Concretely Efficient Byzantine Agreement and Interactive Consistency for Honest Majority*

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

November 19, 2026

Abstract

We consider two of the main notions of round-efficient Byzantine agreement (BA): *expected constant rounds* and *early stopping*. Randomized protocols can terminate in an expected constant number of rounds, while early-stopping protocols terminate in a number of rounds that depends only on the actual number $f \leq t$ of corrupted parties. Recent works achieve both guarantees at once, but at a high concrete cost: assuming PKI, for the optimal threshold $t < n/2$, the previous protocol with the smallest expected round complexity needs 37 rounds in expectation and $15f + 29$ in the worst case.

In this work, we present a BA protocol for $t < n/2$, secure against an adaptive adversary assuming a PKI, that terminates in 18 rounds in expectation and in at most $8f + 18$ rounds in the worst case. Moreover, our protocol has the following *optimistic feature*: if no party is corrupted and all inputs agree, every party outputs after a single round. Our protocol is natively multi-valued and does not rely on random oracles. At the core of our result is a new randomized BA protocol in which a successful leader election leads to agreement within the same phase.

We extend our results to Byzantine broadcast, obtaining a protocol that has 19 rounds in expectation and $8f + 19$ in the worst case, and to interactive consistency (IC), retaining the same round complexity while providing an optimistic fast path in which all parties output after only 2 rounds when everyone is honest. We do so by directly parallelizing our BB protocol, instead of using a generic IC compiler, for which, at $t < n/2$, the best known bounds are more than 4000 rounds in expectation and $184f + 730$ in the worst case.

*We would like to thank Muhammad Ishaq Malik for useful conversations about the implementation of these protocols and the needs of real deployment environments. Vassilis Zikas was supported in part by the Stellar Development Foundation and JPMorgan Chase. All authors were supported in part by Sunday Group, Inc.

†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 contributions in detail . . . 2
  - 1.1.1 Comparison with Prior Work . . . 3
- 1.2 Technical overview . . . 4

2 Preliminaries and Model **9**
- 2.1 Model . . . 9
- 2.2 Building Blocks . . . 10
- 2.3 Cryptographic Primitives . . . 12
- 2.4 The Randomized Early-Stopping Compiler . . . 13

3 Byzantine Agreement and Broadcast in Expected Constant Rounds **15**
- 3.1 Expected-Constant-Round Byzantine Agreement . . . 15
  - 3.1.1 The Protocol . . . 15
  - 3.1.2 Analysis . . . 17
- 3.2 From Agreement to Broadcast . . . 20

4 Early-Stopping Byzantine Agreement and Broadcast in Expected Constant Rounds **21**
- 4.1 Expected-Constant-Round Early-Stopping Byzantine Agreement . . . 21
- 4.2 Expected-Constant-Round Early-Stopping Byzantine Broadcast . . . 24

5 Parallel Byzantine Agreement and Interactive Consistency **24**

Bibliography **26**

A Constructions of the Building Blocks **28**

1 Introduction

Byzantine agreement (BA), Byzantine broadcast (BB) and interactive consistency (IC) are core problems in cryptography and distributed computing. In BA, n parties, each holding an input, must agree on a common output (Agreement), and this output must be the honest parties' input whenever they all hold the same one (Validity). In BB, a designated sender distributes a value so that all honest parties output the same value, which is the sender's value whenever the sender is honest. IC lets every party broadcast a value at the same time: all honest parties output the same vector, whose j-th entry is P_j's input whenever P_j is honest. IC is precisely the broadcast channel assumed by many protocols for secure multi-party computation (MPC), and realizing it over point-to-point channels often dominates the round complexity[1] of the resulting MPC protocol [CFLZ26].

Deterministic BA protocols require $t+1$ rounds in the worst case [DS83], where t is the maximum number of corrupted parties. Two lines of work circumvent this bound. *Randomized* protocols terminate in an expected constant number of rounds [FM88, KK06a], but don't necessarily guarantee when they terminate in the worst case. *Early-stopping* protocols terminate in a number of rounds that depends only on the number $f \leq t$ of parties that are actually corrupted. The optimal worst-case round complexity is $\min(f+2, t+1)$ [DRS90], and we treat $O(f)$-round early stopping as near optimal. A few works achieve both guarantees at once [Zam96, ELP24, EKL25, CFLZ26]; in particular, [CFLZ26] gives a compiler that turns a broad class of expected-constant-round BA protocols into protocols that are also early stopping. These protocols, however, are expensive in terms of concrete round complexity. At the optimal threshold $t < n/2$, the protocol of Elsheimy, Loss and Papamanthou [ELP24] needs at least 52 rounds in expectation, its communication-efficient successor [EKL25] at least 72, and the compiled Katz–Koo protocol of [CFLZ26] 37 rounds in expectation and $15f+29$ in the worst case. For IC the gap is wider: the parallel compiler of [CFLZ26] yields bounds of more than 4000 rounds in expectation and $184f + 730$ in the worst case. Since these protocols replace an idealized broadcast channel that costs a single round, their constants matter. We thus ask:

> *Can we design Byzantine agreement and interactive consistency protocols for $t < n/2$ that are both expected-constant-round and early-stopping, with a small concrete round complexity, both in expectation and in the worst case?*

We answer this question positively. Our main result is the following.

Theorem (informal)**.** *Assume signature schemes and verifiable random functions (VRFs) exist. There exists a BA protocol in the PKI model, secure against an adaptive adversary corrupting up to $t < n/2$ parties, that terminates in fewer than 18 rounds in expectation and in at most $8f + 18$ rounds in the worst case, where $f \leq t$ is the actual number of corruptions.*

Our protocol is natively multi-valued (the input of each party can be a bit-string and the round complexity does not depend on the string's length). Our protocols do not rely on random oracles, but we do require the VRF keys to be honestly generated (we can remove this assumption relying on RO as we discuss later).

Moreover, our protocol has an optimistic path, which guarantees that if no party is corrupted and all inputs agree, then every party outputs after a single round. Moving to interactive consistency costs only one additional round, in which every party sends its input as a sender, hence allowing to claim the following.

[1] Unless otherwise specified, round complexity refers to the synchronous point-to-point setting: in one round, every party sends one message over each point-to-point channel.

Theorem (informal)**.** *Under the same assumptions, there exists an IC protocol, secure against an adaptive adversary corrupting up to $t < n/2$ parties, that outputs in (i) 2 rounds optimistically, (ii) 19 rounds in expectation and (iii) in at most $8f + 19$ rounds in the worst case.*

In the next section, we give a more detailed overview of our results.

1.1 Our contributions in detail

A round-efficient randomized BA for $t < n/2$. At the core of our results is a new phase-based randomized BA protocol $\pi_{\mathsf{rand\text{-}ba}}$. Each phase takes seven rounds: two rounds of *certified crusader interactive consistency* (which we define in Definition 2.4), in which every party obtains a publicly verifiable vector of all parties' current values or $\perp$; one round in which every party sends its vector to all parties; a one-round VRF-based leader election; and three rounds of graded agreement on the most frequent value in the leader's vector. With probability at least $(n-t)/n > 1/2$ the elected leader is honest, which leads to all honest parties outputting at the end of that same phase. Crucially, the leader is revealed in a round *after* every party has sent its vector, so an adaptive adversary learns who the leader is too late to change the leader's proposal.

Informal Corollary. *Assuming EUF-CMA signature schemes and VRFs with exponential-size range, $\pi_{\mathsf{rand\text{-}ba}}$ is a BA protocol for $t < n/2$ in the signature-PKI and VRF-PKI model, with expected round complexity $7n/(n-t) + 1 < 15$.*

An expectation over all executions needs some care here: with negligible probability, for instance under a setup in which the corrupted parties win every election, an execution may never end. We therefore stop $\pi_{\mathsf{rand\text{-}ba}}$ after $\lceil \kappa n/(n-t) \rceil$ phases; since the first k elections all fail with probability at most $(t/n)^k$, this costs only a negligible error.

The protocol terminates in two rounds when all parties are honest and all inputs agree, and if either condition holds on its own, every party outputs within one phase.

Expected constant rounds and early stopping. On its own, $\pi_{\mathsf{rand\text{-}ba}}$ is not early-stopping: every phase can fail to elect an honest leader. We thus apply the compiler $\mathsf{Comp}^{\mathsf{PKI}}$ of [CFLZ26]. In the compiled protocol, if no party claims to have committed to an output, every party that has not committed switches to a fixed default value, so the honest parties become unanimous and commit in the next phase. Corrupted parties can only delay this by falsely claiming to have committed. Such a claim is signed and forwarded to all parties. If an honest party has not committed one phase after receiving it, the claim was false, and from then on it ignores the sender. Hence, each corrupted party delays termination by at most one phase. By plugging our protocol $\pi_{\mathsf{rand\text{-}ba}}$ into $\mathsf{Comp}^{\mathsf{PKI}}$, we then can claim the following.

Theorem (informal)**.** *Assuming EUF-CMA signature schemes and VRFs with exponential-size range, $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}ba}})$ is a BA protocol for $t < n/2$ in the signature-PKI and VRF-PKI model, secure against an adaptive adversary, with expected round complexity $8n/(n-t) + 2 < 18$ and worst-case round complexity at most $8f + 18$.*

Byzantine broadcast. BB follows by letting the sender sign and send its value in one round before the BA protocol starts. This yields expected round complexity of 19 and worst-case round complexity of $8f+19$ (Theorem 4.3). Moreover, every party outputs within 8 rounds when the sender is honest and after just 2 rounds when there are no corrupted parties.

Parallel BA and interactive consistency. Running m independent instances of a randomized BA protocol in parallel does not, in general, preserve expected constant round complexity [BOEY03]. While generic parallel-composition compilers [BOEY03, CFLZ26] do exist, they incur high round complexity overhead. We propose a new compiler that is based on the simple idea of re-using a single leader election for multiple instances of the input protocol. We show that this guarantees

that every instance terminates by the end of the first phase whose election returns an honest leader. In particular, we can claim the following.

Theorem (informal)**.** *Under the assumptions of the previous theorem, for any polynomial m, our m-parallel BA has expected round complexity* 18 *and worst-case round complexity at most* $8f + 18$*, and IC has expected round complexity* 19 *and worst-case round complexity at most* $8f + 19$*.*

For IC at $t < n/2$, this improves on the 4153 rounds in expectation and $184f + 730$ rounds in the worst case obtained from the state-of-the-art parallel compiler of [CFLZ26].

1.1.1 Comparison with Prior Work

Tables 1 and 2 compare our protocols with the instantiation of the same compiler with the point-to-point protocol of [KK06a] given in [CFLZ26], and with the protocols of [ELP24] and [EKL25], the only prior protocols we know of that are both expected-constant-round and early-stopping at or near the optimal threshold $t < n/2$. For interactive consistency, the only prior protocol with both guarantees is the parallel compiler of [CFLZ26].

The Katz–Koo instantiation of [CFLZ26] has the same threshold and setup, with expected round complexity 37 and worst-case round complexity $15f + 29$. These figures assume that their leader election, originally based on moderated VSS, is replaced by the one-round VRF-based election that we also use, which is why that row of the table also assumes a VRF PKI; with their original election both figures would be considerably larger. Both protocols take seven rounds per iteration,[2] so our improvement does not come from shorter iterations. It has two sources.

First, we need half as many phases. The compiler must not reset values to the default between a good election and the commitment that this election causes, since the reset would undo the agreement the election has just produced. In [KK06a], parties lock on a value before the leader is elected, so a good election leads to commitment only in the following iteration, and the compiler may therefore reset values only after every second iteration. This doubles the number of phases, both in expectation and for each corrupted party in the worst case. In $\pi_{\mathsf{rand\text{-}ba}}$, by contrast, every party's proposal is made publicly verifiable and delivered to all parties before the leader is known, and graded agreement follows in the same phase. A good election therefore leads to commitment within its own phase, and the compiler can reset values after every phase.

Second, our parties stop sooner after committing. In the Katz–Koo instantiation, a committed party must keep running for almost a full further iteration, because the parties that have not yet committed still need its signatures; in ours, it sends a terminal certificate and halts one round later.

The protocol of [ELP24] is the closest prior point of comparison. It is likewise expected-constant-round and early-stopping under a PKI at $t < \frac{n}{2}$. In its version with termination certificates, its worst-case round complexity is $(d + 9)(\lfloor f/d \rfloor + 1) + 2$, and its expected round complexity is $(\frac{2}{p} + 1)(d + 9) + 2$; they quantify the expectation only for the version without certificates, and we derive this figure from their analysis. Here $d \geq 1$ is a tunable integer governing how many corrupted parties are detected, and how many extra rounds are spent, per iteration, and p is the agreement probability of the common coin they assume. The protocol of [EKL25] builds on it with expected round complexity $6(d+11)$ and worst-case round complexity $(d+11)(\lfloor f/d \rfloor + 2)$. In both, raising d drives the coefficient of f, which is $1+\frac{9}{d}$ and $1+\frac{11}{d}$ respectively, towards the optimal $(1+\epsilon)f$, which

[2] We refer to the point-to-point protocol of Appendix B of the full version [KK06b]; the proceedings version [KK06a] does not contain it. Their iteration could be shortened to six rounds by running the leader election in parallel with the round before it, which would lower their coefficient of f from 15 to 13. This requires the atomic model of [GKKZ11], which we do not assume; under that model our protocol shortens in the same way, to 16 rounds in expectation and $7f + 16$ in the worst case, so the comparison is unchanged.

we do not achieve; but it drives the expectation up in step, and even at $d = 1$ that expectation is 52 and 72, the former at coin agreement probability $p = \frac{1}{2}$.

The Katz–Koo instantiation of [CFLZ26] already beats both on the expected bound, but its coefficient of f, namely 15, exceeds theirs at every d. Our expected bound of 18 is the lowest in the table, and at $d = 1$, where each of them attains its smallest expectation, our coefficient of f is 8 against their 10 and 12. The additive constant of [ELP24] is smaller than ours, so at $d = 1$ its worst-case bound $10f + 12$ is below our $8f + 18$ for $f \leq 2$ and equal to it at $f = 3$. For $d \geq 2$ their coefficients drop below ours, but only at an expectation of at least 57 and 78. When no party is corrupted and all inputs agree, every party of our protocol halts after two rounds, against $d + 11 \geq 12$ and $2(d + 11) \geq 24$ for [ELP24] and [EKL25].

Beyond round complexity, the protocols differ in their assumptions, their domain and their communication.

Assumptions. The protocol of [ELP24] tolerates $t < \frac{n}{2}$ only given an ideal common coin. The protocol of [EKL25] builds this coin from VRFs, which it also uses to sample small committees. This has two costs. First, its threshold is slightly below $\frac{n}{2}$, namely $t < (1 - \alpha)\frac{n}{2}$ for a constant $\alpha > 0$, so that every committee has an honest majority with overwhelming probability. Second, it relies on the random oracle model: its VRFs are modelled by the functionality of [DGKR18], which also covers maliciously generated keys and which [DGKR18] realize in the random oracle model. Our protocol tolerates $t < \frac{n}{2}$ exactly and uses no random oracle, since our leader election needs only a standard VRF; in exchange, we assume that all VRF keys, including those of corrupted parties, are generated honestly. On the other hand, [ELP24] is proven against a strongly adaptive adversary, which may delete the messages a party sent in the round in which it is corrupted; our leader election, like the committee elections of [EKL25], relies on such messages being delivered.

Domain. Our protocol is natively multi-valued: inputs can be arbitrary strings, and the round complexity does not depend on their length. The protocols of [ELP24] and [EKL25], by contrast, are binary. To the best of our knowledge, extending them to longer values with standard techniques would increase their round complexity.

Communication. Here [EKL25] is clearly better: each step is run by a small committee sampled with VRFs, so its communication is linear in n, whereas our protocol is all-to-all. In exchange, our protocol needs a quarter of the expected rounds, tolerates $t < \frac{n}{2}$ exactly, and is natively multi-valued. It is therefore the better fit when latency matters more than bandwidth, as when it replaces a broadcast channel inside an MPC protocol.

1.2 Technical overview

Our first goal is to design a BA protocol for $t < n/2$ that terminates in a small expected constant number of rounds and in $O(f)$ rounds in the worst case. We proceed in two steps. We first design a new randomized BA protocol $\pi_{\mathsf{rand\text{-}ba}}$ with expected constant round complexity with some special properties. These special properties allow us to use $\pi_{\mathsf{rand\text{-}ba}}$ as input to the compiler of [CFLZ26], and obtain an early-stopping protocol that preserves the expected constant termination.

The challenging part is to design $\pi_{\mathsf{rand\text{-}ba}}$ so that the special properties hold while keeping a low expected round-complexity. Finally, we show that many instances of the resulting protocol can run in parallel at no extra cost in terms of rounds, which yields our final interactive consistency protocol.

The standard template for randomized BA. Most expected-constant-round BA protocols follow a similar template [FM88, KK06a]. The protocol proceeds in phases, and each phase combines two steps. The first is a *graded agreement* execution. In this every party outputs a value together with a grade that tells it how confident it can be that agreement has been reached; a party that obtains

	Threshold	Expected	Worst case	No corruption[†]
Byzantine agreement				
This work	$\boldsymbol{t < \frac{n}{2}}$	**18**	$\boldsymbol{8f + 18}$	**2**
[CFLZ26] with [KK06a]	$t < \frac{n}{2}$	37	$15f + 29$	29
[ELP24]	$t < \frac{n}{2}$	$(\frac{2}{p} + 1)(d + 9) + 2 \geq 52^{\mathcal{P}}$	$(d + 9)(\lfloor f/d \rfloor + 1) + 2$	$d + 11$
[EKL25]	$t < (1 - \alpha)\frac{n}{2}$	$6(d + 11) \geq 72$	$(d + 11)(\lfloor f/d \rfloor + 2)$	$2(d + 11)$
Interactive consistency				
This work	$\boldsymbol{t < \frac{n}{2}}$	**19**	$\boldsymbol{8f + 19}$	**3**
[CFLZ26]	$t < \frac{n}{2}$	4153	$184f + 730$	730

Table 1: Round complexity of protocols that are both expected-constant-round and early-stopping at or near the optimal threshold $t < n/2$; $f \leq t$ is the actual number of corruptions and $d \geq 1$ a tunable integer. [†]Rounds until every party halts when no party is corrupted and, for BA, all inputs agree. For our protocols this is the exact number of rounds of such an execution, which is much smaller than our worst-case bound at $f = 0$. None of the prior works states a separate bound for this case, so for them we give the worst-case bound at $f = 0$; for [ELP24] and [EKL25] this coincides with the number of rounds their protocols take on unanimous inputs. [$\mathcal{P}$]Derived from their analysis for the version with termination certificates, whose expectation their Theorem 3 does not quantify; at coin agreement probability $p = \frac{1}{2}$.

	Setup	Adversary	Domain
This work	**sig. PKI, honest VRF keys**	**adaptive**	**multi-valued**
[CFLZ26] with [KK06a]	sig. PKI, honest VRF keys	adaptive	multi-valued
[ELP24]	sig. PKI, ideal coin	strongly adaptive[◇]	binary
[EKL25]	sig. PKI, ideal VRF (ROM)	adaptive	binary

Table 2: Assumptions and model of the BA protocols of Table 1. [◇]Strongly adaptive: upon corrupting a party, the adversary may delete or replace the messages that this party sent in the same round and that have not yet been delivered. The other rows assume, as is standard for VRF-based elections, that a message cannot be withdrawn once sent.

the highest grade *decides* (i.e., that value will represents its output) while the others carry their value into the next phase. The second step is a leader election, whose purpose is to bring together parties that have different values. Every party that is not confident in its value (low grade in the previous step) adopts the value proposed by the leader. If the leader is honest, it proposes the same value to every party, so all honest parties end up holding the same value, and the next graded agreement makes them all decide. A standard way to elect the leader uses a verifiable random function (VRF). Every party evaluates its VRF on the identifier of the phase and sends the result, and the party with the smallest value becomes the leader. A party that is honest at the start of the election holds the smallest value with probability at least $(n-t)/n > \frac{1}{2}$, so the leader is honest, and the parties reach agreement, with constant probability in each phase.

This approach must be implemented while guaranteeing the following. First, a corrupted leader must not be able to override a value that all honest parties already share otherwise Validity fails. Second, once the leader is known, an adaptive adversary corrupting it cannot have it propose different values to different parties.

The protocol of Katz and Koo [KK06a] solves the first problem by running the graded agreement *before* the leader election in each iteration. A party that obtains the highest grade decides on its

value; a party with an intermediate grade keeps its value; and only the remaining parties adopt the leader's value. In particular, if all honest parties start an iteration with the same value, they all decide on it before the election, and a corrupted leader cannot change it. The cost of this approach is that even if an honest leader is elected, the parties still need to go to the next phase and run the graded consensus to terminate. As we explain below, this costs a factor of two once the compiler of [CFLZ26] is applied.

Our approach. We avoid running the graded agreement before the election by changing what the leader proposes. The problem with proposing a single value is that nobody can check it: a corrupted leader can propose any value it likes, including one that no honest party holds. In our protocol the leader instead proposes an *explanation* about how a value is computed, in a form that it cannot be falsified. Every party computes a *certified vector* with one entry per party. The entry of P_j is either a value x_j together with P_j's signature on x_j, or $\perp$ together with public evidence that P_j misbehaved: two different values signed by P_j, or signed accusations against P_j from $t+1$ distinct parties, stating that P_j sent them nothing. Anyone can verify such a vector. After the election, every party takes the vector it received from the leader and, if the vector verifies, adopts the most frequent value among its entries different from $\perp$; otherwise it keeps its own value.

A corrupted leader cannot alter the entry of an honest party, since that would require forging its signature, and it cannot replace it with $\perp$ either: an honest party never signs two different values, and it is accused by at most the t corrupted parties. Every vector that verifies therefore contains the true value of every honest party, and the leader's only freedom lies in the at most t entries of corrupted parties. Since the honest parties are a majority, this freedom never suffices to outvote a value that all honest parties share: if all honest parties hold v, at least $n - t > t$ entries equal v, so every party adopts v, whoever the leader is and whichever valid vector it sends. Validity therefore no longer requires running the graded agreement before the election, and we can run it right after. This becomes important as it allows us to save rounds when the leader is honest. Indeed, in this case, all honest parties enter the graded agreement with the same value, so they all decide at the end of that same phase, rather than one iteration later as in [KK06a]. As we explain below, this halves the number of phases once the compiler of [CFLZ26] is applied.

Computing the certified vectors. Every party computes its certified vector with a two-round building block that we call *certified crusader interactive consistency* (CCIC). In the first round, every party signs its current value and sends it to all parties. In the second round, every party forwards the signed values it received, at most two per party, and sends a signed accusation against every party from which it received none. The entry of P_j is then $\perp$ if the party holds evidence against P_j, and the value signed by P_j otherwise (evidence takes priority in case of conflicting certificates). These vectors then can expose a corrupted party that signs different values. As in crusader agreement, from which the building block takes its name, two honest parties never hold different values in the same entry, although one of them may hold $\perp$ while the other holds a value.

Electing the leader after the proposals are fixed. Above we mentioned a second critical aspect is related to the fact that an adversary may corrupt a leader after its election and change its proposal. We avoid this by fixing the proposals before the leader becomes known. In the third round of a phase, every party sends its certified vector to all parties, and only in the fourth round is the leader elected. A leader that was honest at the start of the election was therefore honest when it sent its vector, which has already reached every party unchanged, so all honest parties adopt the same value. Hence, corrupting the leader after the election becomes useless. Due to the pseudorandomness and uniqueness properties of the VRF, we can argue that in each election, an honest leader is elected with probability at least $(n - t)/n$. We note that this all argument holds assuming that the adversary cannot retract the VRF value of an honest party after seeing it and

corrupting, which is a standard assumption for VRF-based elections [ACD+19, EKL25].

Making the final decision. After the first four rounds of a phase, all honest parties hold the same value whenever the leader was honest, and also whenever they already held the same value at the start of the phase. However, no party can tell whether this has happened, so no party can safely decide to output. To solve this, we let parties run a graded agreement that must guarantee two things: if all honest parties hold the same value, they all obtain grade 1 and decide; and if some honest party obtains grade 1 on a value, every other honest party leaves the graded agreement with that same value, so that a decision by one honest party can never be contradicted later.

We design such a graded agreement protocol in three rounds. In the first, every party signs its value and sends it to all parties. In the second, a party that receives signed values on the same v from $n - t$ distinct parties holds an *input certificate* for v, and sends it together with a signed *attestation* of v. In the third, a party that has seen input certificates for a single value y and attestations of y from $t + 1$ distinct parties sends a *justified vote* for y, consisting of its signature on y together with an input certificate and $t + 1$ attestations for y. A party that receives justified votes for y from $t + 1$ distinct parties obtains y with grade 1 and decides; otherwise, it takes the value of a justified vote it received, if any, with grade 0, and carries it into the next phase.

We rely on these two thresholds to provide the two guarantees. The threshold $n - t$ ensures that, if all honest parties hold v, every party can certify v and no party can certify any other value, so all of them vote for v and obtain grade 1. The threshold $t + 1$ ensures that every set of attestations contains one from an honest party, which sent the corresponding input certificate to every party; hence, once an honest party votes for y, no justified vote for any other value can exist. If some honest party obtains grade 1 on y, at least one of the $t + 1$ votes it received comes from an honest party, which sent its vote to every party, so every honest party receives a justified vote for y and none for any other value, and adopts y. Since votes carry their own justification, a single vote suffices to bring a party to the right value.

The reason a single vote suffices is because every vote carries its own justification: a party that receives one justified vote can check, from the input certificate and attestations it contains, that y is the only value that can be safe, and adopt it.

A phase of $\pi_{\mathsf{rand\text{-}ba}}$ therefore takes 7 rounds: two for CCIC, one to send the certified vectors, one for the election and three for the graded agreement. A phase whose election returns an honest leader ends with every honest party deciding, which gives expected round complexity $7n/(n-t)+1 < 15$. In addition, a party that receives signatures from all n parties on the same value in the first round of a phase can decide on it right away, since every honest party then holds that value; with this *fast path*, when no party is corrupted and all inputs agree, every party decides after a single round and halts after the second, once it has sent its termination certificate.

Why $\pi_{\mathsf{rand\text{-}ba}}$ is not early-stopping. On its own, $\pi_{\mathsf{rand\text{-}ba}}$ gives no worst-case guarantee: each phase fails to elect an honest leader with constant probability, and nothing prevents this from happening in every phase. To bound round complexity in terms of f, we use the compiler of [CFLZ26]. The compiler takes as input a randomized BA protocol that proceeds in phases, and runs it phase by phase. At certain phase boundaries, it adds a *signaling round*, in which every party signs and sends ready if it has decided and unready otherwise. A party that has not decided and has seen no valid ready concludes that no honest party has decided, and replaces its value with a default one. If all honest parties do so, they become unanimous and decide in the next phase. A corrupted party can block this replacement by falsely signaling ready. However, signals are forwarded to all parties, and a party that signals ready is trusted only for a fixed number of phases after its first ready, which is long enough for every honest party to reach a decision if the signal was genuine. Hence each corrupted party can delay the termination by at most a constant number of phases, and the

compiled protocol terminates in $O(f)$ rounds.

The compiler works for any input protocol that satisfies the following four properties: (i) if a phase starts with all honest parties holding the same value, then within a bounded number of phases all honest parties decide on that value; (ii) in each phase, with constant probability, a random event (in our case, a good election) leads all honest parties to decide on the same value within a bounded number of phases; (iii) once some honest party decides on a value, all other honest parties decide on the same value within a bounded number of phases; and (iv) a party that has decided keeps running for a bounded number of rounds before halting. The round complexity of the compiled protocol depends on these bounds. For $\pi_{\mathsf{rand\text{-}ba}}$, properties (i) and (iii) hold with very small bounds: a phase that starts unanimous ends with every honest party deciding, as shown above, and once an honest party decides, the graded agreement brings every honest party to its value, so all of them decide in the next phase. We spend the last part of this section to instead explain how we obtain good bounds (much better than those of [KK06a, KK09]) for (ii) and (iv).

Property (ii): a good election leads to a decision within its own phase. The compiler replaces values with the default only at phase boundaries, and only at boundaries that do not separate a good election from the decision it causes, since a replacement in between could undo the agreement that the election has just produced. In [KK06a], the election comes after the graded agreement of its phase, so a good election and the resulting decision always lie in two consecutive phases. The compiler can therefore signal and replace values only at every second phase boundary, which doubles the number of phases, both in expectation and in the worst case. In $\pi_{\mathsf{rand\text{-}ba}}$, the election comes before the graded agreement of the same phase, so a good election and the resulting decision lie in the same phase, and the compiler can signal and replace values at every phase boundary.

Property (iv): a party that decides halts one round later. In the protocol of [KK06a], a decided party must keep running for almost a full additional iteration, because the others still need its signatures to finally decide their output and later terminate. In $\pi_{\mathsf{rand\text{-}ba}}$, a party that decides sends, in the next round, its signature on its output, which we call a *terminal certificate*, and then halts. The remaining parties use this certificate in its place wherever its signatures would otherwise be counted, including as its value in the next execution of CCIC. Once some honest party has decided, all honest parties hold the decided value, so the certificate of an honest party only ever stands for a value every honest party already holds, and the certificate of a corrupted party gives the adversary nothing it could not sign itself.

Putting it together. A phase takes seven rounds both in $\pi_{\mathsf{rand\text{-}ba}}$ and in the protocol of [KK06a], so our improvement does not come from shorter phases. It comes from the two properties above: we need half as many phases, and our parties halt sooner after deciding. Since a good election leads to a decision within its own phase, the compiler signals after every phase, so a compiled phase costs 8 rounds and each corrupted party delays termination by at most one phase. In the worst case, the compiled protocol therefore runs for $f + 2$ phases, against $2f + 3$ for the protocol of [KK06a], whose compiled phases cost 7.5 rounds on average, as the compiler signals only after every second phase. Since a party that decides halts one round later, the rounds that follow the last decision add only 2 to our bounds, against 7 for the protocol of [KK06a]. Altogether, we obtain expected round complexity $8n/(n - t) + 2 < 18$ and worst-case round complexity $8(f + 2) + 2 = 8f + 18$, against 37 and $15f + 29$ for the instantiation of [CFLZ26] with [KK06a].

A note on VRF key generation. Our analysis of the election uses only standard VRF properties, which is why it requires the VRF keys of all parties, including corrupted ones, to be generated honestly by the setup. This requirement can be traded for the random oracle model, as in [DGKR18, EKL25]. It suffices to replace the VRF by the ideal functionality $\mathcal{F}_{\mathsf{VRF}}$ of David et al. [DGKR18], which samples a fresh uniform output for every key and input, including keys generated by the

adversary, and accepts only that output as valid; they realize it in the random oracle model under the CDH assumption, using the 2-Hash-DH construction. With $\mathcal{F}_{\mathsf{VRF}}$, all VRF values are uniform and independent, the corrupted parties' included, so the idealized part of our analysis of the elections applies unchanged and no reduction to pseudorandomness is needed. One precaution is required: since $\mathcal{F}_{\mathsf{VRF}}$ lets the adversary generate and evaluate many keys before registering one, the election inputs must not be known when keys are registered, as otherwise a corrupted party could register a key whose values on the first elections are small. As in [DGKR18, GHM+17], it suffices that every election identifier includes a public random string sampled after the keys are registered.

Parallel BA and interactive consistency. In general, running m instances of a randomized BA protocol does not preserve its expected constant-round complexity, since the slowest of m independent instances needs $\Theta(\log m)$ times as many phases in expectation. Our approach is to let all instances share the same leader election execution, as [KK09] also does, following [FG03]. Each instance runs its own CCIC and graded agreement, but in each phase every party evaluates its VRF only once, and the resulting leader serves all m instances. A good election is therefore good for all instances at once, so in the first phase whose election returns an honest leader, every instance that has not yet decided does so. The parallel execution thus needs no more phases than a single instance, and its expected round complexity is that of a single instance (and worst-case complexity is trivially preserved). Interactive consistency (IC) follows by running n such BA instances in parallel, one per party, each preceded by a round in which a party sends its input. If no party is corrupted, every BA instance starts with all parties holding the same value in every instance, and every party outputs its whole vector after two rounds.

2 Preliminaries and Model

Notation. We let κ denote the security parameter and, for $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. A function $\mu : \mathbb{N} \to [0,1]$ is *negligible* if $\mu(\kappa) < 1/q(\kappa)$ for every polynomial q and all sufficiently large κ; we write $\mathsf{negl}(\kappa)$ for an unspecified negligible function. We write PPT for probabilistic polynomial-time algorithms, and $\{0,1\}^*$ for the set of finite binary strings. For a distribution $\mathcal{D}$ we write $x \leftarrow \mathcal{D}$ to denote sampling x according to $\mathcal{D}$, and for a finite set S we write $x \xleftarrow{\mathrm{R}} S$ to denote sampling x uniformly from S. Throughout, n denotes the number of parties $P_1, \ldots, P_n$, t the corruption threshold, and $f \leq t$ the number of parties actually corrupted in a given execution.

2.1 Model

Communication model. We consider a synchronous network of n parties $P_1, \ldots, P_n$ that communicate over a complete network of pairwise authenticated point-to-point channels, so the adversary can neither modify nor forge a message exchanged between two honest parties, and a receiver always learns the identity of the sender. There is no broadcast channel, and an instruction to send a message to all parties means n point-to-point sends, one of which is to the sender itself. Communication proceeds in rounds. In each round, parties first send their messages and then receive all messages sent to them in that round, so that every message sent in round r is delivered before the beginning of round $r+1$. Parties have synchronized clocks and know the current round number.

Adversarial model. All our protocols are analyzed against adaptive, rushing, malicious, PPT adversaries corrupting at most t parties. *Adaptive*: the adversary chooses whom to corrupt as the execution proceeds and learns a party's internal state on corrupting it. Corruptions are irreversible, so a party honest at the start of a round was honest at the start of every earlier round, which several of our arguments use. A party honest at the start of a round sends its prescribed messages for

that round and corrupting it thereafter neither retracts nor alters them, but the adversary may send further messages on its behalf in that same round. *Rushing*: in each round the adversary sees the messages the honest parties send before choosing those of the corrupted parties. *Malicious*: corrupted parties may deviate arbitrarily from the prescribed protocol. *PPT*: the adversary is polynomially bounded, as we rely on signatures and on verifiable random functions.

Throughout we assume an honest majority, $t < \frac{n}{2}$, which is necessary, since for $t \geq \frac{n}{2}$ and a domain with at least two values, no protocol satisfies Definition 2.1 below, whatever setup is assumed, by the standard partitioning argument.

Security model. We use standard property-based security definitions. Each primitive is specified by an explicit list of properties, and a protocol realizes it if all of them hold against the stated class of adversaries. Since all our building blocks are computational, every property and bound we state, is required to hold only with overwhelming probability. That is, except with negligible probability in κ, where the probability is over the coins of the setup, of the honest parties and of the adversary; we do not repeat this, and a fairness p attained up to a negligible error is read as p.

2.2 Building Blocks

We first define the problem we solve, and then the three primitives from which Section 3 builds a protocol for it. Oblivious leader election and graded agreement are classical; the third combines Dolev's crusader relaxation with public certification. Constructions for the three primitives are deferred to Appendix A.

Definition 2.1 (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 and halts.*
- ***Agreement.*** *For every two honest parties $P_i, P_{i'}$ that output, $o_i = o_{i'}$.*
- ***Validity.*** *If every honest party has the same input $v \in \mathcal{M}$, then every honest party that outputs outputs $o_i = v$.*

Here and throughout, honest *means never corrupted in the execution, so the honest parties number at least $n - t$ and are contained in the set of parties honest at the start of any given round.*

Definition 2.2 (Broadcast)**.** *An n-party protocol π_{bb} with domain $\mathcal{M}$ and a designated* sender *P_s is a Byzantine broadcast protocol if P_s has input $v \in \mathcal{M}$, the other parties have none, each honest party P_i outputs a value $o_i \in \mathcal{M}$, and the following properties hold.*

- ***Termination.*** *Every honest party eventually outputs and halts.*
- ***Agreement.*** *For every two honest parties $P_i, P_{i'}$ that output, $o_i = o_{i'}$.*
- ***Validity.*** *If P_s is honest, then every honest party that outputs outputs $o_i = v$.*

We write $\mathcal{M}$ for the agreement domain whenever it is left abstract, and instantiate it with $\{0,1\}^*$ in the protocols of Section 3 and Appendix A; the symbol $\perp \notin \mathcal{M}$ is reserved throughout for the absence of a value. It is never an admissible input, though the outputs of some building blocks below range over $\mathcal{M} \cup \{\perp\}$, as their definitions make explicit.

The definition follows Katz and Koo [KK06b], who adapt it from Fitzi and Garay [FG03], with three changes: their election runs in two phases and needs a leader honest at the end of the first, while we need one honest only at the start of the invocation, which a one-round election such as that of Appendix A can provide; we add the a-priori round bound R_{ole}; and we state fairness over a sequence of invocations sharing one setup rather than for a single invocation.

Definition 2.3 (Oblivious Leader Election)**.** *An n-party protocol π_{ole} is a leader-election protocol with fairness p if for every PPT adversary corrupting at most t parties 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.*** *For any polynomial number k of invocations that share one setup, used by no other protocol, and are run by every honest party in the order of a list of distinct session identifiers fixed before the setup, the probability that none of them is* good*, that is, one in which every honest party outputs the index of one and the same party honest at its start, is at most $(1-p)^k$.*

Summing the bound over k, the first good invocation has expected index at most $1/p$; the tail itself is what lets an execution be cut after polynomially many invocations at a negligible cost.

Fairness asks for a leader honest at the start of the invocation rather than at its end: an adaptive adversary can corrupt the elected party the moment the election's messages are out, and Section 3 arranges that everything it needs from the leader has been sent by then.

Fairness is stated over a sequence of invocations because Section 3 invokes the election once per phase under one setup, and under a shared setup a bound for each invocation given the earlier ones can fail. The identifiers must be distinct, as two invocations under the same identifier elect the same leader, and fixed before the setup, so that the adversary cannot pick them after learning the keys of the parties it corrupts. Their order must be fixed as well, or an adversary that knows those keys could run first an identifier at which the parties it corrupts are favoured. Nor may another protocol use the setup, as it could reveal the randomness of an invocation before that invocation starts.

Interactive consistency can be stated with a designated sender, as broadcast, or with all parties acting as senders at once; Section 3 uses the latter, so we define it that way.

Definition 2.4 (Certified crusader interactive consistency)**.** *A* certified crusader interactive consistency *(CCIC) protocol is a triple $(\mathsf{Setup}_{\mathsf{ccic}}, \pi_{\mathsf{ccic}}, \mathsf{Ver}_{\mathsf{ccic}})$, where the setup algorithm $\mathsf{Setup}_{\mathsf{ccic}}(1^\kappa)$ outputs a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for every $i \in [n]$, hands sk_i to P_i and publishes $\mathsf{pp} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$; π_{ccic} is an n-party protocol with session identifier sid in which each party P_i, on input pp, sk_i and $b_i \in \{0,1\}^*$, outputs a pair $(\boldsymbol{v}_i, \omega_i)$ with $\boldsymbol{v}_i \in (\{0,1\}^* \cup \{\bot\})^n$; and the deterministic verification algorithm $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}, \omega)$ outputs a bit. For every PPT adversary corrupting at most t parties, the following conditions are satisfied:*

- ***Termination.*** *There exists an a-priori-known round R_{ccic} such that every honest party terminates within R_{ccic} rounds.*
- ***Completeness.*** *Every party P_i honest at the end of the protocol outputs a pair $(\boldsymbol{v}_i, \omega_i)$ such that $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}_i, \omega_i) = 1$.*
- ***Consistency.*** *For every two honest parties $P_i, P_{i'}$ and every $j \in [n]$ with $\boldsymbol{v}_i[j] \neq \bot$ and $\boldsymbol{v}_{i'}[j] \neq \bot$, it holds that $\boldsymbol{v}_i[j] = \boldsymbol{v}_{i'}[j]$.*
- ***Validity.*** *If $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}, \omega) = 1$ for some honest P_i, then the j-th coordinate of $\boldsymbol{v}$ equals P_j's input for every honest P_j.*
- ***Transferability.*** *If $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}, \omega) = 1$ for some party P_i honest at the end of the protocol, then $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, j, \boldsymbol{v}, \omega) = 1$ for every honest P_j.*

Validity and transferability range over every pair $(\boldsymbol{v}, \omega)$ produced in the execution, by the adversary included, during or after the protocol.

Consistency is the crusader guarantee of Dolev [Dol82], imposed coordinate by coordinate, two honest parties that both hold a value for P_j holding the same one, while others may hold $\bot$.

Validity constrains only honest coordinates, which is all that can be asked, since a corrupted party has no well-defined input; together with completeness it gives the usual guarantee of interactive consistency there. Consistency gives the primitive its name; the protocol of Section 3 reads one vector at a time and so relies only on completeness, validity and transferability.

Definition 2.5 (2-Graded Agreement)**.** *An n-party protocol π_{2ga} is a* 2-graded agreement *(2GA) protocol if each party P_i holds an input $z_i \in \{0,1\}^*$ and outputs a pair $(y_i, g_i) \in (\{0,1\}^* \times \{0,1\}) \cup \{(\perp, 0)\}$, such that for every PPT adversary corrupting at most t parties the following conditions are satisfied:*

- ***Termination.*** *There exists an a-priori-known round R_{2ga} such that every honest party terminates within R_{2ga} rounds.*
- ***Graded validity.*** *If all honest parties have input $v \in \{0,1\}^*$, then every honest party outputs $(v, 1)$.*
- ***Graded consistency.*** *If some party honest at the end of the protocol outputs $(y, 1)$, then every honest party that produces an output outputs $(y, \cdot)$.*

2.3 Cryptographic Primitives

Digital signatures. We rely on a digital signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$. On input 1^κ, the key-generation algorithm Gen outputs a signing key and a verification key $(\mathsf{sk}, \mathsf{pk})$. Given a message m, the signing algorithm $\mathsf{Sign}_{\mathsf{sk}}(m)$ outputs a signature σ, and the deterministic verification algorithm $\mathsf{Ver}_{\mathsf{pk}}(m, \sigma)$ outputs 1 if σ is a valid signature on m with respect to pk, and 0 otherwise. We assume that Σ is existentially unforgeable under chosen-message attacks (EUF-CMA), that is, no PPT adversary can produce, with non-negligible probability, a valid signature on a message it has not asked to be signed, even after adaptively obtaining signatures on messages of its choice.

Signature public-key infrastructure. In the *signature-PKI model*, each party P_i has a registered verification key pk_i that is available to all parties. For every honest P_i the key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ is generated honestly by $\mathsf{Gen}(1^\kappa)$, and the signing key sk_i is known only to P_i. For corrupted parties, the adversary may choose and register the verification keys after observing those of the honest parties; this relaxation is specific to the signature PKI, and the VRF PKI defined below does not grant it.

Verifiable random functions. A verifiable random function [MRV99] is a keyed function whose owner can prove that it evaluated the function correctly, but whose values remain unpredictable to everyone else.

Definition 2.6 (Verifiable Random Function)**.** *A verifiable random function with range $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ is a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{VRF}, \mathsf{Ver})$, where $\mathsf{Gen}(1^\kappa)$ outputs a key pair $(\mathsf{sk}, \mathsf{pk})$; the evaluation algorithm $\mathsf{VRF}_{\mathsf{sk}}(x)$ maps an input $x \in \{0,1\}^*$ to a pair (ρ, ω) consisting of a value $\rho \in \mathcal{Y}_\kappa$ and a proof ω; and the verification algorithm $\mathsf{Ver}_{\mathsf{pk}}(x, \rho, \omega)$ outputs 1 if ρ is the value of the function keyed by pk at x and ω witnesses this, and 0 otherwise. The following properties are required.*

- ***Provability.*** *For every $(\mathsf{sk}, \mathsf{pk})$ in the support of $\mathsf{Gen}(1^\kappa)$ and every $x \in \{0,1\}^*$, if $(\rho, \omega) = \mathsf{VRF}_{\mathsf{sk}}(x)$ then $\mathsf{Ver}_{\mathsf{pk}}(x, \rho, \omega) = 1$.*
- ***Uniqueness.*** *For every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}(1^\kappa)$ outputs a verification key pk, an input x, and two pairs (ρ, ω) and (ρ', ω') with $\rho \neq \rho'$ such that $\mathsf{Ver}_{\mathsf{pk}}(x, \rho, \omega) = \mathsf{Ver}_{\mathsf{pk}}(x, \rho', \omega') = 1$ is negligible in κ. In particular, this is required also of keys pk outside the support of Gen.*

– ***Pseudorandomness.*** *For every PPT adversary $\mathcal{A}$, consider the experiment in which $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$; $\mathcal{A}$ is given pk and oracle access to $\mathsf{VRF}_{\mathsf{sk}}(\cdot)$, and outputs a challenge input $x^* \in \{0,1\}^*$ on which it has not queried the oracle; a bit $b \xleftarrow{\mathrm{R}} \{0,1\}$ is sampled, and $\mathcal{A}$ is given ρ^*, which is the value component of $\mathsf{VRF}_{\mathsf{sk}}(x^*)$ if $b = 0$ and a uniform $\rho^* \xleftarrow{\mathrm{R}} \mathcal{Y}_\kappa$ if $b = 1$; $\mathcal{A}$ then continues with oracle access on inputs other than x^* and outputs a guess b'. The probability that $b' = b$ is at most $\frac{1}{2} + \mathsf{negl}(\kappa)$.*

The range is totally ordered, with efficiently computable ranks, as for bit strings read as binary numbers. We say that the range is of exponential size *if $|\mathcal{Y}_\kappa| = 2^{\Omega(\kappa)}$.*

We overload Gen and Ver for both signatures and verifiable random functions, as the two are never used with the same key and the scheme in question is always clear from the context; likewise, the value of $\mathsf{VRF}_{\mathsf{sk}}(x)$ is understood as the pair (ρ, ω) when both components are needed and as ρ alone otherwise.

VRF public-key infrastructure. In the *VRF-PKI model*, the setup samples $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ independently for every $i \in [n]$, publishes $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$, and hands sk_i to P_i. A corrupted party learns its own sk_i, but neither chooses it nor selects among candidate key pairs. We refer to the setting in which both registrations are in place as the *signature-PKI and VRF-PKI model*.

2.4 The Randomized Early-Stopping Compiler

One method to achieve expected-constant-round, early-stopping Byzantine agreement is to apply the compiler of [CFLZ26]. We do so in Section 4 on our expected-constant-round protocol of Section 3. We recall the compiler here, in the form we use. It takes a phase-based randomized Byzantine agreement protocol π_{rand} satisfying four structural properties and returns one that is simultaneously expected-constant-round and early-stopping.

We use the compiler's variant in the signature-PKI model. Unlike the setup-free variant of [CFLZ26], which coordinates the parties through parallel crusader agreement, it places no constraint of its own on the corruption threshold and spends one round on coordination instead of two.

The compiler $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand}})$ runs π_{rand} phase by phase, starting from the party's own input, and attempts an *injection* — an overwrite of an uncommitted party's local value with a fixed default $\phi \in \mathcal{M}$ — only at the end of every L-th phase, for an integer *injection period* $L > \Delta_2$. At the end of each such phase every party signs and sends to all parties a signal, ready if its copy of π_{rand} has committed and $\mathsf{unready}$ otherwise, bound to the session identifier of the invocation and to the phase index; at the end of the other phases nothing is sent. The $\mathsf{ready}/\mathsf{unready}$ signals, including any two contradictory ones from the same party for the same phase, are forwarded in the background, so that a signal seen by one honest party is seen by all of them one round later, and any party detected to have equivocated is not admitted to the set S below. Each party keeps the set S of parties not known to have equivocated from which it has seen a valid ready, together with the earliest phase of such a signal from each of them. From phase $\Delta_1 + 1$ on, a party that has not committed and has $S = \emptyset$ at an injection phase concludes that no honest party has committed, and injects. Injections drive the worst-case bound. Once all honest parties inject they are unanimous, and (P2) then forces them to commit. A corrupted party can delay an injection by signalling ready falsely, but it is dropped from S once Δ_3 phases have passed since its earliest ready, so each corruption costs at most $\lceil \Delta_3 / L \rceil$ injection attempts, which is the sole source of the dependence on f. A genuine ready blocks the injection for exactly as long as (P4) needs, so a converging run is never disturbed; and since injections fall only at the end of a block of L phases, the first $\Delta_2 + 1$ phases of every block form an undisturbed window for (P3). A party outputs as soon as π_{rand} fixes its local output, sends

its ready for the current phase in the next round, and halts at the end of that round or once Δ_0 further rounds of π_{rand} have been executed, whichever is later.

Properties needed for the randomized procedure. Each party running π_{rand} maintains a local value, a monotone *commit* flag, and a local output. A window of consecutive phases is *undisturbed* if no injection takes place at a boundary interior to it; local values at the first phase of a window are unrestricted. The required properties are the following.

- **(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 $committed_i$; upon setting $committed_i$, 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 $committed_i$ 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 signalling 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 Δ_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.
- **(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.

With $\Delta_2 = 0$, Property (P3) has all honest parties commit within the very phase whose randomness they use, in which case $L = 1$ is admissible and the compiler may inject at every phase boundary; and with $\Delta_0 = 0$, Property (P1) has a party halt as soon as it commits.

The guarantee established in [CFLZ26] is the following.

Theorem 2.7. *Assume EUF-CMA signature schemes exist. Let π_{rand} be a computationally secure phase-based randomized Byzantine agreement protocol tolerating fewer than t_{rand} corruptions and satisfying Properties (P1)–(P4), and let $L > \Delta_2$ be an integer. Then $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand}})$ 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*

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

and its worst-case round complexity is

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

where $f \leq t$ is the actual number of corruptions, $c, p, \Delta_0, \Delta_1, \Delta_2, \Delta_3$ are the constants of Properties (P1)–(P4), and $\Delta_0^* := \max(\Delta_0, 1) + \lceil \Delta_0/(Lc) \rceil$.

A run of M phases costs at most $M(c + \frac{1}{L})$ rounds, each phase of π_{rand} taking c rounds, a signalling round following every L-th phase and no other, and the background forwarding running in parallel with the main execution. The trailing term Δ_0^* exceeds Δ_0 because the trailing rounds of a committer are rounds of π_{rand}, and the compiled execution interposes a signalling round at every injection phase they cross. It is at least one, as a committer sends its ready in the round after its output is fixed.

3 Byzantine Agreement and Broadcast in Expected Constant Rounds

We compose the three building blocks of Section 2.2 — an oblivious leader election protocol π_{ole}, a certified crusader interactive consistency protocol π_{ccic} with verification predicate $\mathsf{Ver}_{\mathsf{ccic}}$, and a 2-graded agreement protocol π_{2ga} — into a Byzantine agreement protocol for $t < n/2$ with expected constant round complexity. The two signature-based blocks are the constructions of Appendix A, π_{ccic} being the protocol of Figure 3, with $R_{\mathsf{ccic}} = 2$, and π_{2ga} that of Figure 4, with $R_{\mathsf{2ga}} = 3$; the protocol uses their defining properties and also, in a white-box manner, their code, which is written out in our protocol. The election is any protocol satisfying Definition 2.3, with fairness p and round complexity R_{ole}, and is invoked as a black box; the protocol of Figure 2 has $R_{\mathsf{ole}} = 1$ and $p = (n - t)/n$. Two things need the code: the fast path, which reads the signed inputs of the first round of π_{ccic}, and the rule that lets a party that has halted still be counted, through the certificate it leaves behind, wherever π_{ccic} or π_{2ga} counts its signature on a value. One round of the phase belongs to the protocol itself rather than to any of the three building blocks, namely the dissemination of a party's π_{ccic} output, which is sent in its own round, strictly before the round in which π_{ole} is invoked. Each is invoked once per phase, under a session identifier $\mathsf{sid}^r_{\mathsf{ccic}}$, $\mathsf{sid}^r_{\mathsf{ole}}$ or $\mathsf{sid}^r_{\mathsf{2ga}}$ formed from the identifier sid of $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$, the phase index r and the name of the primitive. We take sid to be fixed before the setup, as Definition 2.3 requires of the election identifiers.

3.1 Expected-Constant-Round Byzantine Agreement

3.1.1 The Protocol

As the basis of our expected-constant-round, early-stopping Byzantine agreement, we present a round-efficient, expected-constant-round protocol that satisfies the properties required by the compiler of [CFLZ26]. It proceeds in phases, each of which tries to make the honest parties agree and succeeds with constant probability. A party carries a current value b into the phase and leaves with a possibly revised one, together with evidence if the phase happened to succeed.

The difficulty a randomized protocol has to solve is that honest parties may enter a phase holding different values, and the natural remedy — adopt an elected leader's view of all the parties' local values — fails in two ways. A corrupted leader may override a value all honest parties share, breaking validity, and a leader that sends its view only once elected may be corrupted at that moment and propose different values to different parties. The second issue is resolved by making all parties send their vector of values *before* a leader is elected, so that corrupting the leader after the election is useless — the adversary cannot retroactively change what that party sent before it was corrupted. To avoid the first issue, parties use CCIC to distribute their current local values. At the end of CCIC, every honest party has, for every j, either a certificate for P_j's local value, or $\perp$. Thus, no corrupted leader can make a party that believes in P_j's local value change its belief. In the first round every party signs and sends its value, and in the second it forwards the signed values it received, at most two per party, and accuses every party it heard nothing from; it then

outputs a vector $\boldsymbol{v}$ of the values it learned, with a proof ω that makes the pair's validity publicly verifiable. Any accepted vector carries the true input of every honest party, so a corrupted party may be exposed but cannot make an accepted vector lie about an honest party.

Each party then sends its pair to everyone, and the leader election runs in the following round. Because the pair goes out before the leader is known, the proposal is fixed before the adversary can learn whom to corrupt, and the leader's pair does not need to be fetched once the leader ℓ is revealed. A party that receives a valid pair from P_ℓ adopts the mode of its non-$\perp$ coordinates (ties broken by a fixed order on $\{0,1\}^*$), and one that does not falls back on the value b it started the phase with. When the elected leader was honest at the start of the election, which happens with probability at least p, every honest party accepts the same pair, by transferability, and so adopts one and the same value.

The phase closes with graded agreement on the adopted value. Its two grades separate "the honest parties certainly agree now" from "they may not yet". Grade 1 licenses a party to output and stop; grade 0 still leaves its value aligned with everyone else's, which is what carries the agreement from one phase into the next.

There are two routes to termination. The conventional one is grade 1 in π_{2ga}; the other is the fast path, taken when a party holds, at the end of the first round of a phase, signed inputs of all n parties on one and the same value. A party that halts leaves behind a *terminal certificate*, its signature on its output y. When an honest party halts on y, every honest party's local value is already y. If the cause of the halt was grade 1 in π_{2ga}, then by graded consistency every honest party also leaves that invocation of π_{2ga} with y. Else the fast path was taken, and all honest parties must also have signed and sent y as their input in that phase. From then on every phase is unanimous on y, so the honest values never change. No party ever outputs solely because it received a terminal certificate; its one use is to stand in for the signature of a party that is no longer there, wherever a later step would have counted it, which keeps the thresholds reachable after its issuer has halted. A certificate of an honest party thus only stands in for a signature on the value every honest party already holds, and one of a corrupted party adds nothing, since the adversary could sign the replaced message itself. Figure 1 gives the protocol.

Protocol $\pi_{\mathsf{rand\text{-}ba},\, t<\frac{n}{2}}$

Below is the code for party P_i in the signature-PKI and VRF-PKI model, for an invocation with session identifier sid.

A *terminal certificate* of P_j on y is a valid signature of P_j on $(\mathsf{sid}, \mathsf{term}, y)$. A certificate is never the sole reason to output or to halt; its only use is the following.

Substitution rule. Wherever π_{ccic} or π_{2ga} would count a signature of P_j on one of its messages towards a threshold or as the proof of a coordinate, or check in step 2 whether it holds one, and the party holds no such signature, a terminal certificate of P_j on y that the party holds counts in its place as a signature of P_j on the corresponding message for y, and is carried inside the object it is counted in, so that the object remains publicly verifiable, its verification accepting the certificate in that place. Counted as a vote of P_j in step 7, it makes a justified vote only together with an input certificate and an attestation certificate for y under $\mathsf{sid}^r_{\mathsf{2ga}}$, which the party counting it attaches from those it holds.

Initialization: on input $(\mathtt{input}, x)$, set $b := x$. Every terminal certificate received is kept.

Main execution

Repeat the following for phase $r = 1, 2, \ldots$, setting $E := \emptyset$, $F := \emptyset$ and $M := \emptyset$ at its start. Every signed input of $\mathsf{sid}^r_{\mathsf{ccic}}$ received is added to E and every accusation of $\mathsf{sid}^r_{\mathsf{ccic}}$ to F.

Certified crusader interactive consistency, Figure 3, opened up:

1. Compute $\sigma_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}^r_{\mathsf{ccic}}, \mathsf{input}, b)$, add (i, b, σ_i) to E, and send it to all parties.
2. For every party P_j:
   - if E contains a signed input of P_j, send one to all parties, and if it also contains one on a different value, send that one too;
   - otherwise, compute $\sigma_{i,j} := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}^r_{\mathsf{ccic}}, j, \mathsf{accuse})$, add it to F, and send $((\mathsf{sid}^r_{\mathsf{ccic}}, j, \mathsf{accuse}), \sigma_{i,j})$ to all parties.

   Then, once the messages of this round have been processed, for every party P_j:
   - if E contains signed inputs s, s' of P_j on different values, set $\boldsymbol{v}[j] := \perp$ and $\omega[j] := (s, s')$;
   - otherwise, if F contains accusations against P_j from at least $t+1$ distinct parties, set $\boldsymbol{v}[j] := \perp$ and set $\omega[j]$ to one accusation from each of $t+1$ of those parties;
   - otherwise, if E contains a signed input (j, x_j, σ_j) of P_j, set $\boldsymbol{v}[j] := x_j$ and $\omega[j] := (j, x_j, \sigma_j)$;
   - otherwise, set $\boldsymbol{v}[j] := \perp$ and $\omega[j] := \perp$.

   Finally, the pair $(\boldsymbol{v}, \omega)$ is the output of π_{ccic}.

Dissemination:

3. Send $(\boldsymbol{v}, \omega)$ to all parties.

Oblivious leader election:

4. Invoke π_{ole} with session identifier $\mathsf{sid}^r_{\mathsf{ole}}$, obtaining ℓ. Then let $(\boldsymbol{v}', \omega')$ be the pair received from P_ℓ in step 3, least under a fixed order on strings if that round carried more than one, and none if it carried none; pairs received in any other round are ignored. If $(\boldsymbol{v}', \omega')$ exists and $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}^r_{\mathsf{ccic}}, i, \boldsymbol{v}', \omega') = 1$, set z to the mode of the non-$\perp$ coordinates of $\boldsymbol{v}'$, ties broken by a fixed order on $\{0,1\}^*$, or to b if all of them are $\perp$; otherwise set $z := b$.

Two-graded agreement, Figure 4, opened up:

5. Compute $\sigma_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}^r_{\mathsf{2ga}}, \mathsf{input}, z)$ and send (z, σ_i) to all parties.
6. For every party P_j, add (j, v, σ) to M if (v, σ) is the pair received from P_j in the previous round for which σ is a signed input of P_j on v, least under a fixed order on strings if more than one arrived. If M contains entries for one and the same value $v \in \{0,1\}^*$ from at least $n-t$ distinct parties, let C be a set of $n-t$ of their signed inputs, compute $\alpha_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}^r_{\mathsf{2ga}}, \mathsf{attest}, v)$, and send (v, C, α_i) to all parties.
7. Let V be the set of values for which an input certificate was received in the previous round. If $V = \{y\}$ for a single value y, and attestations for y were received in the previous round from at least $t+1$ distinct parties, assemble an input certificate C_y and an attestation certificate A_y for y from those messages, compute $\beta_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}^r_{\mathsf{2ga}}, \mathsf{vote}, y)$, and send the justified vote (y, C_y, A_y, β_i) to all parties. Then, if justified votes for one and the same value $y \in \{0,1\}^*$ are received in this round from at least $t+1$ distinct parties, set $(y, g) := (y, 1)$; otherwise, if a justified vote for some value was received in this round, set $(y, g) := (y, 0)$ for the value y of the one least under a fixed order on strings; if not, set $(y, g) := (\perp, 0)$. The pair (y, g) is the output of π_{2ga}. Set $b := y$ if $y \neq \perp$.

Termination

- *Grade one.* If $g = 1$, output $(\mathtt{output}, y)$, send to all parties in the next round its terminal certificate on y, and halt.
- *Fast path.* At the end of step 1, if E contains signed inputs of all n parties on one and the same value u, output $(\mathtt{output}, u)$, send to all parties in the next round its terminal certificate on u, and halt.

Figure 1: Randomized Byzantine agreement for $t < n/2$.

3.1.2 Analysis

Theorem 3.1 (Byzantine agreement for $t < n/2$)**.** *Assume EUF-CMA signature schemes exist and let π_{ole} be a leader-election protocol with fairness p and round complexity R_{ole}. Then, for any PPT adversary corrupting at most $t < n/2$ parties, protocol $\pi_{\mathsf{rand\text{-}ba},\, \mathsf{t}<\frac{\mathsf{n}}{2}}$ is a Byzantine agreement protocol in the signature-PKI model together with the setup of π_{ole}, with expected round complexity*

at most $(6 + R_{\mathsf{ole}})/p + 1$.

Proof. By Theorems A.2 and A.3, the two blocks written out realize Definitions 2.4 and 2.5, in $R_{\mathsf{ccic}} = 2$ and $R_{2\mathsf{ga}} = 3$ rounds, and we use them through those definitions except where the fast path and the substitution rule read their code. Their guarantees hold per invocation up to a negligible error, and the execution is cut after polynomially many phases, as explained at the end of the proof, so a union bound over its invocations keeps the total error negligible.

Three facts carry the proof.

Fact 1. Unanimity causes honest parties to halt soon with the same value. Suppose every honest party still running begins a phase with the same value v, and every certificate left by an honest party that has halted carries v. Then every honest party still running outputs v in that phase, on the fast path or with grade 1 at its end. A party that fires the fast path does so on a value all n parties signed, hence on v. In any pair a party accepts, the coordinates of the honest parties equal v, by validity of π_{ccic} for those still running, and for a halted one because its single terminal certificate, which carries v, is the only signature that can stand for it, and the honest parties still running hold that certificate and so do not accuse it; being at least $n - t > t$ of the n coordinates, they make v the mode. A party whose pair is invalid or missing falls back on $b = v$. So every honest party enters step 5 with $z = v$. In $\pi_{2\mathsf{ga}}$ the substitution rule supplies a signed input and an attestation for v from every halted honest party, so every running honest party forms an input certificate for v and holds attestations for v from all honest parties, at least $t + 1$ of them; these complete the vote the rule supplies for each halted honest party, so every threshold is met for v, while no other value reaches the $n - t$ signed inputs it needs, only the at most t corrupted parties being able to sign it. Hence every honest party still running obtains grade 1 on v and halts.

Fact 2. The first honest decision fixes every honest value and certificate. Consider the earliest moment at which a party honest at that moment decides, with grade 1 or on the fast path, say on the value y; if two such parties decide at that moment, we show they decide the same value. On the fast path this is because a never-corrupted party signs one input only. With grade 1, no honest party has halted or fired the fast path in that phase, either of which would be an earlier decision, so every honest party outputs from that invocation of $\pi_{2\mathsf{ga}}$, and by graded consistency all with the same value. At that moment every honest party's local value is y. With grade 1, graded consistency has every honest party output y from that invocation, and step 7 writes it into b; on the fast path, the n signed inputs on y include those of all honest parties, implying that every honest party entered this phase with y. No honest party has halted before this moment, since halting requires a decision. So the previous fact applies with $v = y$ to the phase that follows, or to this phase on the fast path. That phase ends with every honest value still y and with every party that decides in it while honest deciding y, so the previous fact applies again to the next phase, and so on. Hence every honest local value is y from the moment of that first decision, and every terminal certificate issued by a party honest at the time carries y. Substitution is therefore faithful: a certificate counted in place of an absent party's signature carries the value every honest party holds if its issuer was honest when it issued it, and adds nothing otherwise, since the adversary then holds that party's key and could sign the replaced message directly. To be counted as a vote, a certificate moreover needs the input and attestation certificates for its value that a justified vote carries.

Fact 3. A halted party costs at most a leader. A corrupted party that falls silent does nothing it could not do anyway, and its certificates add nothing, by the previous fact. An honest party halts only with grade 1 or on the fast path, hence on a decision, say on y, so by the previous fact every honest local value is already y, every certificate of an honest party carries y, and every phase from then on meets the preconditions of the first fact, which settles it on the thresholds alone,

whatever the leader does. The election therefore matters only in phases in which no honest party has halted, where all n parties are present to be elected. What the halted party's silence removes is thus harmless, but Section 4 needs us to account for it. In π_{ccic} the substitution rule supplies its signed input but not its forwards and accusations, so a signed input that it alone holds is lost and accusations against a party may fall short of $t+1$; the coordinate then falls to the final "otherwise" rule of Figure 3 and the pair is rejected by $\mathsf{Ver}_{\mathsf{ccic}}$ rather than accepted with a coordinate it cannot support. What $\mathsf{Ver}_{\mathsf{ccic}}$ demands of a coordinate is untouched, a value needing its owner's signature and $\perp$ an equivocation pair or $t+1$ genuine accusations, so no pair that misreports a party still running becomes acceptable: missing accusations can make a pair unusable, not make a false one pass. In π_{2ga} the substitution rule supplies its signed input and its attestation for y. It also supplies its vote for y, which every running honest party completes with the input and attestation certificates for y that it holds. So the thresholds are met for y and for no other value, as in the first fact. In the election the halted party's VRF message is missing, so it cannot be elected.

Validity. If all honest parties start with input v, phase 1 meets the preconditions of the first fact, no party having halted yet, so every honest party outputs v in it.

Agreement. Let P_a be the first honest party to output, in some phase r, with output y. By the second fact every honest local value is y from then on and every certificate of an honest party carries y. By the first fact every other honest party still running outputs y too, at the end of phase r if P_a fired the fast path and at the end of phase $r+1$ otherwise; and nothing else is ever output, every honest decision being on y by the second fact. Hence all honest outputs are equal.

Termination and expected round complexity. Call a phase good if π_{ole} elects a leader honest at the start of the invocation and every honest party outputs its index. Consider a good phase at which some honest party has not yet output. If some party has decided while honest, before this phase or on its fast path, the phase meets the preconditions of the first fact by the second, and every remaining honest party outputs at its end. Otherwise no party has halted while honest, and every honest party runs the whole phase. The leader was honest throughout steps 1 to 3, so its pair satisfies $\mathsf{Ver}_{\mathsf{ccic}}$ by completeness and went to everyone in step 3, which closes before the election; it sent exactly one pair, so every honest party reads the same one, accepts it by transferability, and sets z to the same mode, which is not $\perp$ because validity puts every honest party's own input at its coordinate. All honest parties therefore enter step 5 with the same input, obtain grade 1 by graded validity, and output. So every honest party has output by the end of the first good phase. Fairness speaks of elections that every honest party runs, and a party that halts stops running them. This does not matter: a corrupted party may stay silent anyway, and once an honest party has halted, every honest party outputs by the end of that phase whatever the election returns, as in the proof of agreement. So we may apply fairness as if halted parties kept running the elections. Their identifiers $\mathsf{sid}^r_{\mathsf{ole}}$ are distinct, fixed before the setup and used in the order of the phases, so by fairness none of the first k phases is good with probability at most $(1-p)^k$. This tail also settles a subtlety: with negligible probability, for instance under a setup in which the corrupted parties win every election, an execution may never end, and an expectation over all executions would then be infinite. We therefore cut the execution after $K = \lceil \kappa/p \rceil$ phases, a party without output by then outputting its local value. The cut takes effect only if none of the first K phases is good, which happens with probability at most $(1-p)^K \leq e^{-\kappa}$, so it costs a negligible error, and the first good phase, counted as K if there is none among the first K, has expected index at most $\sum_{k<K}(1-p)^k \leq 1/p$. Each phase takes $2+1+R_{\mathsf{ole}}+3 = 6+R_{\mathsf{ole}}$ rounds, and a party halts in the round after its output, in which it sends its terminal certificate, which gives the bound. □

The fast path reads the signed inputs of the first round rather than the vector the block returns a round later, because that vector can only say less, a coordinate whose owner equivocated being

set to $\perp$ there.

Corollary 3.2. *Assuming existence of EUF-CMA signatures and of VRFs with exponential-size range, $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$ with π_{ole} the protocol of Figure 2 is a Byzantine agreement protocol in the signature-PKI and VRF-PKI model with expected round complexity $\frac{7n}{n-t}+1<15$.*

Proof. By Theorem A.1, π_{ole} has $R_{\mathsf{ole}}=1$ and fairness $p=(n-t)/n$. Theorem 3.1 then gives expected round complexity $(6+1)/p+1=\frac{7n}{n-t}+1$, which is below 15 because $t<n/2$ gives $n/(n-t)<2$. □

Proposition 3.3 (Fast termination)**.** *If no party is corrupted and every party has the same input, every party outputs at the end of the first round, and halts in the second.*

Proof. In step 1 every party signs its input and sends it to all parties, its own self included, so at the end of that step every party's E holds signed inputs of all n parties on the common value u. The fast path fires there. The round that follows carries the certificate out. □

Remark 3.4 (Unanimity alone)**.** *If every honest party has the same input and $R_{\mathsf{ole}}=1$, every honest party outputs within 7 rounds, that is, at the end of the first phase, and halts in the eighth. The fast path need not fire, since a corrupted party need not sign the common value.*

Remark 3.5 (No corruption alone)**.** *If no party is corrupted and π_{ole} is the protocol of Figure 2, every party outputs within 7 rounds and halts in the eighth, whatever the inputs. Every party signs one input and receives all n of them, so no coordinate is certified $\perp$ and every party holds the same fully populated pair; the election is agreed and elects an honest party, there being no other kind; so every party adopts the mode of one and the same vector and graded validity returns grade 1 at the end of the phase. The two hypotheses are thus independently sufficient for one phase, and together they bring the output forward to the first round by Proposition 3.3.*

3.2 From Agreement to Broadcast

Byzantine broadcast reduces to Byzantine agreement in a single round: the sender signs its value and multicasts it, every party adopts what it received as its input, and agreement does the rest. The round is spent before the first phase rather than inside it, so that every phase keeps its length; Section 4.2 says what folding it in would cost.

Fix a value $\perp_0\in\mathcal{M}$. Protocol $\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}}$ is the following.

1. The sender P_s multicasts (v,σ_s) with $\sigma_s:=\mathsf{Sign}_{\mathsf{sk}_s}(\mathsf{sid},\mathsf{send},v)$; the other parties send nothing.
2. Every party sets b to the value x of a pair (x,σ) received in the first round for which σ is a valid signature of P_s on $(\mathsf{sid},\mathsf{send},x)$, least under a fixed order on strings if more than one such pair arrived, and to $\perp_0$ if none did; it then runs $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$ from its first round with that local value.

Theorem 3.6. *Under the assumptions of Theorem 3.1, and for any PPT adversary corrupting at most $t<n/2$ parties, $\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}}$ is a Byzantine broadcast protocol in the signature-PKI model together with the setup of π_{ole}, with expected round complexity at most one more than that of $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$.*

Proof. Agreement and termination are those of Theorem 3.1, which constrain the inputs in no way, and the output lies in $\mathcal{M}$ because every local value does. For validity, let P_s be honest throughout: it signs v and nothing else, so every honest party sets $b:=v$, and validity of Theorem 3.1 gives every honest party the output v. The sender's round precedes the first phase, so every round bound of $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$ grows by exactly one. □

Corollary 3.7. *Assuming existence of EUF-CMA signatures and of VRFs with exponential-size range, $\pi_{\mathsf{rand\text{-}bb},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ with π_{ole} the protocol of Figure 2 is a Byzantine broadcast protocol in the signature-PKI and VRF-PKI model with expected round complexity $\frac{7n}{n-t}+2<16$.*

Proof. By Corollary 3.2 the expected round complexity of $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ is $\frac{7n}{n-t}+1$, and by Theorem 3.6 that of $\pi_{\mathsf{rand\text{-}bb},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ is one more, namely $\frac{7n}{n-t}+2$, which is below 16. □

Remark 3.8 (Honest sender)**.** *If P_s is honest throughout, every honest party outputs within* 8 *rounds and halts in the ninth, and the leader election matters only when the sender is corrupted. This is the analogue of Remark 3.4, with honesty of the sender in place of unanimity of the inputs, and one round more.*

Proposition 3.9 (Fast termination for BB)**.** *If no party is corrupted, every party outputs at the end of the second round, and halts in the third.*

Proof. The sender is honest, so every party receives (v,σ_s) in the first round, and it is the only signed value of P_s, so every party sets $b:=v$. Proposition 3.3 then applies to the execution of $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ that follows, whose first round is the second round here. □

4 Early-Stopping Byzantine Agreement and Broadcast in Expected Constant Rounds

4.1 Expected-Constant-Round Early-Stopping Byzantine Agreement

The protocol $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ of Section 3 terminates in an expected constant number of rounds but has no worst-case bound, since every phase fails to elect an honest leader with constant probability. Early-stopping protocols terminate in $O(\min\{f+2,t+1\})$ rounds, but take $\Theta(t)$ rounds when $f=t$, so they are never constant in expectation. As observed in [CFLZ26], the two guarantees do not follow from running a randomized prefix with a deterministic fallback, whatever the length of the prefix, even when the fallback is itself early-stopping.

The compiler of [CFLZ26], recalled in Section 2.4, resolves this by interleaving the two mechanisms rather than concatenating them, and we obtain such a protocol by applying its signature-PKI variant $\mathsf{Comp}^{\mathsf{PKI}}$ to $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$. The work of this section is to verify that $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ satisfies Properties (P1)–(P4), and to identify the constants, since these are what the bounds of Theorem 2.7 are expressed in.

The constants established below are collected in Table 3 for reference.

	Value	Meaning	Established in
c	7	rounds per phase	(P1)
p	$(n-t)/n$	fairness of π_{ole}	(P3), via Theorem A.1
Δ_0	1	rounds of $\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$ after committing	(P1)
Δ_1	0	phases after a unanimous one until commitment	(P2)
Δ_2	0	phases from the coin to commitment	(P3)
Δ_3	1	phases for a commitment to propagate	(P4)
L	1	injection period, admissible as $L>\Delta_2$	Theorem 4.1

Table 3: The constants of Properties (P1)–(P4) for $\pi_{\mathsf{rand}}:=\pi_{\mathsf{rand\text{-}ba},\,\mathsf{t}<\frac{\mathsf{n}}{2}}$, and the injection period we choose for them. The derived quantity is $\Delta_0^*=\max(\Delta_0,1)+\lceil\Delta_0/(Lc)\rceil=2$.

Theorem 4.1. *Assume EUF-CMA signature schemes and VRF schemes with exponential-size range exist. Then, for any PPT adversary corrupting at most $t < n/2$ parties, $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}ba},\, t<\frac{n}{2}})$ with injection period $L = 1$ is a Byzantine agreement protocol in the signature-PKI and VRF-PKI model whose expected round complexity is less than 18 and whose worst-case round complexity is at most $8f + 18$, where $f \leq t$ is the actual number of corruptions in the execution.*

Proof. We instantiate π_{rand} with $\pi_{\mathsf{rand\text{-}ba},\, t<\frac{n}{2}}$ and take one phase to be one iteration of its main loop. The seven steps of Figure 1 occupy the seven rounds one each: rounds 1 and 2 are steps 1 and 2, the two rounds of π_{ccic} written out; round 3 is the dissemination of step 3; round 4 is the single round of π_{ole} of step 4, whose adoption rule is local computation on the messages of rounds 3 and 4; and rounds 5 to 7 are steps 5 to 7, the three rounds of π_{2ga} written out, whose closing assignment is local computation on the messages of round 7. A party that commits in mid-phase, which only the fast path does, emits the ready of the current phase in the next round, as the compiler's halting rule prescribes.

Property (P1) holds with $c = R_{\mathsf{ccic}} + 1 + R_{\mathsf{ole}} + R_{\mathsf{2ga}} = 2 + 1 + 1 + 3 = 7$, the added round being the dissemination round, so that with $L = 1$, a signalling round following every phase, a compiled phase costs $c + \frac{1}{L} = 8$ rounds; $\pi_{\mathsf{rand\text{-}ba},\, t<\frac{n}{2}}$ runs no sub-protocol in the background, so the allowance (P1) makes for them is not needed. A party sets $committed_i$ on either of the two termination conditions of Figure 1, grade 1 or the fast path, and fixes its local output at that moment; each condition is monotone and depends only on the party's own view, hence is locally computable. Setting the flag suppresses no transmission the others need: grade 1 is decided by the messages of the last round of a phase, and a party that fires the fast path in mid-phase sends the terminal certificate that replaces it at every later threshold. We take $\Delta_0 = 1$, the one round being the one in which that certificate goes out, for the reason given at the end of this proof. The compiler also assumes that local values, like its default ϕ, range over the agreement domain; they do, since step 7 keeps the old b when the graded agreement returns $\perp$, so no party ever signs $\perp$ as its input.

Before the remaining properties we record what makes the certificates compatible with the injection. *Halts and injections never co-occur.* An honest party halts only on grade 1 or the fast path, and in both cases it has committed; being never corrupted, it emits the ready of its phase and is never caught equivocating. Every honest party therefore has a non-empty S at the boundary in question and none injects. The complementary case costs nothing: if the issuer of a certificate is corrupted at any point, the adversary holds its signing key and could sign any message of any block directly, so the substitution rule adds nothing to any threshold it could not already have met. A certificate minted by a corrupted party is thus inert, and a certificate issued by an honest one comes with the guarantee that no injection intervened.

For (P2), suppose all honest parties start a phase with the same local value v and none has committed. By Fact 1 in the proof of Theorem 3.1, every honest party still running obtains $(v, 1)$ from π_{2ga} in that phase and outputs v at its end, the halted parties being replaced at every threshold by their certificates, which carry v by Fact 2 there. A party that fires the fast path inside the phase does so on a value all n parties signed, which the honest parties' own inputs make v, so it too commits to v. Every honest party therefore commits to v by the end of that same phase, and keeps $b = v$ until it does. Hence $\Delta_1 = 0$.

For (P3), call a phase *good* if π_{ole} elects a leader P_ℓ that is honest at the start of the invocation and every honest party outputs ℓ. Consider a phase at which no honest party has committed, with arbitrary phase-start local values. Since the commit flag is monotone, no honest party has halted either, so every honest party runs the whole phase and no threshold needs a substitution. If a party fires the fast path inside the phase, all n parties signed its value, so the phase-start

values were unanimous and (P2) applies. Otherwise, in a good phase, the termination argument of Theorem 3.1 applies as it stands: every honest party reads the leader's pair before the election, adopts the same mode, which is not $\perp$, and obtains grade 1 by graded validity at the end of that same phase. Hence $\Delta_2 = 0$, and a window, which is now a single phase, fails only if no honest party has committed at its start and its phase is not good. An honest party that does not run the election of a window has halted, having committed before the window or on its fast path, so that window has not failed, and we may apply fairness as if halted parties kept running the elections. The elections of m windows fixed in advance have distinct identifiers, fixed before the setup and used in the order of the windows, so by Theorem A.1 the first k windows all fail with probability at most $(1-p)^k$, with $p = (n-t)/n > 1/2$, and summing over $k \le m$, the expected index of the first window that does not fail, counted as m if there is none, is at most $1/p$.

For (P4), let P_g be an honest party that commits in phase r with output y, at a phase entered with no honest party committed. If it committed on the fast path, all n signed inputs of phase r read y, so every honest party entered phase r with y and (P2) applies within phase r itself. If it committed on grade 1, then by graded consistency of π_{2ga} every honest party that produces an output in that invocation outputs y and step 7 sets $b := y$ at each of them. Every honest party therefore begins phase $r+1$ with local value y, those that committed holding it frozen and those that did not being free of injections, since P_g signalled ready at the boundary. Phase $r+1$ is thus a unanimity phase, and (P2) gives every remaining honest party commitment to y at its end. Hence $\Delta_3 = 1$. One cannot do better, since graded consistency equalizes values but not grades, so a party that does not reach grade 1 ends phase r with grade 0 and commits only in the phase after.

It remains to fix Δ_0, which is where the terminal certificates earn their keep. At $n = 2t+1$ the honest parties are exactly $n - t = t + 1$, so every threshold of a later phase needs every one of them: the $n - t$ signed inputs of an input certificate, the $t + 1$ attestations, the $t + 1$ votes. A committer that simply stopped would break each in turn, which is why a design without certificates must keep it running for a whole further phase. Here its certificate stands in its place at each of them, by the substitution rule and the faithfulness of the substitution established in the proof of Theorem 3.1, and what its silence removes from π_{ccic} and the election costs at most a leader, in a phase that is already unanimous, as that proof also establishes. Nothing is required of it after it commits beyond letting that one message go out.

That one message occupies the round after the one in which the output was fixed, grade 1 being decided by the messages of the last round of a phase and the fast path by those of the first; in each case the party has already sent for the round that fixed its output. Property (P1) meters Δ_0 from that moment and in rounds of π_{rand}, so $\Delta_0 = 1$.

The trailing term of Theorem 2.7 is then $\Delta_0^* = \max(\Delta_0, 1) + \lceil \Delta_0/(Lc) \rceil = 1 + \lceil \frac{1}{7} \rceil = 2$, and we take it as the theorem states it.

It remains to choose the injection period. Since $\Delta_2 = 0$, every integer $L \ge 1$ is admissible, and we take $L = 1$, so that an injection is attempted at every phase boundary, which costs nothing here because the window of (P3) is a single phase and has no interior boundary for an injection to disturb. This choice is optimal for the constants at hand. As $\Delta_3 = 1$ we have $\lceil \Delta_3/L \rceil = 1$ for every L, so the expected bound is $(cL+1)/p + \Delta_0^*$ and the coefficient of f in the worst-case bound is $cL + 1$; both grow with L, as does the remaining worst-case term $(c + \frac{1}{L})(L+1)$, while $\Delta_0^* = 2$ for every $L \ge 1$. Substituting $c = 7$, $p = (n-t)/n$, $\Delta_0 = 1$, $\Delta_1 = 0$, $\Delta_2 = 0$, $\Delta_3 = 1$ and $L = 1$, so that $c + \frac{1}{L} = 8$, $L(\lceil \Delta_3/L \rceil - 1) = 0$ and $L\lceil \Delta_3/L \rceil = 1$, the expected round complexity is at most $8 \cdot \frac{1}{p} + \Delta_0^* = \frac{8n}{n-t} + 2$, which is below 18 because $t < n/2$ gives $n/(n-t) < 2$, and the worst-case round complexity is at most $8\big(f + (\Delta_1 + 1) + \max(\Delta_1 + 1, \Delta_3)\big) + \Delta_0^* = 8(f+2) + 2 = 8f + 18$. As this bound holds only up to a negligible error, we also let every party halt at round $8t + 18$ in any

case; this changes no execution in which the bound holds and keeps the expectation finite. □

Over uncompiled $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$, whose expected round complexity is $\frac{7n}{n-t}+1<15$ and which offers no worst-case guarantee at all, the compiled protocol costs one signalling round out of every eight, plus the signalling round the general bound charges for the trailing certificate round.

Remark 4.2 (Fast termination survives)**.** *Proposition 3.3 survives the compilation: if no party is corrupted and every party has the same input, every party outputs at the end of the first round of* $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}})$*, since the compiler outputs as soon as* π_{rand} *fixes its local output, and halts at the end of the second, in which it sends its* ready *and its terminal certificate, the one round of* Δ_0*.*

4.2 Expected-Constant-Round Early-Stopping Byzantine Broadcast

Protocol $\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}}$ of Section 3.2 is one round of the sender sending their value, followed by $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$, so there is nothing to verify a second time, the compiler being applied to $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$ exactly as in Section 4.1 with the sender's round before it; we write $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}})$ for the result. This is also why the sender's round is kept outside the first phase rather than folded into it. Property (P1) asks every phase to have the same length c, so a first phase of $c+1$ rounds would force $c=8$ throughout, and c multiplies through both bounds of Theorem 2.7, so the expected bound would become $9n/(n-t)+2<20$ and the worst case $9(f+2)+2=9f+20$, against the 19 and $8f+19$ below.

Theorem 4.3. *Assume EUF-CMA signature schemes and VRF schemes with exponential-size range exist. Then, for any PPT adversary corrupting at most* $t<n/2$ *parties,* $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}})$ *with injection period* $L=1$ *is a Byzantine broadcast protocol in the signature-PKI and VRF-PKI model whose expected round complexity is less than* 19 *and whose worst-case round complexity is at most* $8f+19$*, where* $f\le t$ *is the actual number of corruptions.*

Proof. What the parties run after the sender's round is $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}})$ on the local values that round leaves them with, so Theorem 4.1 applies to it verbatim, with the constants of Table 3. Agreement and termination are inherited from it directly. For validity, let P_s be honest throughout; it signs v and nothing else, so every honest party enters the compiled protocol with local value v, and the validity clause of Theorem 2.7, the compiler initializing each local value to the value it is given, returns v at every honest party. Both bounds of Theorem 4.1 therefore grow by the single round of the sender, giving $1+\frac{8n}{n-t}+2<19$ and $1+(8f+18)=8f+19$. □

Remark 4.4 (Fast termination survives)**.** *Proposition 3.9 survives the compilation, for the same reason. If no party is corrupted, every party outputs at the end of the second round of* $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}})$*, the sender's round coming first and the fast path firing in the round after it.*

5 Parallel Byzantine Agreement and Interactive Consistency

In *m-parallel Byzantine agreement* each party holds a vector of m inputs and the parties run m agreements at once, every honest party outputting a vector of m values, agreeing coordinatewise, with coordinate k satisfying the validity of Definition 2.1 for the k-th inputs. In *interactive consistency* every party is the sender of one broadcast, every honest party outputting the same n-vector, whose j-th coordinate is P_j's input whenever P_j is honest. Interactive consistency is thus n-parallel Byzantine broadcast.

Running m randomized agreements side by side ordinarily costs rounds. Each instance terminates after a number of phases that is geometric in its own randomness, and a party is done only when all m are, so with independent randomness the expected phase count is the expectation of a maximum of m geometric variables, which is $\Theta(\log m)$ times a single instance's. Avoiding that blow-up is what the parallel-composition compiler of Ben-Or and El-Yaniv [BOEY03] and its later variants are for, and what the parallel treatment in [CFLZ26] achieves for the protocols compiled there.

Our approach is itself a compiler, but a much lighter one, applicable to leader-based protocols such as ours. Instead of running each instance many times over, as those compilers do, it lets all instances share one leader election. The randomness of $\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}}$ and $\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}}$ enters through the leader election alone, and the event their analyses rely on, that π_{ole} elects a party honest at the start of the invocation and every honest party outputs its index, mentions no local value, no input and no particular instance. So let the m instances invoke π_{ole} under one and the same session identifier, formed from that of the parallel protocol and the phase index but not from the instance index, while keeping their π_{ccic} and π_{2ga} identifiers distinct. The parties then compute and send one VRF value each per phase, which every instance reads, the m invocations are a single invocation in the sense of Definition 2.3, and its good event is shared by every instance: whatever else has happened, every instance still running commits in the first good phase, so the last instance commits when a single one would have, and the maximum costs nothing.

Theorem 5.1 (Parallel Byzantine agreement)**.** *Assume EUF-CMA signature schemes and VRF schemes with exponential-size range exist, and let m be polynomial in κ. Then, for any PPT adversary corrupting at most $t < n/2$ parties, running m instances of $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}ba},\mathsf{t}<\frac{\mathsf{n}}{2}})$ that share the session identifiers of their leader elections is an m-parallel Byzantine agreement protocol in the signature-PKI and VRF-PKI model, with the round complexities of Theorem 4.1, expected less than 18 and at most $8f + 18$ in the worst case.*

Proof. Sharing the election identifier leaves each instance a protocol to which Theorem 4.1 applies, since a party's behaviour in π_{ole} reads no local value and the shared identifier only makes the elections of different instances coincide; so each instance has agreement, validity and both bounds on its own inputs, and coordinatewise agreement and validity follow. The worst-case bound is deterministic and holds in every execution of every instance, so the maximum over the m instances obeys it, whatever phases the corrupted parties' false signals fall in, each instance keeping its own compiler identifier and so its own signals. For the expected bound, the good event of a phase is common to all instances, so every instance still running commits in the first good phase and the expected phase count is the single-instance one. The blocks' guarantees hold per invocation up to a negligible error; m is polynomial in κ and each instance runs $O(f)$ phases, so a union bound over the polynomially many invocations leaves the error negligible. □

Theorem 5.2 (Interactive consistency)**.** *Under the assumptions of Theorem 5.1, running n instances of $\mathsf{Comp}^{\mathsf{PKI}}(\pi_{\mathsf{rand\text{-}bb},\mathsf{t}<\frac{\mathsf{n}}{2}})$, the j-th with sender P_j, that share the session identifiers of their leader elections is an interactive consistency protocol in the signature-PKI and VRF-PKI model, with the round complexities of Theorem 4.3, expected less than 19 and at most $8f + 19$ in the worst case.*

Proof. As for Theorem 5.1, with Theorem 4.3 in place of Theorem 4.1. Agreement on each coordinate is that instance's agreement, and coordinate j carries P_j's input whenever P_j is honest by that instance's Byzantine broadcast validity. □

Both round complexities are those of a single agreement, and only the messages multiply, a phase sending m or n times the messages of one instance, save for the one leader election they share.

Fast termination survives the parallel run too, and at the same round. The trigger of the fast path of Figure 1 reads the signed inputs of one instance's own π_{ccic} and nothing else, so it fires in each instance independently of the others. Hence, if no party is corrupted and every party holds the same input in each coordinate, every party outputs its vector at the end of the first round, by Remark 4.2 applied to each instance at once; and if no party is corrupted, the interactive consistency protocol outputs its whole vector at the end of the second round, by Remark 4.4, every sender being honest. In both cases a party halts one round after it outputs, at the end of round 2 for parallel BA and of round 3 for IC.

Remark 5.3. *We note that this method of obtaining 'IC for free' applies not just specifically to our protocol, but also other expected constant round, phase-based protocols, where the successful election of a common honest leader leads to termination in constant number of rounds. For example, the protocol of [KK09] explicitly uses this idea (citing [FG03]). Similar to Theorem 5.1, parallelizing their protocol and using the same election for each coordinate, keeps the protocol expected constant rounds.*

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 Peter Robinson and Faith Ellen, editors, *38th ACM PODC*, pages 317–326, Toronto, ON, Canada, July 29 – August 2, 2019. ACM.

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

[CDG+24] Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. DARE to agree: Byzantine agreement with optimal resilience and adaptive communication. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *43rd ACM PODC*, pages 145–156, Nantes, France, June 17–21, 2024. ACM.

[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, Paper 2026/1146, 2026.

[CFLZ26] Michele Ciampi, Pouyan Forghani, Yun Lu, and Vassilis Zikas. Randomized early-stopping Byzantine agreement with applications to round-efficient MPC. Cryptology ePrint Archive, Paper 2026/2284, 2026.

[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.

[DKLZ25] Giovanni Deligios, Ivana Klasovita, and Chen-Da Liu-Zhang. Asymptotically optimal early termination for dishonest majority broadcast. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part V*, volume 15605 of *LNCS*, pages 116–144, Madrid, Spain, May 4–8, 2025. 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.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[EKL25] Fatima Elsheimy, Simon Holmgaard Kamp, and Julian Loss. Early stopping is cheap. Cryptology ePrint Archive, Report 2025/1107, 2025.

[ELP24] Fatima Elsheimy, Julian Loss, and Charalampos Papamanthou. Early stopping byzantine agreement in $(1 + \epsilon) \cdot f$ rounds. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 398–424, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[FG03] Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 211–220, Boston, MA, USA, July 13–16, 2003. ACM.

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017.

[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.

[KK06a] 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.

[KK06b] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. Cryptology ePrint Archive, Report 2006/065, 2006.

[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.

[LN24] Julian Loss and Jesper Buus Nielsen. Early stopping for any number of corruptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 457–488, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press.

[WXSD25] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round Byzantine broadcast under dishonest majority. *Journal of the ACM*, 72(4):30:1–30:39, 2025.

[Zam96] Arkady Zamsky. A randomized byzantine agreement protocol with constant expected time and guaranteed termination in optimal (deterministic) time. In James E. Burns and Yoram Moses, editors, *15th ACM PODC*, pages 201–208, Philadelphia, PA, USA, August 23–26, 1996. ACM.

A Constructions of the Building Blocks

This appendix gives constructions for the three primitives of Section 2.2, together with their analyses. The leader election is in the VRF-PKI model and the other two in the signature-PKI model; all three tolerate $t < n/2$ corruptions, and their round complexities are the $R_{\mathsf{ole}} = 1$, $R_{\mathsf{ccic}} = 2$ and $R_{\mathsf{2ga}} = 3$ used in Section 3.

Protocol π_{ole}

Below is the code for party P_i in the VRF-PKI model, for an invocation with session identifier sid; ties between VRF values are broken by party index.
Initialization: on input $(\mathtt{input})$, initialize $E := \emptyset$.
Main execution

1. Compute $(\rho_i, \omega_i) := \mathsf{VRF}_{\mathsf{sk}_i}(\mathsf{sid})$ and send (i, ρ_i, ω_i) to all parties.
2. Upon receiving valid VRF messages (j, ρ_j, ω_j) such that $\mathsf{Ver}_{\mathsf{pk}_j}(\mathsf{sid}, \rho_j, \omega_j) = 1$, add (j, ρ_j, ω_j) to E. Let ℓ be the party index j minimizing (ρ_j, j) lexicographically among all $(j, \rho_j, \omega_j) \in E$. Output $(\mathtt{output}, \ell)$. This step is local computation on the messages of round 1 and costs no round.

Figure 2: A VRF-based oblivious leader election protocol.

Theorem A.1. *Assume VRF schemes with exponential size range exist. Then, for any PPT adversary corrupting at most $t < n/2$ parties, protocol π_{ole} is an oblivious leader election protocol with fairness $p = (n - t)/n$ in the VRF-PKI model. Moreover, π_{ole} terminates in one round.*

Proof. Termination follows directly from the protocol: each party evaluates its VRF and sends its value and proof in one round.

For fairness, fix a polynomial number m of invocations as in Definition 2.3. Let $\rho_{i,j}$ be P_i's VRF value at sid_j, whether or not it sends that value, and let C_j contain the parties corrupted before invocation j starts. Call invocation j *bad* if the smallest pair $(\rho_{i,j}, i)$ belongs to a party in C_j.

If an invocation is not bad, the party with the smallest pair is honest at its start and sends its value and proof to everyone. By provability and uniqueness, all honest parties accept that value, and no valid message can contain a smaller pair. Thus the invocation succeeds, except with negligible probability. It therefore suffices to prove that, for every $k \le m$,

$$\Pr[\text{the first } k \text{ invocations are all bad}] \le \left(\frac{t}{n}\right)^k + \mathsf{negl}(\kappa). \tag{1}$$

Fix $k \ge 1$. We count two kinds of *disclosure events*, *disclosures* for short: a corruption reveals the target's VRF values evaluated at the identifiers of all invocations that have not started yet, and the start of an invocation reveals the currently uncorrupted parties' values at the invocation's identifier. Up to and including the start of invocation k, there are at most $k + t$ disclosures. Pad with idle steps afterward to obtain $M = k + t$ steps.

For $s \in \{0, \ldots, M\}$, define G_s as follows. Run the real execution through the first s disclosures and stop it. Then, only to decide the output of G_s, *complete* the first k invocations. That is, fix what the execution has not yet fixed, using the following completion rules:

- Invocations that have already started keep their bad or good status, determined by the parties actually corrupted at their start.
- For unstarted invocations, retain the VRF values of the currently corrupted parties. If there are c such parties, choose another $t - c$ parties to count as corrupted, say the currently uncorrupted ones of smallest index; any choice that does not look at the values sampled next would do.

Then assign independent uniform values to all currently uncorrupted parties at the identifiers of all invocations that have not started yet.

G_s outputs 1 if all first k invocations are bad under this completion; we call this check the *completion test*, and say that it *accepts* when G_s outputs 1. The additional corruptions affect only unstarted invocations; they never change the status of an invocation that has already started.

In G_0, every value used by the test is uniform and exactly t parties are counted as corrupted. Hence

$$\Pr[G_0 = 1] \leq (t/n)^k + \mathsf{negl}(\kappa).$$

In G_M, all first k invocations have started, so the output is precisely the indicator that they are all bad. Throughout the argument, ties involving uniform values contribute only negligible error because the VRF range has exponential size.

We show that, for every $s \geq 1$,

$$\Pr[G_s = 1] \leq \Pr[G_{s-1} = 1] + \mathsf{negl}(\kappa). \tag{2}$$

Let $\widetilde{G}_s$ be G_s, except that the values revealed by disclosure s are replaced by independent uniform values in the completion test. All earlier disclosures remain real, and the game ends without continuing the execution using the replaced values.

First compare $\widetilde{G}_s$ with G_{s-1}. We consider the two kinds the s-th disclosure may be.

Suppose the disclosure is a corruption, with c parties previously corrupted. The target is chosen before its replacement values are sampled. In G_{s-1}, the completion assigns uniform values to $t - c$ additional parties counted as corrupted and $n - t$ parties counted as honest. In $\widetilde{G}_s$, the target supplies one independent collection of uniform values, and the completion supplies the other $t - c - 1$ corrupted parties and the $n - t$ honest parties. Thus the values counted as corrupted and honest have the same distribution in both tests. The previously corrupted parties' values and the status of past invocations are unchanged. Consequently, the acceptance probabilities agree, up to negligible error.

Suppose instead that the disclosure starts an invocation. Its values have the same distribution in both tests, as does the completion of later invocations. In G_{s-1}, the test counts some additional parties as corrupted in this invocation. In $\widetilde{G}_s$, it records the invocation's status using only the parties actually corrupted at its start. Counting fewer parties as corrupted cannot increase the probability that the smallest value belongs to a corrupted party. Past invocations retain their status, so the probability that the test accepts cannot increase. Therefore,

$$\Pr[\widetilde{G}_s = 1] \leq \Pr[G_{s-1} = 1] + \mathsf{negl}(\kappa). \tag{3}$$

It remains to compare G_s and $\widetilde{G}_s$. Order all party–identifier pairs in advance and replace the values revealed by disclosure s one at a time, skipping pairs not revealed there. We show that each replacement changes the acceptance probability by at most a negligible amount.

Consider neighbouring hybrids, which differ only in the value of the party–identifier pair (P_a, sid_j). The reduction registers the VRF challenger's public key as P_a's key, generates every other key honestly, and simulates the execution. It obtains P_a's values and proofs at other identifiers from its evaluation oracle.

If P_a is corrupted or invocation j starts before disclosure s, the reduction immediately returns 0. It also returns 0 if disclosure s does not reveal the selected pair. On these executions, the neighbouring hybrids do not replace the selected value and therefore have the same output distribution. Returning 0 on both sides preserves their difference in acceptance probabilities.

Otherwise, P_a is still uncorrupted and invocation j has not started. The reduction has not queried its oracle at sid_j, since the identifiers are distinct and the setup is used only for these

invocations. It requests the challenge at sid_j, uses the returned value for the selected pair, generates the remaining values as prescribed by the hybrid, and outputs the completion test's bit.

The test needs only numerical values. At a corruption, the reduction can therefore finish before returning P_a's secret key; at an invocation, it can finish before providing the proof for the challenge value. The two challenge distributions give exactly the two neighbouring hybrids on the remaining executions. VRF pseudorandomness thus bounds their difference by a negligible amount. Summing over the polynomially many replacements gives

$$\left|\Pr[G_s = 1] - \Pr[\widetilde{G}_s = 1]\right| \leq \mathsf{negl}(\kappa).$$

Together with (3), this proves (2). Applying (2) over all M steps establishes (1).

Finally, a non-bad invocation succeeds except with negligible probability, so by (1) and a union bound over the first k invocations, none of them succeeds with probability at most $(t/n)^k + \mathsf{negl}(\kappa)$. This proves p-Fairness with $p = (n-t)/n$.

□

The certification mechanism of π_{ccic} — a coordinate backed either by its owner's signature on a value or by public evidence that its owner misbehaved — is the idea of the Learn-or-Expose line of [CDG+24, CEG26], where it is a leader-driven subroutine for one accused party at a time, certifies exposure by a $(t+1)$-threshold signature, and delivers its vector to the leader alone. The construction below is leaderless and one-shot, places no precondition on the inputs, and needs only plain EUF-CMA signatures and two rounds. That the accusation certificate is this cheap is a consequence of the honest majority: a set of $t+1$ accusations can never be assembled against a party that reaches everyone in the first round, and can always be assembled against one that stays silent, which all $n - t \geq t + 1$ honest parties accuse. No threshold has both properties once the majority may be corrupted, and the dishonest-majority construction of [CEG26] instead certifies exposure by separation in a graph built from the accusations, after [LN24, DKLZ25] and the trust graphs of [WXSD25].

Protocol π_{ccic}

Below is the code for party P_i in the signature-PKI model, for an invocation with session identifier sid.
Initialization: on input $(\mathtt{input}, x)$, where $x \in \{0,1\}^*$, initialize $E := \emptyset$ and $F := \emptyset$. Every signed input P_i receives is added to E, and every accusation it receives is added to F.

Main execution

1. Compute $\sigma_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, \mathsf{input}, x)$, add (i, x, σ_i) to E, and send it to all parties.
2. For every party P_j:
   - if E contains a signed input of P_j, send one to all parties, and if it also contains one on a different value, send that one too;
   - otherwise, compute $\sigma_{i,j} := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, j, \mathsf{accuse})$, add it to F, and send $((\mathsf{sid}, j, \mathsf{accuse}), \sigma_{i,j})$ to all parties.

   Then, once the messages of this round have been processed, for every party P_j:
   - if E contains signed inputs s, s' of P_j on different values, set $\boldsymbol{v}[j] := \perp$ and $\omega[j] := (s, s')$;
   - otherwise, if F contains accusations against P_j from at least $t+1$ distinct parties, set $\boldsymbol{v}[j] := \perp$ and set $\omega[j]$ to one accusation from each of $t+1$ of those parties;
   - otherwise, if E contains a signed input (j, x_j, σ_j) of P_j, set $\boldsymbol{v}[j] := x_j$ and $\omega[j] := (j, x_j, \sigma_j)$;
   - otherwise, set $\boldsymbol{v}[j] := \perp$ and $\omega[j] := \perp$.

   Finally, output $(\mathtt{output}, \boldsymbol{v}, \omega)$.

Figure 3: A certified crusader interactive consistency protocol for $t < n/2$ in two rounds.

Consistency rests on the priority of $\perp$: a party that holds evidence against P_j records $\perp$ even when it also holds a value from P_j. That priority is what buys consistency without a third round.

The setup algorithm $\mathsf{Setup}_{\mathsf{ccic}}$ runs $\mathsf{Gen}(1^\kappa)$ of the signature scheme for every party, so that $\mathsf{pp} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ consists of the verification keys and sk_i is the signing key of P_i. For verification, use the following predicate. The predicate $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}, \omega) = 1$ holds if, for every party P_j, one of the following is true:

- $\boldsymbol{v}[j] \in \{0,1\}^*$ and $\omega[j]$ contains a valid signature by P_j on $(\mathsf{sid}, \mathsf{input}, \boldsymbol{v}[j])$; or
- $\boldsymbol{v}[j] = \perp$ and $\omega[j]$ contains two valid signatures by P_j, on $(\mathsf{sid}, \mathsf{input}, x)$ and $(\mathsf{sid}, \mathsf{input}, x')$ for some $x \neq x'$; or
- $\boldsymbol{v}[j] = \perp$ and $\omega[j]$ contains valid signatures from at least $t+1$ distinct parties on $(\mathsf{sid}, j, \mathsf{accuse})$.

Otherwise, $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}, \omega) = 0$. The predicate does not depend on i, so transferability will be immediate; we keep the argument in the signature because a verifier-dependent certificate is what the dishonest-majority constructions cited above require.

Theorem A.2. *Assume EUF-CMA signature schemes exist. Then, for any PPT adversary corrupting at most $t < n/2$ parties, $(\mathsf{Setup}_{\mathsf{ccic}}, \pi_{\mathsf{ccic}}, \mathsf{Ver}_{\mathsf{ccic}})$ is a certified crusader interactive consistency protocol, also in the signature-PKI model, where the adversary chooses the keys of the parties it corrupts. Moreover, π_{ccic} terminates in two rounds.*

Proof. Write G for the set of parties honest at the start of the second round; since at most t parties are ever corrupted and $t < n/2$, we have $|G| \geq n - t \geq t + 1$. By the convention of Section 2.1 a party in G carries out its second round as prescribed even if it is corrupted during it, and by synchrony everything it sends reaches every party before any output is computed. We assume throughout that the adversary forges no signature of an honest party, and, since sid is inside every signed string, no signature from another invocation counts here. The proof uses nothing else about the keys and treats the signatures of corrupted parties as arbitrary, so it holds as well when the adversary chooses the keys of the parties it corrupts.

Termination. Two communication rounds are followed by local computation, so every honest party terminates in two rounds.

Completeness. Fix a party P_i honest at the end of the invocation, which therefore computed its output honestly, and a coordinate j. If the first or the second rule applies, then $\omega_i[j]$ is an equivocation pair or a set of $t+1$ accusations from distinct parties, either of which $\mathsf{Ver}_{\mathsf{ccic}}$ accepts. Otherwise we claim that the third rule applies, so that the fourth never does. Some party in G must hold a signed input of P_j at the end of the first round: if none did, every party in G would accuse P_j in the second, and P_i would hold accusations from $|G| \geq t+1$ distinct parties, so the second rule would have applied. That party forwards a signed input of P_j in the second round, so E contains one at P_i; and since the first rule did not apply, all signed inputs of P_j in E carry the same value. The third rule therefore sets $\boldsymbol{v}_i[j]$ to that value with a signed input on it as proof. Thus every coordinate of $(\boldsymbol{v}_i, \omega_i)$ satisfies the verification predicate, and $\mathsf{Ver}_{\mathsf{ccic}}(\mathsf{pp}, \mathsf{sid}, i, \boldsymbol{v}_i, \omega_i) = 1$.

Consistency. Fix a coordinate j and honest parties $P_i, P_{i'}$ with $\boldsymbol{v}_i[j] = x \neq \perp$ and $\boldsymbol{v}_{i'}[j] = x' \neq \perp$. Suppose first that no party in G holds a signed input of P_j at the end of the first round. Then every party in G accuses P_j in the second, so every honest party holds accusations from $|G| \geq t+1$ distinct parties, the first or the second rule applies at P_i, and $\boldsymbol{v}_i[j] = \perp$, a contradiction. So some $P_h \in G$ holds one. If it holds signed inputs of P_j on two different values, it forwards two of them on different values, every honest party holds these two, and the first rule makes every honest coordinate $\perp$, contradicting $x \neq \perp$; so all signed inputs of P_j that P_h holds carry one and the same value x_h, which it forwards to every party. If $x \neq x_h$, then P_i holds signed inputs of P_j on two

different values, the first rule applies, and $\boldsymbol{v}_i[j] = \perp$, again a contradiction. Hence $x = x_h$, and the same argument gives $x' = x_h$, so $x = x'$.

Validity. Fix a party P_j that is never corrupted, with input x_j. It signs exactly one message of the form $(\mathsf{sid}, \mathsf{input}, \cdot)$, namely $(\mathsf{sid}, \mathsf{input}, x_j)$, so by unforgeability no equivocation pair against P_j exists and the second clause of $\mathsf{Ver}_{\mathsf{ccic}}$ is unavailable in coordinate j. Since every honest party receives P_j's signed input in the first round, no honest party accuses it, so accusations against P_j come from at most t distinct parties, fewer than the $t+1$ the third clause requires. Therefore every pair $(\boldsymbol{v}, \omega)$ that $\mathsf{Ver}_{\mathsf{ccic}}$ accepts carries a value at coordinate j, and that value is x_j.

Transferability. The predicate $\mathsf{Ver}_{\mathsf{ccic}}$ is public, depends only on publicly verifiable signatures, and does not depend on the verifier's identity. Therefore, if an honest party accepts a pair $(\boldsymbol{v}, \omega)$, then every honest party that receives the same pair performs the same verification checks and also accepts it.

□

Protocol $\pi_{2\mathsf{ga}}$

Below is the code for party P_i in the signature-PKI model, for an invocation with session identifier sid.
Initialization: on input $(\mathtt{input}, x)$, where $x \in \{0,1\}^*$, initialize $M := \emptyset$.

Main execution

1. Compute $\sigma_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, \mathsf{input}, x)$ and send (x, σ_i) to all parties.
2. For every party P_j, add (j, v, σ) to M if (v, σ) is the pair received from P_j in the previous round for which σ is a signed input of P_j on v, least under a fixed order on strings if more than one arrived. If M contains entries for one and the same value $v \in \{0,1\}^*$ from at least $n-t$ distinct parties, let C be a set of $n-t$ of their signed inputs, compute $\alpha_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, \mathsf{attest}, v)$, and send (v, C, α_i) to all parties.
3. Let V be the set of values for which an input certificate was received in the previous round. If $V = \{y\}$ for a single value y, and attestations for y were received in the previous round from at least $t+1$ distinct parties, assemble an input certificate C_y and an attestation certificate A_y for y from those messages, compute $\beta_i := \mathsf{Sign}_{\mathsf{sk}_i}(\mathsf{sid}, \mathsf{vote}, y)$, and send the justified vote (y, C_y, A_y, β_i) to all parties. Then, if justified votes for one and the same value $y \in \{0,1\}^*$ are received in this round from at least $t+1$ distinct parties, output $(\mathtt{output}, y, 1)$; otherwise, if a justified vote for some value was received in this round, output $(\mathtt{output}, y, 0)$ for the value y of the one least under a fixed order on strings; if not, output $(\mathtt{output}, \perp, 0)$.

Figure 4: A 2-graded agreement protocol for $t < n/2$ in three rounds.

A signed input, an attestation and a vote of P_j for v are signatures of P_j on $(\mathsf{sid}, \mathsf{input}, v)$, $(\mathsf{sid}, \mathsf{attest}, v)$ and $(\mathsf{sid}, \mathsf{vote}, v)$. An input certificate for v is a set of signed inputs on v from $n-t$ distinct parties, an attestation certificate for v a set of attestations for v from $t+1$ distinct parties, and a justified vote for v a vote for v together with one of each, which is what a party checks before counting it.

The attestation is what removes a round. A vote here is *self-justifying*: it travels with an input certificate and an attestation certificate, so a single vote from one party suffices to carry every other party to the value at grade 0. A protocol that votes on a bare signature must instead spend a fourth round disseminating a certificate of $t+1$ votes, since a bare vote from a corrupted party is backed by nothing and a party that adopted its value could be left alone with it. The two thresholds are as far apart as $t < n/2$ allows: $n-t$ for an input certificate, so that under unanimity every party can form one while no second value can be certified, and $t+1$ for an attestation certificate, so that

it cannot be assembled by corrupted parties alone.

Theorem A.3. *Assume EUF-CMA signature schemes exist. Then, for any PPT adversary corrupting at most $t < n/2$ parties, protocol π_{2ga} is a 2-graded agreement protocol in the signature-PKI model. Moreover, π_{2ga} terminates in three rounds.*

Proof. For $k \in \{1, 2, 3\}$ write H_k for the set of parties honest at the start of round k; since at most t parties are ever corrupted, $|H_k| \geq n - t \geq t + 1$, and $H_3 \subseteq H_2 \subseteq H_1$ because corruption is irreversible. By the convention of Section 2.1 a party in H_k carries out round k as prescribed even if it is corrupted during that round, and by synchrony everything it sends in round k reaches every party before round $k + 1$ begins. We assume throughout that the adversary forges no signature of an honest party, and, since sid is inside every signed string and the three tags are disjoint, no signature produced in another invocation or under another tag counts here.

Witnessing. If an attestation certificate for a value u exists at any point in the execution, then $u \in V$ at every party. Indeed such a certificate carries attestations from $t + 1$ distinct parties, and at most t parties are ever corrupted, so at least one attester is never corrupted. A party never corrupted attests u only in round 2 and only together with an input certificate for u, which it sends to all parties. Every party therefore receives an input certificate for u in round 2.

Vote uniqueness. If a party P_h never corrupted sends a justified vote for y, then no justified vote for any $u \neq y$ exists at any point. A justified vote for u contains an attestation certificate for u, so by witnessing $u \in V$ at P_h; but P_h voted, so $V = \{y\}$ at P_h, whence $u = y$.

Termination. The protocol consists of three communication rounds followed by local computation, so every honest party terminates in three rounds.

Graded validity. Assume all honest parties have input v. Every party never corrupted sends a signed input on v in round 1 and signs no other value, so at every party M contains an entry on v for each of them, hence from at least $n - f \geq n - t$ distinct parties; a party of H_1 corrupted during round 1 cannot be counted here, the adversary being free to have it sign a second value in that same round. No input certificate for any $u \neq v$ can exist, since it would require signed inputs on u from $n - t$ distinct parties while only the at most $t < n - t$ parties corrupted at some point can produce one. Hence every party in H_2 forms an input certificate for v in round 2 and sends it together with an attestation for v; as $|H_2| \geq t + 1$, every party then holds an input certificate and $t + 1$ attestations for v, and $V = \{v\}$. So every party in H_3 sends a justified vote for v in round 3, and since $|H_3| \geq t + 1$ every honest party receives justified votes for v from at least $t + 1$ distinct parties and outputs $(v, 1)$.

Graded consistency. Suppose some party honest at the end of the protocol outputs $(y, 1)$. Having computed its output as prescribed, it received justified votes for y from $t+1$ distinct parties, and at most t parties are ever corrupted, so one of them is the vote of a party P_h never corrupted, which by unforgeability P_h produced itself. Now P_h signs a vote only in round 3 and only while sending a justified vote to all parties, so every party that reaches the local computation receives a justified vote there and the $(\perp, 0)$ branch is never taken; by vote uniqueness no justified vote for any $u \neq y$ exists, so every justified vote received is for y. Hence every honest party that produces an output outputs $(y, \cdot)$.

□