanism for sequential composition of broadcasts [KK09]. Civit et al. [CDG+25] studied amortizations under multiple sequential compositions, and its applications to our work may be of future interest.