# Byzantine Broadcast with Unknown Participants

Wonseok Choi* Ran Cohen† Juan Garay‡ Nikos Skoumios§ Vassilis Zikas¶

**Abstract**

A sender wishes to broadcast a message across an open, permissionless network (e.g., on the dark web) such that whoever is around and active reaches agreement, even if the sender is malicious. Crucially, no a priori knowledge of the number of parties or their identities is given, and no assumptions are made regarding the number of honest parties or existence of resource-allocation mechanisms, like proof-of-work or stake.

The above is an instance of *Byzantine broadcast (BB)* with *unknown-participants* (UP Broadcast). Despite extensive research on dishonest-majority BB, existing approaches (such as Dolev-Strong broadcast) fail in this setting, as they rely on knowing the number of parties or corruption bounds. A key challenge, which might appear as an inherent limitation, is that malicious parties can join dynamically, making it difficult for honest parties to terminate without violating consistency.

In this work, we provide the first definitions of UP Broadcast, accommodating both static and dynamic participation under arbitrarily many corruptions. We show that the problem is solvable by introducing a novel primitive called *active-parties set agreement (APSA)*, which we use to construct a polynomial-time *deterministic* UP Broadcast protocol (as well as *UP Interactive Consistency*). Our protocols guarantee consistency and validity for all honest parties active throughout the execution.

Furthermore, we settle the round complexity of UP Broadcast by proving that our deterministic protocols are round-optimal, matching a lower bound that holds even for *randomized* protocols. Finally, while we establish negative results for unrestricted *dynamic participation* where parties may join late, we formulate best-possible guarantees for late-joining parties alongside matching round-optimal constructions.

*DGIST. E-mail: wonseok@dgist.ac.kr.

†Reichman University. E-mail: cohenran@runi.ac.il.

‡Department of Computer Science and Engineering, Texas A&M University. E-mail: garay@tamu.edu.

§University of Washington. E-mail: nskoum@cs.washington.edu.

¶School of Cybersecurity and Privacy, Georgia Institute of Technology. E-mail: vzikas@gatech.edu.

# Contents


**1 Introduction** **1**
- 1.1 Technical Overview . . . 2
- 1.2 Related Work . . . 6

**2 Modeling Broadcast with Unknown Participants** **9**
- 2.1 The Unknown-Participants Model . . . 9
- 2.2 Byzantine Broadcast in the UP Model . . . 12

**3 Active-Parties Set Agreement Protocol** **13**
- 3.1 Adapting Dolev-Strong to the UP Model . . . 14
- 3.2 The APSA Protocol . . . 15
- 3.3 APSA Protocol Analysis . . . 16

**4 Interactive Consistency with Unknown Participants** **18**
- 4.1 Building on the APSA Protocol . . . 19
- 4.2 The UPIC Protocol . . . 19
- 4.3 Interactive Consistency Protocol Analysis . . . 20

**5 Byzantine Broadcast with Unknown Participants** **22**

**6 Round-Complexity Lower Bounds** **23**
- 6.1 A Lower Bound for UP Broadcast . . . 23
- 6.2 Lower Bounds for APSA and UPIC . . . 26

**7 Late-Joining Parties** **27**
- 7.1 Impossibility of UP Broadcast with Late-Joining Receivers . . . 27
- 7.2 UP Broadcast with Selective Abort for Late-Joining Receivers . . . 29
- 7.3 Security with a Late-Joining Sender . . . 31

**References** **35**

# 1 Introduction

Byzantine agreement (BA) [PSL80, LSP82] is a fundamental primitive in fault-tolerant distributed computing. It enables a set of parties to reach agreement on their inputs in an adversarial setting where malicious parties attempt to disrupt consistency or validity. BA comes in two flavors, *Consensus* and *Broadcast*.[1] In Consensus, every party has an input and the goal is to agree on the output (*consistency*), such that if all honest parties start the protocol with the same value, then this value is the output (*validity*). In Broadcast, a single designated party, called the *sender*, has an input, and the requirements are that all honest parties agree on the same value, which equals the sender's input whenever the sender is honest.

The bulk of the literature on BA considers a known set of $n$ parties who are connected with each other via a complete network of reliable and authenticated channels. In this setting, BA (both Consensus and Broadcast) can be achieved if and only if at most $t < n/3$ parties are malicious, unless a trusted setup (e.g., a public key infrastructure for digital signatures, PKI) is available [LSP82, FLM86, Bor96]. Assuming such a PKI setup, Broadcast is achievable for an arbitrary number $t < n$ of corruptions [DS83], and Consensus for $t < n/2$ corruptions, a bound also known to be tight [Fit02].

The assumption of a known universe of participants has been implicit in the BA literature—and more generally in the fault-tolerant distributed computing literature—for over four decades. The advent of blockchains, however, fueled by the massive success of Bitcoin, highlighted the importance of permissionless models where the upper bound on the number of participants is unknown *a priori*, and parties may dynamically join or leave during execution (*dynamic availability*) [PS17, GKL17, BGK+18]. These assumptions have become mainstream in the blockchain setting and novel instantiations of state-machine replication [Sch90] (a.k.a. "Nakamoto Consensus" or "Ledger Consensus").

This leads to the foundational question regarding the feasibility of BA when the set and number of participants are **unknown** and with fluctuating and dynamic participation; a model we refer to as the *unknown participants* (UP) model. Current blockchain systems assume that the majority of some resource is in honest hands—e.g., hashing power in Proof-of-Work-based system or amount of stake in a Proof-of-Stake-based system. This so-called *resource-restricted setting* [GKO+20], allows for replacing assumptions on the honest majority of parties by assumptions on the honest majority of resources; making such resource assumptions has proven sufficient for constructing BA in the UP model [PS17, GKL17, BGK+18]. However, the question of whether BA (in particular Broadcast) is possible in this UP model without assuming an honest majority of neither resources nor parties has not yet been addressed. This is the question we tackle in this work:

> *Can a sender broadcast a value to whomever is available, without* a priori *knowledge of the number and the identities of participants, and without any assumptions on number of corruptions or distribution of resources?*

**Our results in a nutshell.** The core obstacle in the UP setting is local termination detection: without knowing the total number of participants $n$ or an upper bound on corruptions $t$, honest parties cannot directly determine when sufficient information has been received to terminate safely without risking consistency. Classical protocols like Dolev-Strong crucially depend on terminating at round $t + 1$, an explicit bound unavailable in the UP setting.

[1] The literature often uses the term BA in a somewhat ambiguous manner. Here we will use the terms *consensus* and *broadcast* to make explicit which of the two flavors we are referring to.

We resolve this challenge by establishing feasibility results for UP Broadcast that match the guarantees of the known-participants setting. Specifically:

- **Formal framework:** We formulate a cryptographic model for BA with unknown participants (UP) that accommodates static and dynamic participation, adaptively corrupted parties, and an overall potential-participant pool exponential in the security parameter.
- **Feasibility of UP Broadcast:** We prove feasibility of UP Broadcast by constructing a polynomial-time *deterministic* protocol secure against any number of corruptions. As a building block, we also solve *UP Interactive Consistency* (the multi-sender variant).
- **Tight round complexity:** We establish matching lower bounds proving that our deterministic protocols achieve optimal round complexity, a result that holds even when considering *randomized* protocols.
- **Security for late joiners:** We show an impossibility result for unrestricted dynamic participation where parties join late at arbitrary points. On the positive side, we formulate best-possible guarantees for such "late" parties and provide matching round-optimal constructions.

### 1.1 Technical Overview

Our work paints a detailed picture of the feasibility landscape of UP Broadcast in the dishonest-majority setting with arbitrarily many corruptions, which includes several protocols with new technical ideas and corresponding lower bounds. We begin by presenting a snapshot of these results and techniques, and outline the primary challenges that the UP model introduces.

**The unknown-participants model.** One of the technical challenges in this work is formalizing the UP model itself; unlike prior models that assume an honest majority of either parties or resources, our goal is to eliminate such assumptions entirely. Without such assumptions, the set of all potential protocol participants, which we call the universe $\mathcal{U}$, can potentially be exponentially larger than the actual number of protocol participants.[2] We note that this deviates from the so-called *ad-hoc* model for secure multi-party computation [BGIK16, BIK17], which assumes a polynomial universe of parties. Furthermore, parties in the universe have no *a priori* knowledge about each other. We carefully formalize our UP model to appropriately define Byzantine broadcast in our setting.

Denote by $\kappa$ the security parameter. The universe $\mathcal{U}$ of parties can be exponentially large in $\kappa$. Without loss of generality, in order to simplify how we refer to parties in our protocols, we will assume that each party in $\mathcal{U}$ is associated with a unique ID from $\{0,1\}^{\mathsf{poly}(\kappa)}$, which is known to this party, but not necessarily to other parties. Inspired by simulation-based security definitions, we consider a PPT environment $\mathcal{Z}$ and a PPT adversary $\mathcal{A}$. The environment $\mathcal{Z}$ can *activate* polynomially many parties in $\mathcal{U}$ that will actually participate in the protocol, and the adversary $\mathcal{A}$ can corrupt activated parties and control them in an arbitrary, malicious manner. One caveat is that the set of active parties (and, in particular, the set of honest parties) can change even within an execution of the protocol. Therefore, as opposed to standard definitions of Broadcast, we *cannot* consider a fixed and known set of parties in the protocol; rather, we refer to specific protocol *executions*. We denote an execution of the protocol by $\mathcal{E}$, and emphasize again that the number of parties and their identities can change between one execution and the next.

[2] In fact, all the challenges of the UP model (both definitional and in terms of feasibility and efficiency results) already appear if this set is polynomially bounded, but by a polynomial which is larger than the one bounding the protocol's running time. Nonetheless, for full generality, we do not put any such polynomial player-set restriction here.

Another definitional challenge stems from the setup assumptions. A minimal assumption for BA without an honest majority (even with known participants) is the existence of a public key infrastructure (PKI). However, the exponential size of the universe renders vanilla PKI models, where every party has a public/secret key pair and a complete list of all public keys in the system, unsuitable for the UP model. Indeed, no PPT entity (not even the environment and the adversary) can store the public keys of all parties; doing so would require exponential space and time, invalidating the computational security of the underlying signature schemes. Nevertheless, because $\mathcal{Z}$ runs in polynomial time, only polynomially many parties are ever activated in any given execution.

To address this issue, we draw inspiration from how public-key infrastructures operate in practice (e.g., on the Internet): Rather than assuming that everyone stores all PKI keys of every node on the Internet, parties rely on a trusted certification authority (CA) to certify their keys as needed. Parties presenting certified public keys can then convince peers to add them to their local view of the PKI. Analogously, instead of vanilla cryptographic PKI we rely on assuming such an *ad hoc PKI* model, where a trusted CA certifies the public keys of parties as they become active.[3] Further details on the above CA-based PKI model are included in Section 2.1. It is worth mentioning that CA-based setup for signatures have been used in the cryptographic literature, e.g., [Can04].

**Modeling UP broadcast.** Recall that in the UP setting the set of participating parties may change in each execution (as opposed to standard BA); this must be taken into account when defining UP Broadcast. This is relatively straightforward in the static setting (which is used for our lower bounds), where the environment activates all parties and the adversary chooses whom to corrupt before the protocol begins. We require that all active parties that are honest in the execution agree on their output, which equals the sender's input if the sender is *activated* and honest.

Defining security in the dynamic setting (used for our protocol constructions) is considerably more subtle. Roughly speaking, we require that parties that are activated from the beginning and never get corrupted by $\mathcal{A}$, reach agreement within a polynomial number of rounds. In addition, if the sender is activated from the beginning and does not get corrupted, then this common output must equal the sender's input. In Section 2.2, we discuss subtle challenges with this model.

**Problems with existing approaches.** One can easily observe that classical dishonest-majority broadcast protocols (e.g., Dolev-Strong [DS83], *DS-Broadcast* for short) cannot satisfy the UP Broadcast definition. For example, it is crucial in DS-Broadcast that all honest parties run for $t+1$ rounds (for $t < n$), but how can this be done if $n$ and $t$ are unknown?

One might be tempted to believe that the above shortcoming of DS-Broadcast in the UP model can be addressed by early-stopping protocols [DRS90]. In a nutshell, such protocols aim to reduce the number of rounds from a function of the upper bound on the number of corrupted parties $t$, to a function of the *actual* number of faults $f$. Examples of such results exist both in the information-theoretic setting [BGP92, AD15a], with protocols terminating in $\min(t+1, f+2)$ rounds assuming $t < n/3$, and in the cryptographic setting [Per85, LN24], which require $\min(t+1, f+2)$ rounds for $t < n/2$, and $O(\min(t, f^2))$ rounds for $t < n$.

However, all early-stopping BA literature suffers from similar issues as DS-Broadcast when run in the UP model. In particular, classical early-stopping protocols aim to correlate their termination with a runtime-identifiable property, namely, whether or not consensus has been achieved. Now,

[3] Note that one might prefer to replace the above assumption by assuming that a trusted CA has pre-certified all public keys before the protocol starts. This might simplify the exposition, but we prefer to avoid assuming in a cryptographic model exponential space/time trusted setups, even though they are only used in a pre-protocol phase.

detecting consensus requires an honest majority which makes their approach inapplicable here. In fact, with the exception of [LN24, DKLZ25], all such protocols assume an honest majority, in addition to knowing the total number of parties. Regardless, the methodology of [LN24, DKLZ25] can also not be used in the UP setting, as it relies on the phase-king approach [BGP89], which assumes an ordering of the players and selects a sequence of kings according to this ordering. Without prior agreement or knowledge of the active players, it is unclear if or how such a king-selection can be realized—in fact, even randomizing the selection does not seem to help as we do not have (assumptions allowing) cryptographic sortition.

The above discussion highlights that the problem with classical approaches lies at a much deeper level. Specifically, already the size of $\mathcal{U}$ (which, recall, is exponential in $\kappa$) poses a challenge: Existing round-complexity lower bounds for *deterministic* broadcast show linear dependency on the number of corruptions [DS83]. The same holds for *randomized* broadcast with an arbitrary number of corruptions [GKKO07, CPS20].[4] Hence, in the dishonest-majority case, this translates to linearly many corruptions in the number of active parties that participate in the protocol. Since the parties (i.e., the protocol) have no information on this number (which is not even fixed when the protocol begins—only when it ends), the number of rounds would need to be linear in $|\mathcal{U}|$, which is exponential in the security parameter $\kappa$. But this means that no such existing protocol can retain its security, since they all must rely on a trusted setup and computational assumptions, which can be broken by a super-polynomial adversary.[5]

**Active-parties set agreement.** The above limitation of the classical model (with known and polynomially many participants) sheds light on what needs to be done in the UP model:

> *Enable parties to agree on (the cardinality of) the set of active parties.*

But how can we do that in the UP model, where an adversarial environment might activate new parties at will during the protocol execution? We achieve this goal via a new primitive we introduce, termed *active-parties set agreement* (APSA) (see Section 3). In a nutshell, APSA allows parties to agree on a set $S$ that includes every party that has been active (and honest) since the beginning of the protocol's execution and until its completion, potentially along with some additional corrupted parties that joined during the execution.

To understand our key technical insight in solving APSA, it is useful to recall the structure of DS-Broadcast: The protocol begins with the sender sending its signed input to every party; then in each subsequent round $r$, every party verifies that it has received a strong enough "certificate" (effectively, a chain including signatures from $r-1$ distinct parties) on the sender's input, appends its own signature, and forwards the updated certificate. The key observations are that (1) this forwarded certificate will be perceived as valid by any honest party in the next round, and (2) since there are at most $n$ parties, the final round in which any party can be newly convinced is round $n$ (because a certificate obtained in that round already contains signatures from all other parties, including all honest ones).

A strawman approach to APSA would be to have every party act as the sender and have the parties agree on the set of senders. This approach is problematic for two main reasons. First,

[4] When a constant fraction of the parties is guaranteed to be honest, the lower bound for randomized protocols is poly-logarithmic [GKKO07, CPS20]; however, this assumption is overly restrictive for the UP setting, where no such bound is guaranteed.

[5] In fact, this super-polynomial gap between the number of potential parties and active parties is not only the natural choice for the UP model, but is also a key point that differentiates the UP setting from the classical (known-participants) one, and eliminates the trivial solution of running DS-Broadcast for $|\mathcal{U}|$ rounds.

we want to agree on parties that have been honest from the beginning—in fact, as we argue in Section 7, validity guarantees cannot be offered for senders who are not honest at the outset. One might hope that requiring parties to also sign the current round solves this problem, but this fails because the adversary can forge certificates for corrupted parties. We resolve this issue via a more sophisticated certificate generation and verification procedure that takes into account not only the set of signatures included in a DS-Broadcast-like certificate, but also their *order*. We note that a challenge here is to make this mechanism efficient, i.e., avoid storing and forwarding all received permutations, which would yield an LSP-like "exponential information gathering" approach [LSP82, BDDS92].

The second challenge is even more fundamental: without prior knowledge of the total number of participants, how can parties terminate in a manner that guarantees no other party continues running (and potentially adds more parties to its local set of active parties, violating consistency)? We resolve this via a novel termination mechanism that detects when the environment has stopped activating new parties for a sufficiently long interval, thus ensuring that honest views remain aligned.

**UP Interactive Consistency.** The APSA protocol described above lies at the heart of our UP Broadcast construction. Building on that protocol, we construct an Interactive Consistency (IC) protocol [PSL80] in the UP model (UPIC). Interactive consistency is a classical primitive corresponding to parallel broadcast (without input independence): Each party holds an input, and the output is the ordered vector containing the broadcast values of all participants (where corrupted parties may adaptively bias their broadcast values based on those of honest players).

A straightforward approach to solving UPIC would be to first execute APSA and agree on the set of participants, followed by parallel instances of DS-Broadcast for each detected sender, ignoring messages from any party not included in the APSA output. While this approach is sound, since our guarantees hold for parties active from the beginning, it effectively doubles the round complexity.

Instead, in Section 4 we design a UPIC protocol that preserves optimal round complexity. We achieve this via a white-box modification of APSA that enables the parties to agree simultaneously on the set $S$ of active parties and on an input for each party in $S$, guaranteeing validity for the inputs of all honest parties in $S$.

**Constructing UP Broadcast.** Given UPIC, constructing a UP Broadcast protocol is straightforward, since IC is strictly more general than Broadcast: all parties execute UPIC, where the designated sender (if activated at the beginning) uses its actual input, while all other parties input a default value. In fact, this reduction cleanly resolves a subtle edge case in the UP Broadcast definition, dealing with an *inactive sender* (i.e., a sender who is not active from the start, see Section 2.2). An honest inactive sender remains silent and is thereby excluded from the active set $S$, prompting the protocol to output $\perp$ whenever the sender is not in $S$; see Section 5.

We note that all our protocol constructions achieve security against adaptive and rushing adversaries. These guarantees represent the UP analog of the standard property-based definitions and do not imply *corruption fairness* [CGZ23]. Incorporating corruption-fairness and achieving simulation-based security is an interesting research direction.

**Round-complexity lower bounds.** In any execution, our protocols require at most as many rounds as the number of activated parties. A natural question is whether this round complexity is optimal or if one can do better.

To answer this, we establish in Section 6 a strong lower bound showing that our protocols are optimal: If the adversary can corrupt any number of the active parties, then it can make the

protocol run for as many rounds as there are activated parties. We prove this result by refining the lower bound by Garay *et al.* [GKKO07]. Specifically, we construct a chain of $k$ indistinguishable scenarios such that, for each $i \in [k]$, certain parties cannot safely decide before round $i$ without violating security.

Interestingly, requiring the protocol to output $\perp$ for an inactive sender yields an additional scenario beyond those in [GKKO07] (namely, the scenario where the sender is never activated), thereby increasing the lower bound by 1. As in [GKKO07], our proof extends to randomized protocols with probabilistic termination, confirming the tight round complexity of our UP Broadcast protocol.

**Late-joining parties.** The definition of UP Broadcast discussed so far provides no guarantees to late-joining parties, i.e., parties activated after the protocol begins but before it terminates. In Section 7, we study UP Broadcast that accommodates such late-joining parties. We consider two variants of the problem: *security for late-joining receivers* and *security for a late-joining sender*.

Informally, security for late-joining receivers (Definition 7.1) requires late-joining honest parties to produce outputs consistent with those parties that have been honest and active from the start. In Lemma 7.2, we prove that no deterministic protocol can satisfy this guarantee when late-joining receivers can be activated at arbitrary rounds. Nonetheless, we show how to modify our UP Broadcast protocol to achieve the next best thing: security with *selective abort* for late-joining receivers (Definition 7.3). Intuitively, this ensures that late-joiners either obtain the same output as parties active from the start, or abort upon detecting an adversarial corruption that prevents convergence. The formal protocol and security proof for this variant appear in Theorem 7.4.

Finally, we turn to security for a late-joining sender (Definition 7.5), which informally requires that the sender's input is successfully included in the output of the parties. In Lemma 7.6 we show that this notion is impossible to satisfy for deterministic protocols. However, this impossibility does not apply if the sender joins sufficiently early, before honest parties are close to terminating. When there is a known upper bound $R$ on the number of rounds by which the sender may be delayed, we show in Theorem 7.8, that the simple strategy of delaying the start of the protocol by $R$ rounds is optimal in terms of round complexity.

### 1.2 Related Work

The seminal results by Lamport, Shostak, and Pease [PSL80, LSP82] triggered a massive literature across multiple research communities studying various aspects of BA in several models. Below, we review the most relevant to ours, omitting the classical feasibility results discussed in the previous section.

An important separating line in the BA literature, often blurred during discussions of feasibility, is the distinction between classical consensus and *continuous consensus* (also known as state machine replication (SMR) [Sch90], or modern "ledger consensus"). Classical consensus is a "single-shot" protocol, where parties start with an input and aim is to terminate with a consistent output. Ledger consensus, on the other hand, is a ongoing protocol, whereby parties continuously receive new inputs. The goal is to incrementally build an agreed-upon distributed ledger that guarantees honest inputs are eventually processed.

In fact, the term "continuous consensus" can be slightly misleading, as the underlying protocol is closer to Byzantine broadcast than consensus. Indeed, in the classical synchronous model with fixed, known-participants, any Byzantine broadcast protocol offering simultaneous termination guarantee (where all parties terminates in the same round) can easily yield a continuous consensus protocol: senders simply broadcast their inputs, which are then ordered by their output round (resolving ties

in the same round via an arbitrary rule, such as lexicographical ordering). Importantly, reducing consensus to broadcast in the reverse direction is impossible. We refer the reader to Garay and Kiayias [GK20] for a survey and systematization of both classical and modern variants.

**BA with an honest majority.** Alchieri et al. [ABdSFG08] studied a setting where an unknown subset from a fixed and known set of parties reaches consensus. Their setting differs substantially than ours as they focused on perfect security over an asynchronous network, and could tolerate $t < n/3$ corruptions and $2t+1$connectivity , where $n$ is the actual number of participants. Khanchandani and Wattenhofer [KW21] considered classical consensus with an unknown *number* of participants. They constructed protocols where participants possess unique but non-consecutive identifiers, and where both the total number of parties $n$ and the number of corruptions $t$ are unknown (yet fixed) under the assumption that $t < n/3$.

Despite sharing similar high-level goals, the underlying assumptions in [ABdSFG08, KW21] make their results less surprising. First, for regarding system modeling, they assumes information-theoretic perfect security with $t < n/3$ corruptions under static participation and adversary models. This makes the objective significantly easier: the protocol begins by having every active party contact all other nodes in the network. Because all honest parties ($n - t > 2t + 1$) participate, this phase yields a reliable estimate of the total and honest population sizes, which can then be refined as parties relay their local views.

The $t < n/3$ threshold is critical to this approach, allowing for simple and elegant solutions. However, these techniques break down in the dishonest-majority setting ($t \geq n/2$), where any such approximation of corrupted parties will be inherently unreliable—a corrupted party can simply send inconsistent messages to different receivers. Indeed, a primary technical challenge in the dishonest-majority regime is establishing a consistent view of participating parties without relying on majority voting. While this obstacle exists even under static environments, it becomes far more pronounced in the dynamic and adaptive settings we consider. Beyond the increased technical complexity, formalizing these guarantees under dishonest majorities requires dedicated foundational work: without an honest majority, broadcast requires a signature setup, and formalizing such a setup in the UP setting demands a novel, cryptographically sound model of executions and corruptions.

**The fully permissionless model of Lewis-Pye and Roughgarden.** Lewis-Pye and Roughgarden [LR23a, LR23b] studied consensus in a setting where at any point the protocol has no knowledge about which parties are currently running it, and as such similar in spirit to ours. They proved several impossibility results, most relevant to our setting: (1) consensus cannot be solved facing any constant fraction of corruptions, (2) every deterministic consensus protocol must have an infinite execution facing any constant fraction of corruptions, and (3) in the dynamic-availability setting, no consensus protocol exists even when there are no corrupted parties.

However, fundamental differences in both technical modeling and definitions separate our work from theirs. Our results neither contradict with nor are implied by [LR23a, LR23b].

- First, these works consider the consensus problem that requires an honest majority even with a known set of parties [Fit02], whereas we consider broadcast (and IC); hence, their impossibility (1) does not apply to our case.
- Second, their impossibility (2) relies on two modeling choices: allowing a potentially *infinite* number of participants and assuming idealized cryptography with *perfect* correctness; in this case, they proved an FLP-style result where a measure-zero set of executions fails to terminate.

In contrast, we consider a more cryptography-compatible model with a *finite* (yet exponential-size) universe and a polynomial number of actual participants (for an arbitrary polynomial), and as we rely on cryptographic assumption, there is a negligible (non-zero) failure probability, which circumvents this impossibility.

– Finally, their impossibility (3) considers a partially synchronous model, which requires a 2/3 honest majority even with a fixed and known party set. As we consider the dishonest-majority setting, we assume synchronous communication, so this result does not apply to our setting.

**Continuous consensus.** Despite being new to the classical BA setting, several works considered models with unknown participants in the continuous-consensus setting. Indeed, interest in such models was largely sparked by Bitcoin and subsequent surge in "permissionless" blockchains. Although our work focuses on classical dishonest-majority broadcast (where the techniques differ significantly), we briefly survey these works for completeness and highlight the main differences. A line of work has conducted rigorous analyses of blockchain protocols, typically assuming (either implicitly or explicitly) that the universe of potential participants is polynomial in size. This includes foundational analyses of Bitcoin in the fixed-difficulty setting [GKL17, PSs17, BMTZ17] as well as extensions to variable difficulty [GKL24].

A model with fluctuating participation was introduced by Pass and Shi on in their work *sleepy consensus* [PS17]. In their *sleepy model*, parties can transition dynamically between online and offline states. Pass and Shi noted that classical consensus protocols perform poorly in this regime because the number of currently awake players may be vastly smaller than the total population. They instantiated a sleepy continuous-consensus protocol under strong assumptions, namely a PKI, a common random strings, sub-exponentially secure collision-resistant hash functions, enhanced trapdoor permutations, and a continuous honest majority (at any point in the execution, a majority of the awake players are honest). Their techniques, while operating in a more general, partially synchronous model, do not apply to dishonest majorities even under their strong assumptions. The latency of Pass and Shi's protocol was later improved by Malkhi, Momose, and Ren [MR22, MMR23].

The sleepy model was later applied to the proof-of-stake setting Snow-white [DPS19] and further generalized to the dynamic-availability framework used to analyze Ouroboros Genesis [BGK+18]. Both of these protocols rely on an honest majority of total stake. Similar to the honest majority of hashing power assumption used in Bitcoin, this offers an alternative mechanism to bound the adversary's capabilities: parties' pseudonyms take the role of parties in the execution, under the assumption that the number of corrupted pseudonyms is balanced by correlating the number of pseudonyms with the amount of resource held by their owners. (We refer to [AD15b, GKO+20, CDM25] for models that abstract this resource-restricting cryptography paradigm.) In contrast, our goal is to achieve broadcast in the cleanest possible setting—analogous to classical BA—without relying on honesty assumptions or on resource balance among participants. As a side note, our model also avoids a key network assumption which is crucial in [PS17], namely that messages sent to an offline (sleeping) party are eventually delivered upon rejoining. In our setting, a party offline during a given round permanently misses the messages sent in that round and obtains only the guarantees (if any) of a late-joiner, based on the round it joins.

**Round complexity of (classical) BA.** Dolev and Strong [DS83] established that any deterministic, $t$-secure BA protocol requires at least $t + 1$ rounds. Ben-Or [Ben83] and Rabin [Rab83] demonstrated that randomized protocols can circumvent this lower bound. Leveraging this approach, Feldman and Micali [FM97] presented an expected constant round randomized consensus

protocol tolerating up to $t < n/3$ corruptions. This result was later extended to the computational setting by Katz and Koo [KK06], constructed an expected constant-round consensus protocol tolerating $t < n/2$ corruptions assuming a PKI and digital signatures.

Garay et al. [GKKO07] pushed the feasibility boundaries of expected constant-round BA to $t = n/2 + O(1)$ corruptions for Broadcast (as consensus is impossible in this regime). They proved this bound to be asymptotically optimal: no broadcast protocol can terminates in less than $2n/(n-t)-1$ rounds when $n - t \geq 2$. Fitzi and Nielsen [FN09] proposed randomized protocols achieving $O(k)$ expected rounds complexity when $t = n/2 + k$. More recently, Wan et al. [WXSD20] extended the feasibility of expected constant round broadcast to any constant fraction of corruptions. We note that all of these works operate within the classical setting of fixed, known participants.

**Adaptive (adversary) security in BA.** Hirt and Zikas [HZ10] demonstrated that adaptive security introduces subtle feasibility barriers for BA. In particular, the classical property-based definition of broadcast fails to imply the simulation-based definition for broadcast (the so-called Megaphone functionality [CLOS02, GL05]) when an adversary can adaptively corrupt parties in the non-atomic multisend model (when corruptions can occur in the middle of transmission while messages are still being routed). To capture such attacks, Hirt and Zikas defined a weaker, *corruption-unfair* broadcast functionality.

Cohen *et al.* [CGZ23] extended the impossibility result to stronger property-based broadcast definitions enforcing corruption fairness: where the adversary cannot first learn the input of an honest party and then corrupt that party and change its input. As they showed, circumventing this impossibility requires strong assumptions like atomic multisend (an approach taken in [GKKZ11]) or time-lock puzzles (TLPs). To avoid the technical complications associated with adaptive security, we consider the classical definition of broadcast, without corruption fairness. Again, all the above works are in the classical, known participants setting.

**Organization of the paper.** In Section 2 we lay out the details of the UP model and define the UP Broadcast problem. In Section 3 we construct an active-parties set agreement (APSA) protocol, and in Section 4 an Interactive Consistency protocol (UPIC). These building blocks are combined into a UP Broadcast protocol in Section 5. Our lower bounds are proven in Section 6. Finally, in Section 7 we treat the case of late-joining parties.

## 2 Modeling Broadcast with Unknown Participants

We start by describing the unknown-participants model in Section 2.1, and then define the problem of Broadcast with unknown participants in Section 2.2.

### 2.1 The Unknown-Participants Model

We begin by formally defining the *unknown-participants* (UP) model. As becomes evident from our discussion, the model itself involves several subtle technical choices. Thus, while the primary focus of this paper is on protocol constructions and lower bounds, we view this formalization as an independent contribution.

**The universe.** In standard BA, parties have complete knowledge of the number and identities of all other parties in the network. In contrast, the *unknown-participants* (UP) model aims to achieve BA among parties who have no *a priori* knowledge about the identities or even an estimate of

the number of other potential protocol participants. We assume that each party is identified by a unique string, which is unknown to the protocol itself (as discussed below).

A caveat is that because parties run in polynomial time the security parameter $\kappa$ (otherwise they will be able to forge signatures) and at the same time be able to at least read the public key of other parties, the size of these strings must also be polynomial in $\kappa$. To capture this we consider a (potentially exponentially large) set of parties $\mathcal{U}$, which we refer to as the *universe*. For simplicity, we assume that each party in $\mathcal{U}$ is associated a unique party ID from $\{0,1\}^{\mathsf{poly}(\kappa)}$. Furthermore, to keep the protocol description simple we will use a party's name and ID interchangeably; e.g., whenever we make a statement about a party $p \in \mathcal{U}$, we mean the party with the corresponding ID. Importantly, no party (except for the adversary, as discussed below) possesses information about which, or how many, other parties are actually participating in an execution of the BA protocol with them.

**The environment and the adversary.** To keep our treatment clean and concise, we adopt property-based security definitions. However, to capture the asymmetry of knowledge mentioned above while making as general statements as possible, it will be useful to incorporate ideas from simulation-based security. In particular, we consider a PPT Interactive Turing Machine (ITM), referred to as the *environment* $\mathcal{Z}$, that can activate parties from $\mathcal{U}$ at its discretion.

We consider two types of environments based on when party activations occur. A *static* environment $\mathcal{Z}$ will activate all participating parties from $\mathcal{U}$ at the beginning of the protocol execution, whereas a *dynamic* environment may activate additional parties throughout the execution. We emphasize that because the environment runs in polynomial time, even though the size of $\mathcal{U}$ can be exponential, only a polynomially large subset of parties can be activated.

In addition to the environment, we assume an adversary $\mathcal{A}$, also modeled as a PPT ITM, that can corrupt activated parties and control them in a malicious manner. Again, we consider two standard adversarial models: *static* adversaries (where all corruptions are made before the protocol begins) and *adaptive* adversaries (where corruptions can occur during the execution).

We stress that a party is *activated* only upon receiving a special input from its environment. In particular, parties cannot be activated by the adversary (e.g., by receiving a protocol message from a corrupted party). Any message sent by the adversary to an inactive party is ignored. This separation of roles between activating and corrupting simplifies our definitions and theorem statements. We remark that since the environment can activate any polynomial (in $\kappa$) number of parties, allowing any PPT adversary ensures that there are adversaries able to corrupt any number of parties. As discussed below, corrupted-majority adversaries are what makes the design of protocols particularly challenging.

To strengthen our results, our negative results will hold for static adversary and environment (the least adversarial scenario). For our protocols, we consider adaptive adversaries in the standard formulation in the MPC literature (see [CGZ23]) where the adversary can corrupt a party right after the party sends its messages to some of the parties, but before completing transmission to all recipients.[6] This is also the standard model for capturing adaptive corruptions in the MPC literature (e.g., [CFGN96, Can00, Can01, CCL15]). We note that, as discussed in Section 1.2, we do not mandate *corruption fairness* in our definitions, rendering our framework compatible with both this strong adaptive and weaker adaptive corruption models considered in the literature.

**Active parties and honest parties in an execution.** A key nuance of the UP model is that the number and identities of active, inactive, honest, and corrupted parties depend on both

[6] This capability has also been referred to as "strongly rushing"; see, e.g., [ADD+19, ACD+19, WXDS20].

the adversary and the environment. Consequently, to make formal statements regarding security properties, we must refer to specific protocol *executions* rather than to the protocol in isolation. This distinguishes the UP model from standard models, where only the adversary's corruption choices need to be considered.

To capture these nuances, we introduce notation and terminology used throughout our definitions and theorem statements. We start with the simple case of a static environment $\mathcal{Z}$ and adversary $\mathcal{A}$. Let $\mathcal{E}$ denote (the transcript[7] of) an execution of a protocol in the UP model with such a static environment and adversary. We denote by $\mathcal{P}_{\mathcal{E}} \subseteq \mathcal{U}$ the set of parties activated by $\mathcal{Z}$, and by $\mathcal{H}_{\mathcal{E}} \subseteq \mathcal{P}_{\mathcal{E}}$ the set of activated parties who are not corrupted by $\mathcal{A}$.

The situation becomes more involved when considering a dynamic environment $\mathcal{Z}$ and an adaptive adversary $\mathcal{A}$. To handle this case, we parameterize this notation by round. For every $r \in \mathbb{N}^+$, let $\mathcal{P}^r_{\mathcal{E}} \subseteq \mathcal{U}$ denote the set of parties activated in any round $r' \leq r$. Let $R_T \in \mathbb{N}^+$ be the first round after which $\mathcal{Z}$ activates no new parties (referred to as the round where $\mathcal{Z}$ *terminates*), then for every $r \geq R_T$, denote $\mathcal{P}^r_{\mathcal{E}} = \mathcal{P}^{R_T}_{\mathcal{E}}$. Because $\mathcal{Z}$ is PPT in $\kappa$, the termination round $R_T$ is also polynomially bounded. Similarly, we denote by $\mathcal{H}^r_{\mathcal{E}} \subseteq \mathcal{P}^r_{\mathcal{E}}$ the set of parties in $\mathcal{P}^r_{\mathcal{E}}$ who are not corrupted by $\mathcal{A}$ at the end of round $r$.

We note that our formulation is different from the *sleepy* model of consensus [PS17, DPS19] (also known as *dynamic availability* [BGK+18]), where parties may join *and leave* at any point in the protocol. In our model, a party that joins (i.e., becomes active) remains active throughout but might get corrupted by the adversary. This choice simplifies both the presentation and analysis, while naturally fitting our focus on single-instance broadcast rather than continuous consensus.

**Network and synchrony.** We assume synchronous communication, where the protocol proceeds in rounds. In contrast to standard BA literature, our constructions do not rely on pairwise point-to-point channels between parties (indeed, pairwise channels are ill-defined in the UP setting). Instead, we consider an abstract communication model any message sent by a party is guaranteed to be delivered to all parties active in that same round, as in the *multicast network* model of [BMTZ17, ACD+19]. The adversary remains rushing and can send different messages to different honest parties in each round.

We adopt this abstraction to simplify the protocol exposition. In the motivating scenario of the UP model—where parties are unaware of other parties—this model can be realized via standard mechanisms from the blockchain literature, such as gossip protocols operating over an incomplete-but-connected synchronous network.

**Ad-hoc PKI setup.** Classical impossibility results in the standard (known-participants) model [LSP82, FLM86, Bor96] imply that UP Broadcast is impossible without restricting the adversary's capabilities in some way. We adopt the traditional approach of employing digital signatures alongside a public-key infrastructure (PKI). However, extra care is needed to model such a PKI in our setting, as the PPT parties (as well as the adversary and the environment) cannot store exponentially many public keys.

To resolve this challenge, we draw inspiration from modern large-scale public-key infrastructures (like the Internet). Rather than an exponentially large pre-distributed PKI, we assume an incorruptible *certification authority* (CA) holding a key-pair $(\mathsf{sk}_{\mathrm{CA}}, \mathsf{vk}_{\mathrm{CA}})$ for a digital signature scheme, where the public verification key $\mathsf{vk}_{\mathrm{CA}}$ is known to all protocol participants. When a party $p$ is activated for the first time, $p$ generates its own signature keys $(\mathsf{sk}_p, \mathsf{vk}_p)$ and sends $\mathsf{vk}_p$ to the CA.

[7] Such a transcript consists of all the inputs, outputs, randomness, setup, and incoming and outgoing messages of all parties, the adversary, and the environment.

The CA signs $\mathsf{vk}_p$ together with the identity of $p$ to issue a certificate $\mathsf{cert}_p \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathrm{CA}}}((\mathsf{id}_p, \mathsf{vk}_p))$ and returns $\mathsf{cert}_p$ to $p$.

Importantly, this CA is never used during the protocol execution other than issuing certificates. Instead, whenever a party $p$ sends a signed message $\sigma_p \leftarrow \mathsf{Sign}_{\mathsf{sk}_p}(m)$ to another party $q$, party $p$ attaches its certified public key $(\mathsf{id}_p, \mathsf{vk}_p, \mathsf{cert}_p)$ to this message.[8] To verify $\sigma_p$, party $q$ first checks that $\mathsf{Verify}_{\mathsf{vk}_{\mathrm{CA}}}((\mathsf{id}_p, \mathsf{vk}_p), \mathsf{cert}_p) = 1$, and then that $\mathsf{Verify}_{\mathsf{vk}_p}(m, \sigma_p) = 1$. We implicitly assume that this CA-credential verification step is part of any signature exchanged in this work.

We abstract digital signatures using the modular approach of Canetti's UC framework [Can01, Can20] and will use an ideal signature functionality $\mathcal{F}_{\mathsf{sig}}$ [Can04, GKZ10]. Specifically, we employ the ideal signature functionality from [CDL+25], which ensures error-free ideal signature services. This formulation allows us to prove *perfect* security for protocols that use digital signatures in a black-box manner (in the spirit of Dolev-Strong) and later apply the UC composition theorem to establish computational security when instantiating $\mathcal{F}_{\mathsf{sig}}$ with any signature scheme that is correct, existentially unforgeable under chosen-message attack (EUF-CMA), consistent (i.e., verifying consistently across all parties even under maliciously generated keys, except for negligible probability), and linear time (i.e., signing complexity is linear in the message length). For brevity, whenever we refer to EUF-CMA signatures, we mean a signature scheme satisfying these four properties.

This framework also enables us to define a *deterministic* protocol that uses digital signatures, i.e., a protocol that is deterministic (parties do not use their random tapes) and *perfectly secure* in the $\mathcal{F}_{\mathsf{sig}}$-hybrid model, which yields a computationally secure protocol when $\mathcal{F}_{\mathsf{sig}}$ is instantiated with standard cryptographic signatures.

## 2.2 Byzantine Broadcast in the UP Model

We formalize Byzantine broadcast in the unknown-participants (UP) model by extending classical definitions [PSL80, LSP82] to accommodate dynamic and unknown sets of active participants.

To build intuition, we begin with the setting of a static environment and a static adversary, where the sets of active and corrupted parties are fixed at protocol initialization. This static variant is used in our impossibility results and highlights the key idiosyncrasies of the UP model. Specifically, the UP setting deviates from standard Byzantine broadcast in two fundamental ways:

1. Security properties cannot be stated as static attributes of the protocol itself, because they inherently depend on the active party set $\mathcal{P}_{\mathcal{E}}$, which is unknown to the protocol *a priori*.
2. The protocol must execute gracefully even if the designated sender is *inactive*, in which case all honest parties must output $\perp$. This requirement adds significant technical complexity to protocol design.

**Definition 2.1** (UP Broadcast: static)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$ be a designated sender. A protocol $\pi$ is a* statically secure UP Broadcast protocol with sender *$s$ if for every static PPT environment $\mathcal{Z}$, every static PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$ satisfies the following properties:*

- ***Validity:*** *Every honest party $p$ terminates with output $m_p \in \{0,1\}^\ell \cup \{\perp\}$, where $m_p = m_s$ if $s \in \mathcal{H}_{\mathcal{E}}$ and $m_p = \perp$ if $s \notin \mathcal{P}_{\mathcal{E}}$.*

[8] In fact, party $p$ need only send its certified public key during its first interaction with $q$; for simplicity, we assume that all signed messages include this certificate.

– ***Consistency:*** *For all honest parties $p$ and $q$ which output $m_p$ and $m_q$, respectively, it holds that $m_p = m_q$.*

– ***Termination:*** *Every honest party terminates and produces an output in a polynomially bounded (in $\kappa$) number of rounds.*

We next extend the definition to dynamic environments and adaptive adversaries. This setting introduces additional subtleties regarding which parties receive security guarantees (e.g., initially active versus late-joining parties) and how these guarantees are formalized. Consider, for example, adapting *validity* to the adaptive setting: If the sender is honest at the end of the protocol, then every honest party outputs the sender's input. There are several questions that this description leaves open. For example, (1) What if the sender did not join the protocol at the beginning? and (2) What constitutes "the end" of the protocol? Indeed, the environment might activate parties at any point so the "end" must be defined by means of looking at the parties' outputs.

There are several ways to answer the above questions, each giving guarantees of different strength. We choose a natural extension of the static definition. For Question 2, because protocols in this model may not exhibit simultaneous termination, the completion of an execution is parameterized per party by the round in which that specific party terminates and outputs. For Question 1, validity guarantees that if the sender is active at the beginning ($s \in \mathcal{P}^1_{\mathcal{E}}$) and remains uncorrupted through the termination round $r$ of an initially active honest party $p$, then $p$ outputs $m_s$. The formal definition follows:

**Definition 2.2** (UP Broadcast: dynamic)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$ be a designated sender. A protocol $\pi$ is an adaptively secure UP Broadcast protocol with sender $s$, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$ satisfies the following properties:*

– ***Validity:*** *Let $r$ be a round number and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ be a party that terminates in round $r$ with output $m$. Then $m \in \{0,1\}^\ell \cup \{\perp\}$, where $m = m_s$ if $s \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$, and $m = \perp$ if $s \notin \mathcal{P}^r_{\mathcal{E}}$.*

– ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r_p}_{\mathcal{E}}$ and $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r_q}_{\mathcal{E}}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $m_p$ and $m_q$, respectively. Then $m_p = m_q$.*

– ***Termination:*** *Every party in $\mathcal{P}^1_{\mathcal{E}}$ either terminates in a polynomially bounded (in $\kappa$) number of rounds or is corrupted prior to termination.*

We note that Definition 2.2 restricts its guarantees to parties active from the beginning. In Section 7, we extend this definition for *late-joining* parties.

# 3 Active-Parties Set Agreement Protocol

In this section, we provide an *active-parties set agreement* (APSA) protocol hat tolerates any number of corruptions in the UP model. In Section 3.1, we adapt the idea of Dolev-Strong broadcast to the UP model and provide a high-level technical description. We present the formal protocol specification in Section 3.2 and its security proof in Section 3.3.

We begin by formally describing the properties required from an APSA protocol. As discussed in the introduction, our description uses parties and their identifiers interchangeably. So the output of an APSA protocol is in fact an agreed upon set of party identifiers. Analogous to UP

Broadcast (Section 2.2), we first present the definition for static environments and adversaries, and subsequently extend the formulation to dynamic environments and adaptive adversaries.

**Definition 3.1** (APSA: static)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter and let $\mathcal{U}$ be a (possibly exponential) set of parties. A protocol $\pi$ is a statically secure active-parties set agreement (APSA) protocol in the UP setting, if for every static PPT environment $\mathcal{Z}$ and every static PPT adversary $\mathcal{A}$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$ satisfies the following properties:*

- ***Validity:*** *Every honest party $p$ outputs a set $S_p$, such that $\mathcal{H}_{\mathcal{E}} \subseteq S_p \subseteq \mathcal{P}_{\mathcal{E}}$.*
- ***Consistency:*** *For all honest parties $p$ and $q$ that output $S_p$ and $S_q$, respectively, it holds that $S_p = S_q$.*
- ***Termination:*** *Every honest party terminates and outputs a set in a polynomially bounded (in $\kappa$) number of rounds.*

**Definition 3.2** (APSA: dynamic)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter and let $\mathcal{U}$ be a (possibly exponential) set of parties. A protocol $\pi$ is an adaptively secure active-parties set agreement protocol in the UP setting if for every dynamic PPT environment $\mathcal{Z}$ and every adaptive PPT adversary $\mathcal{A}$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$ satisfies the following properties:*

- ***Validity:*** *Let $r$ be a round number, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ be a party that terminates in round $r$ with output set $S_p$. Then $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}} \subseteq S_p \subseteq \mathcal{P}^r_{\mathcal{E}}$.*
- ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r_p}_{\mathcal{E}}$ and $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r_q}_{\mathcal{E}}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $S_p$ and $S_q$, respectively. Then $S_p = S_q$.*
- ***Termination:*** *Every party in $\mathcal{P}^1_{\mathcal{E}}$ either terminate in a polynomially bounded (in $\kappa$) number of rounds or is corrupted prior to termination.*

Note that in APSA, parties hold no initial input. Furthermore, output guarantees apply exclusively to parties that are activated at the beginning of the execution ($\mathcal{P}^1_{\mathcal{E}}$) and remain honest until termination.

### 3.1 Adapting Dolev-Strong to the UP Model

To achieve agreement on a set of active parties, we adapt the classical Dolev-Strong (DS) broadcast protocol [DS83] following the approach in Section 1.1. In particular, instead of transferring signed certificate chains for the input of a designated sender, parties propagate signatures endorsing the active participation of other parties, accompanied by proofs of activity. A party $p \in \mathcal{U}$ is convinced of the active participation of another party $u \in \mathcal{U}$ if and only if it verifies $u$'s signature on its identity $\mathsf{id}_u$, along with a CA-issued certificate binding $u$'s identifier to its verification key.

The protocol proceeds as follows: In the first round, every activated party $p$ signs $\mathsf{id}_p$ and sends it to all active parties. In every round $r > 1$, a party $p$ accepts a previously unaccepted party $u$ as active in round $r$ if and only if it observes signatures on $\mathsf{id}_u$ from at least $r-1$ distinct parties including $u$, where every signer (excluding $u$) was already accepted as active by $p$ by round $r-1$.

Note that accepting a new party in round $r$ requires $p$ to have accepted at least $r-1$ parties (including itself) by round $r-1$. Consequently, if $p$ holds at most $r-1$ accepted parties at the end of round $r$, it cannot accept any additional parties in subsequent rounds and may safely terminate.

Otherwise, to ensure that all other honest parties learn of newly accepted party $u$, $p$ appends its own signature on $\mathsf{id}_u$ to the observed signature chain in support of the participation of $u$, and broadcasts the augmented chain to all active parties in round $r$.

To guarantee *consistency*, $p$ must be assured that no honest party terminates before round $r+1$ without accepting $u$. Intuitively, every party in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^1_{\mathcal{E}}$ convinces all parties in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^2_{\mathcal{E}}$ of its participation by round 2, forcing them to advance to round 3. Similarly (by induction), any party in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ can convince every party in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r+1}_{\mathcal{E}}$ in round $r+1$ regarding all parties accepted by round $r-1$ (including itself), ensuring that all honest parties advance to round $r+2$. We formally prove this claim and analyze the protocol in Section 3.3.

### 3.2 The APSA Protocol

We proceed to present the formal specification of our APSA protocol. We start by defining *valid batches of signatures* and *well-formed messages*.

Because APSA outputs party identifiers, we recall that each party $p$ possesses a unique identifier $\mathsf{id}_p$ used to register its verification key $\mathsf{vk}_p$ with the CA. Recall also that, without loss of generality, every signature implicitly includes the signer's credentials; specifically, transmitting $\mathsf{Sign}_p(m)$ means sending the tuple $(\mathsf{id}_p, \mathsf{vk}_p, \mathsf{cert}_p, m, \sigma_p(m))$, where $\sigma_p(m)$ denotes the signature on $m$ under $p$'s signing key $\mathsf{sk}_p$.

**Valid batches of signatures.** Let $S^r_p \subseteq \mathcal{U}$ denote the set of parties accepted as active by a party $p$ by round $r \geq 1$. Let $B$ be a collection of signatures on $\mathsf{id}_u$, for some party $u \in \mathcal{U}$. We say $B$ is a $(p,r)$-*valid batch of signatures* for $u$ if $B$ contains valid signatures on $\mathsf{id}_u$ from at least $r-1$ distinct parties (including $u$), such that at least $r-2$ of them are included in $S^{r-1}_p$.

Formally, for $B = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V}$ with $V \subseteq \mathcal{U}$, the batch $B$ is $(p,r)$-valid if $u \in V$ and $|V \cap (S^{r-1}_p \setminus \{u\})| \geq r-2$.

**Well-formed messages.** In every round $r \geq 1$, honest parties that proceed to round $r+1$ send batches of signatures endorsing all newly accepted parties. To bound message complexity and prevent corrupt parties from sending unbounded messages, an active party $p$ includes at most one signature batch per newly accepted party in its round-$r$ message.

A message $M^r_p$ sent by party $p$ in round $r$ is *well-formed* if $M^r_p = \{B_i\}_{i=1}^k$ for some $k \geq 1$, where each $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i})\}_{v \in V_i}$ is a signature batch for $u_i \in \mathcal{U}$ with $u_i \in V_i$, and the targets $u_1, \ldots, u_k$ are pairwise distinct. Honest parties parse only well-formed messages and discard all malformed transmissions.

The formal protocol specification is presented in Figure 1.

**Protocol $\pi_{\mathsf{apsa}}$**

**Upon first activation:** Once a party $p \in \mathcal{U}$ with $\mathsf{id}_p \in \{0,1\}^{\mathsf{poly}(\kappa)}$ is activated for the first time with the security parameter $\kappa$ and the CA's verification key $\mathsf{vk}_{\mathrm{CA}}$, $p$ generates signing key pair $(\mathsf{sk}_p, \mathsf{vk}_p)$ and obtains a certificate $\mathsf{cert}_p$ from the CA for $(\mathsf{id}_p, \mathsf{vk}_p)$.

**Notation:** Given a message $m$, we denote by $\mathsf{Sign}_p(m)$ the tuple $(\mathsf{id}_p, \mathsf{vk}_p, \mathsf{cert}_p, m, \sigma_p(m))$, where $\sigma_p(m)$ is a signature on $m$ using $\mathsf{sk}_p$.

**Round 1:** Every active party $p \in \mathcal{U}$ initializes $S^1_p := \{p\}$ and sends $\{\{\mathsf{Sign}_p(\mathsf{id}_p)\}\}$ to all active parties.

**Loop:** In round $r$, for $r \geq 2$, every active party $p \in \mathcal{U}$ does the following:

1. Initializes $S^r_p := S^{r-1}_p$.

2. Upon receiving a well-formed message $M$, for every $B \in M$, do the following: if $B$ is a $(p, r)$-valid batch of signatures for some party $u \notin S_p^{r-1}$, then:
   - if $u \notin S_p^r$, initialize the set $C_{p,u}^r := \emptyset$,
   - set $S_p^r := S_p^r \cup \{u\}$ and $C_{p,u}^r := C_{p,u}^r \cup B$.
3. If $|S_p^r| \leq r - 1$, then terminate with output $S_p^r$.
   Else, send $\{C_{p,u}^r \cup \{\mathsf{Sign}_p(\mathsf{id}_u)\}\}_{u \in S_p^r \setminus S_p^{r-1}}$ to all active parties and proceed to round $r + 1$.

Figure 1: Active-Parties Set Agreement Protocol.

### 3.3 APSA Protocol Analysis

We proceed to formally prove that protocol $\pi_{\mathsf{apsa}}$ satisfies Definition 3.2.

**Theorem 3.3.** *Assuming the existence of EUF-CMA signatures, Protocol $\pi_{\mathsf{apsa}}$ (Figure 1) is an adaptively secure APSA protocol (according to Definition 3.2) in the CA-model.*

*Moreover, in every execution $\mathcal{E}$ of $\pi_{\mathsf{apsa}}$, all parties in $\mathcal{P}_{\mathcal{E}}^1$ that terminate uncorrupted do so simultaneously in at most $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds, where $R_T$ denotes the final round in which the environment activates new parties.*

*Proof.* Let $\mathcal{Z}$ be a dynamic PPT environment and let $\mathcal{A}$ be an adaptive PPT adversary. Let $\mathcal{E}$ be an execution of $\pi_{\mathsf{apsa}}$ under $\mathcal{Z}$ and $\mathcal{A}$. We prove that *validity*, *consistency*, and *termination* hold perfectly in $\mathcal{E}$ when signatures are modeled ideally in the $\mathcal{F}_{\mathsf{sig}}$-hybrid model; this implies that the properties hold with overwhelming probability over the randomness of $\mathcal{Z}$, $\mathcal{A}$, active parties, and the CA when $\mathcal{F}_{\mathsf{sig}}$ is instantiated with EUF-CMA signatures (note that active parties and the CA use their randomness only for the signature scheme).

In the proof, we say that party $p$ *accepts* party $u$ in round $r$, if $r$ is the first round in which $u$ is in $S_p^r$, i.e., $r = \min\{r' \in \mathbb{N}^+ : u \in S_p^{r'}\}$. We prove validity in Lemma 3.4, consistency in Lemma 3.5, and simultaneous termination in Lemma 3.7.

**Lemma 3.4** (Validity)**.** *Let $r$ be a round number and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ be a party that terminates in round $r$ with output set $S_p$. Then, $\mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r \subseteq S_p \subseteq \mathcal{P}_{\mathcal{E}}^r$.*

*Proof.* First, for any $u \in S_p \setminus \{p\}$, party $p$ received a $(p, r')$-valid batch of signatures, denoted by $B$, for $u$ in some round $r' \leq r$. By the definition of valid batches, $\mathsf{Sign}_u(\mathsf{id}_u) \in B$ implying $u \in \mathcal{P}_{\mathcal{E}}^r$ and thus $S_p \subseteq \mathcal{P}_{\mathcal{E}}^r$.

It remains to show that $\mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r \subseteq S_p$. By construction, $p \in S_p^1 \subseteq S_p$. Moreover, every party $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ with $q \neq p$ sends $\{\{\mathsf{Sign}_q(\mathsf{id}_q)\}\}$ at the end of round 1, which $p$ receives in round 2. Since $q \notin S_p^1$ and $\{\mathsf{Sign}_q(\mathsf{id}_q)\}$ forms a $(p, 2)$-valid batch of signatures for $q$, party $p$ accepts $q$ in round 2. So, $q \in S_p^2 \subseteq S_p$, completing the proof. □

To prove *consistency*, it suffices to show that whenever an uncorrupted party $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ accepts a party $u$ in round $r$, every party $q \in \mathcal{P}_{\mathcal{E}}^1$ that terminates uncorrupted also accepts $u$ before terminating. We prove this statement by induction on the round number $r$.

**Lemma 3.5** (Consistency)**.** *Let $r$ be a round number and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ be a party which accepts some party $u$ in round $r$. Then, every party $q \in \mathcal{P}_{\mathcal{E}}^1$ that terminates uncorrupted accepts $u$ by round $r + 1$ before terminating.*

*Proof.* First, we establish the base cases $r = 1$ and $r = 2$. In round 1, every party $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^1$ accepts only itself and sends $\{\{\mathsf{Sign}_p(\mathsf{id}_p)\}\}$ to all active parties. Hence, every party $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^2$,

with $q \neq p$, receives $p$'s message in round 2 and accepts $p$, since $p \notin S_q^1 = \{q\}$ and $\{\mathsf{Sign}_p(\mathsf{id}_p)\}$ is a $(q,2)$-valid batch of signatures for $p$.

For round 2, let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^2$. Party $p$ accepts a party $u \neq p$ in round 2 upon receiving a $(p,2)$-valid batch of signatures for $u$ of the form $\{\{\mathsf{Sign}_u(\mathsf{id}_u)\}\}$. Party $p$ then sends $\{\{\mathsf{Sign}_u(\mathsf{id}_u)\}, \{\mathsf{Sign}_p(\mathsf{id}_u)\}\}$ to all active parties. Consider a party $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^2$: if $q = u$ then $q$ accepted $u$ in round 1; if $q = p$ then $q$ accepted $u$ in round 2. If $q \notin \{p,u\}$, then either $q$ accepted $u$ in round 2, or $q$ receives $p$'s message in round 3 and accepts $u$, since $u \notin S_q^2$ (since $q$ did not accept $u$ yet) and $\{\{\mathsf{Sign}_u(\mathsf{id}_u)\}, \{\mathsf{Sign}_p(\mathsf{id}_u)\}\}$ is a $(q,3)$-valid batch of signatures for $u$.

Now let $r \geq 3$ and assume that inductive claim holds for every round $r' < r$. Let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ and assume that $p$ accepts $u$ in round $r$. Then $p$ receives a $(p,r)$-valid batch of signatures $B = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V}$ for $u$ in round $r$, where $u \notin S_p^{r-1}$ and that $|V \cap S_p^{r-1}| \geq r-2$. By definition of a valid batch, there exists a subset $W \subseteq V \cap S_p^{r-1}$ with $|W| \geq r-2$ such that $p$ accepted every party of $W$ by round $r-1$. Unforgeability of $\mathcal{F}_{\mathsf{sig}}$ guarantees that $p \notin V$ and thus $p \notin W$ (since $p \in V$ would imply that $p$'s signature on $\mathsf{id}_p$ was forged prior to $r$).

Let $q \in \mathcal{P}_{\mathcal{E}}^1 \setminus \{p\}$ be some party that terminates uncorrupted. Because $p$ accepted all parties in $W$ by round $r-1$, the induction hypothesis implies that $q$ accepts all parties in $W$ by round $r$. We consider two cases:

1. If $q$ accepted $u$ by round $r$, the claim holds directly.
2. Otherwise, $q$ has not accepted $u$ by round $r$, so $q \notin W$ (as all honest parties in $W$ accepted $u$ before round $r$). Let $r' \leq r$ be round in which $q$ accepted final party of $W$ (note that $r' \geq 2$ since $q \notin W$). Party $q$ accepts itself in round 1 and accepts $p$ in round 2 (since $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^1$), which means that $W \cup \{p,q\} \subseteq S_q^{r'}$. Consequently, for every round $r'' \in [r', r]$ it holds that

$$|S_q^{r''}| \geq |S_q^{r'}| \geq |W \cup \{p,q\}| \geq r > r'' - 1,$$

where the first inequality holds because $r'' \geq r'$ hence $S_q^{r'} \subseteq S_q^{r''}$, the second because $W \cup \{p,q\} \subseteq S_q^{r'}$, the third because $|W| \geq r-2$ and $p, q \notin W$, and the fourth because $r \geq r''$.

Therefore, in every round $r'' \in [r', r]$, it holds that $|S_q^{r''}| > r'' - 1$, and thus $q$ participates in round $r+1$. Since $p$ is honest and accepted $u$ in round $r$, it sends a well-formed message in round $r$ containing a batch of signatures $B' = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V'}$ for $u$, with $V \cup \{u\} \subseteq V'$. Party $q$ receives $B'$ in round $r+1$ and observes that $u \in V' \setminus S_q^r$ and that $|V' \cap S_q^r| \geq |W \cup \{p\}| \geq r-1$. Thus, $q$ accepts $u$ in round $r+1$. □

**Lemma 3.6** (Termination)**.** *Let $R_T \in \mathbb{N}$ be the round in which $\mathcal{Z}$ terminates. Every party in $\mathcal{P}_{\mathcal{E}}^1$ either terminates in at most $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds or gets corrupted prior to termination.*

*Proof.* Let $\overline{R} = |\mathcal{P}_{\mathcal{E}}^{R_T}|$ and let $p \in \mathcal{P}_{\mathcal{E}}^1$ be a party that remains honest and does not terminate by round $\overline{R}$. Then $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{\overline{R}}$ and for every $u \in S_p^{\overline{R}}$, party $p$ holds a signature $\mathsf{Sign}_u(\mathsf{id}_u)$. Thus, after receiving round-$\overline{R}$ messages (i.e., when processing messages for round $\overline{R}+1$, but before sending messages for round $\overline{R}+1$) party $p$ observes $S_p^{\overline{R}+1} \subseteq \mathcal{P}_{\mathcal{E}}^{\overline{R}} \subseteq \mathcal{P}_{\mathcal{E}}^{R_T}$, i.e., $|S_p^{\overline{R}+1}| \leq \overline{R}$. Therefore, $p$ terminates after round $\overline{R}$ (before sending round $\overline{R}+1$ messages). □

**Lemma 3.7** (Simultaneous Termination)**.** *All parties in $\mathcal{P}_{\mathcal{E}}^1$ that terminate uncorrupted terminate simultaneously in the same round.*

*Proof.* Let $S_p$ be the output of a party $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ terminating in round $r$. Party $p$ evaluates $S_p = S_p^r$ and $|S_p^r| \leq r-1$. If $|S_p^r| < r-1$, then $|S_p^{r-1}| \leq |S_p^r| \leq r-2$, which would have triggered termination in round $r-1$, yielding a contradiction. Thus $r-1 = |S_p^r| = |S_p|$.

By Lemma 3.5, all parties that terminate uncorrupted output the same set $S_p$. Hence, every uncorrupted party terminates in round $r = |S_p| + 1$. □

Lemmas 3.4, 3.5, 3.6, and 3.7 together complete the proof of Theorem 3.3. □

**From APSA to UP Broadcast.** The simultaneous termination property of $\pi_{\mathsf{apsa}}$ provides a natural foundation for constructing UP Broadcast. Specifically, one can solve UP Broadcast by executing APSA to establish consensus on a set of active parties, and subsequently running DS-Broadcast on the agreed party set. However, this solution doubles the number of rounds required for termination.

In the next section, we extend our APSA protocol into an *interactive consistency* protocol in the UP model that completes in the exact same number of rounds as APSA. We then leverage this interactive consistency protocol to achieve UP Broadcast without round overhead.

# 4 Interactive Consistency with Unknown Participants

In this section, we formalize and construct an adaptively secure *unknown-participants interactive consistency* (UPIC) protocol tolerating an arbitrary number of corruptions. We begin by defining the security properties of UPIC. Next, in Section 4.1, we present a high-level overview of our construction. In Section 4.2, we describe the formal protocol, and in Section 4.3, provide its security proof.

In the UPIC problem, every active party $p$ holds an initial input value $m_p$, and all uncorrupted terminating parties must agree on a common set of party-value pairs. Each pair $(u, m)$ indicates that $m$ is agreed upon as the input value of party $u$; if $u$ was activated at the beginning of the execution and remained honest until the time that $p$ terminates, then $m$ must be equal to $m_u$. Analogous to our treatment of broadcast and APSA, we formulate security definitions for both static and dynamic/adaptive settings.

**Definition 4.1** (UPIC: static environments & adversaries)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $\ell = O(\kappa)$. A protocol $\pi$ is a statically secure UP interactive consistency protocol, if for every static PPT environment $\mathcal{Z}$, every static PPT adversary $\mathcal{A}$, and every set of input messages $\{m_p \in \{0,1\}^\ell : p \in \mathcal{U}\}$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$, satisfies the following properties:*

- ***Validity:*** *Every party $p \in \mathcal{H}_\mathcal{E}$ outputs a set of party-value pairs $A_p = \{(q, m'_q)\}_{q \in S_p}$, for some $S_p \subseteq \mathcal{U}$, such that*
  1. *$\mathcal{H}_\mathcal{E} \subseteq S_p \subseteq \mathcal{P}_\mathcal{E}$.*
  2. *For every $q \in S_p$, it holds that $m'_q \in \{0,1\}^\ell \cup \{\perp\}$, and if $q \in \mathcal{H}_\mathcal{E}$ then $m'_q = m_q$.*
- ***Consistency:*** *If parties $p, q \in \mathcal{H}_\mathcal{E}$ output $A_p$ and $A_q$, resp., then $A_p = A_q$.*
- ***Termination:*** *Every honest party terminates and produces an output in a polynomially bounded (in $\kappa$) number of rounds.*

**Definition 4.2** (UPIC: dynamic environments & adaptive adversaries)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $\ell = O(\kappa)$. A protocol $\pi$ is an adaptively secure UP interactive consistency protocol, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every set of input messages $\{m_p \in \{0,1\}^\ell : p \in \mathcal{U}\}$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$, satisfies the following properties:*

– ***Validity:*** *Let $r$ be a round number, and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ be a party terminating in round $r$ with output $A_p = \{(q, m'_q)\}_{q \in S_p}$, for some $S_p \subseteq \mathcal{U}$. Then,*

  1. *$\mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r \subseteq S_p \subseteq \mathcal{P}_{\mathcal{E}}^r$.*
  2. *For every $q \in S_p$, it holds that $m'_q \in \{0,1\}^\ell \cup \{\perp\}$, and if $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ then $m'_q = m_q$.*

– ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{r_p}$ and $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{r_q}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $A_p$ and $A_q$, respectively. Then $A_p = A_q$.*

– ***Termination:*** *Every party in $\mathcal{P}_{\mathcal{E}}^1$ either terminates in a polynomially bounded (in $\kappa$) number of rounds or is corrupted prior to termination.*

### 4.1 Building on the APSA Protocol

Having established a protocol for agreement on the set of active parties (which includes all honest parties), we use it as the basis of an UPIC protocol.

In round 1, every active party $p$, signs $\mathsf{id}_p \| m_p$ (rather than $\mathsf{id}_p$ alone) and sends it to all active parties. Throughout the execution, parties propagate batches of signatures supporting party-value pairs of the form $\mathsf{Sign}_v(\mathsf{id}_u \| m)$ for $u, v \in \mathcal{U}$ and $m \in \{0,1\}^\ell$. Acceptance of a party-value pair $(u, m)$, which also implies acceptance of the participation of $u$, proceeds analogously to party acceptance in our APSA protocol.

The key modification lies in the output representation: every party $p$ produces a set of party-value pairs specifying the decided input message for each accepted party. Specifically, for each accepted party $u$, party $p$ decides the default value $\perp$ if it accepts multiple distinct pairs $(u, \cdot)$ across the execution, or the unique value $m \in \{0,1\}^\ell$ if $(u, m)$ is the sole pair accepted for $u$. Agreement on the output set follows from a relay property analogous to Lemma 3.5, now applied to party-value pairs.

### 4.2 The UPIC Protocol

We proceed to present the formal description of our UPIC protocol. We first extend the definitions of *valid signature batches* and *well-formed messages* from Section 3.2. The formal protocol is given in Figure 2.

**Valid batches of signatures.** We extend the notion of valid batches of signatures to party-value pairs. Let $B$ be a batch of signatures on a party-value pair $(u, m)$. We say $B$ is a $(p, r)$-*valid batch of signatures* if it contains signatures on $\mathsf{id}_u \| m$ from at least $r-1$ distinct parties including $u$, such that at least $r-2$ of them are included in $S_p^{r-1} \setminus \{u\}$.

Formally, for $B = \{\mathsf{Sign}_v(\mathsf{id}_u \parallel m)\}_{v \in V}$ with $V \subseteq \mathcal{U}$, the set $B$ is $(p, r)$-valid if $u \in V$ and $|V \cap (S_p^{r-1} \setminus \{u\})| \geq r - 2$.

**Well-formed messages.** In the UPIC protocol, honest parties send batches of signatures endorsing newly accepted party-value pairs. To maintain the efficiency of APSA, a party includes at most one batch of signatures for each newly accepted pair. A message $M_p^r$ sent by $p$ in round $r$ is *well-formed* if $M_p^r = \{B_i\}_{i=1}^k$ for some $k \geq 1$, where each $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i} \| m_i)\}_{v \in V_i}$ is a signature batch for a distinct pair $(u_i, m_i)$ with $u_i \in V_i$.

**Output.** Upon termination, parties extract their output from the values received thus far using the function $\mathsf{Out}$. Let $A_p^r$ be the set of party-value pairs accepted by $p$ by round $r$. For every $u \in S_p^r$, let $A_{p,u}^r = \{m \in \{0,1\}^\ell : (u,m) \in A_p^r\}$ be the set of values accepted by $p$ for $u$ by round $r$. Upon terminating in round $r$, party $p$ outputs $m'_u$ for party $u$ if $A_{p,u}^r = \{m'_u\}$ is a singleton, and $\perp$ if $|A_{p,u}^r| > 1$. Formally:

$$\mathsf{Out}(A_p^r) = \Big\{(u, m'_u)\Big\}_{u \in \hat{S}_p^r} \bigcup \Big\{(u, \perp)\Big\}_{u \in S_p^r \setminus \hat{S}_p^r},$$

where $\hat{S}_p^r = \Big\{u \in S_p^r : \exists m'_u \in \{0,1\}^\ell \text{ such that } A_{p,u}^r = \{m'_u\}\Big\}$.

For signature tracking, each party $p$ maintains sets $C_{p,u,m}^r$ in every round $r > 1$, storing all signatures observed in support of each newly accepted party-value pair $(u,m) \in A_p^r \setminus A_p^{r-1}$. Message construction at the conclusion of round $r$ follows the same procedure as in Protocol $\pi_{\mathsf{apsa}}$.

**Protocol $\pi_{\mathsf{ic}}$**

**Upon first activation:** Once a party $p \in \mathcal{U}$ with $\mathsf{id}_p \in \{0,1\}^{\mathsf{poly}(\kappa)}$ is activated for the first time with security parameter $\kappa$, CA verification key $\mathsf{vk}_{\mathrm{CA}}$, and input $m_p \in \{0,1\}^\ell$, party $p$ generates signing key pair $(\mathsf{sk}_p, \mathsf{vk}_p)$ and certifies $(\mathsf{id}_p, \mathsf{vk}_p)$ with the CA to obtain $\mathsf{cert}_p$.

**Notation:** Given a message $m$, we denote by $\mathsf{Sign}_p(m)$ the tuple $(\mathsf{id}_p, \mathsf{vk}_p, \mathsf{cert}_p, m, \sigma_p(m))$, where $\sigma_p(m)$ is a signature on $m$ using $\mathsf{sk}_p$.

**Round 1:** Every active party $p \in \mathcal{U}$ initializes $S_p^1 := \{p\}$ and $A_p^1 := \{(p, m_p)\}$ and sends $\{\{\mathsf{Sign}_p(\mathsf{id}_p \| m_p)\}\}$ to all active parties.

**Loop:** In round $r$ for $r \geq 2$, every active party $p \in \mathcal{U}$ does the following:

1. Initializes $S_p^r := S_p^{r-1}$ and $A_p^r := A_p^{r-1}$.
2. Upon receiving some well-formed message $M$, for all $B \in M$, do the following: if $B$ is a $(p,r)$-valid batch of signatures for some pair $(u,m) \notin A_p^{r-1}$, then:
   - if $(u,m) \notin A_p^r$, initialize the set $C_{p,u,m}^r := \emptyset$,
   - set $S_p^r := S_p^r \cup \{u\}$, $A_p^r := A_p^r \cup \{(u,m)\}$, and $C_{p,u,m}^r := C_{p,u,m}^r \cup B$.
3. If $|S_p^r| \leq r-1$, then terminate with output $\mathsf{Out}(A_p^r)$; else, send to all active parties the set

$$\Big\{C_{p,u,m}^r \cup \{\mathsf{Sign}_p(\mathsf{id}_u \| m)\}\Big\}_{(u,m) \in A_p^r \setminus A_p^{r-1}}$$

and proceed to round $r+1$.

Figure 2: UP Interactive Consistency Protocol.

### 4.3 Interactive Consistency Protocol Analysis

We proceed to prove that Protocol $\pi_{\mathsf{ic}}$ satisfies Definition 4.2. Similarly to the analysis of Protocol $\pi_{\mathsf{apsa}}$, we say that a party $p$ *accepts* a pair $(u,m)$ in round $r$ if $r$ is the first round in which $(u,m) \in A_p^r$, i.e., $r = \min\{r' \in \mathbb{N}^+ : (u,m) \in A_p^{r'}\}$.

Notice that the termination condition of $\pi_{\mathsf{ic}}$, as well as the tracking of accepted parties $S_p^r$, is identical to that of $\pi_{\mathsf{apsa}}$ (Figure 1). Further, the number of distinct values accepted with respect to a given party does not affect the termination criterion beyond the acceptance of the party identifier itself. Hence, the party acceptance and simultaneous termination properties established for $\pi_{\mathsf{apsa}}$ (Lemmas 3.5, 3.6, and 3.7) carry over directly to $\pi_{\mathsf{ic}}$.

**Theorem 4.3.** *Assuming the existence of EUF-CMA signatures, Protocol $\pi_{\mathsf{ic}}$ (Figure 2) is an adaptively secure UPIC protocol (according to Definition 4.2) in the CA-model.*

*Moreover, in every execution $\mathcal{E}$ of $\pi_{\mathsf{ic}}$, all parties in $\mathcal{P}^1_{\mathcal{E}}$ that terminate uncorrupted, terminate simultaneously in at most $|\mathcal{P}^{R_T}_{\mathcal{E}}|$ rounds, where $R_T$ is the last round in which the environment activates new parties.*

*Proof.* Let $\mathcal{Z}$ be a dynamic PPT environment and let $\mathcal{A}$ be an adaptive PPT adversary. Let $\mathcal{E}$ be an execution of $\pi_{\mathsf{ic}}$ under $\mathcal{Z}$ and $\mathcal{A}$. We prove that *validity*, *consistency*, and *termination* hold perfectly in $\mathcal{E}$ in the $\mathcal{F}_{\mathsf{sig}}$-hybrid model, which implies that the properties hold with overwhelming probability over the randomness of $\mathcal{Z}$, $\mathcal{A}$, the activated parties, and the CA, once $\mathcal{F}_{\mathsf{sig}}$ is instantiated with an EUF-CMA signature scheme.

**Lemma 4.4** (Validity)**.** *Let $r$ be a round number, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ be a party that terminates in round $r$ with output set $A_p$. Then, $A_p = \{(q, m'_q)\}_{q \in S_p}$, where:*

1. $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}} \subseteq S_p \subseteq \mathcal{P}^r_{\mathcal{E}}$.
2. *For every $q \in S_p$, it holds that $m'_q \in \{0,1\}^\ell \cup \{\perp\}$, and if $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ then $m'_q = m_q$.*

*Proof.* For every $u \in S^r_p \setminus \{p\}$, party $p$ must have received a $(p, r')$-valid batch of signatures, denoted by $B$, in support of $(u, m)$ for some $m \in \{0,1\}^\ell$, in some round $r' \leq r$. By definition of valid batch, $\mathsf{Sign}_u(\mathsf{id}_u \| m) \in B$, implying $u \in \mathcal{P}^r_{\mathcal{E}}$. Thus, by construction of $A_p = \mathsf{Out}(A^r_p)$, we have $A_p = \{(q, m'_q)\}_{q \in S_p}$, where $S_p \subseteq \mathcal{P}^r_{\mathcal{E}}$ and $m'_q \in \{0,1\}^\ell \cup \{\perp\}$ for every $q \in S_p$.

It remains to establish that $\{(q, m_q)\}_{q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}} \subseteq A_p$. First, $(p, m_p) \in A^1_p$. Every $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}} \setminus \{p\}$ sends $\{\{\mathsf{Sign}_q(\mathsf{id}_q \| m_q)\}\}$ at the end of round 1, which $p$ receives in round 2. Hence, $p$ accepts $(q, m_q)$ because $(q, m_q) \notin A^1_p$ and $\{\mathsf{Sign}_q(\mathsf{id}_q \| m_q)\}$ is a $(p, 2)$-valid batch of signatures for $(q, m_q)$. Further, if $p$ were to accept a pair $(q, m)$ with $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ and $m \neq m_q$, then $p$ must have received a $(p, r')$-valid batch of signatures $B'$ for $(q, m)$ in some round $r' \leq r$, such that $\mathsf{Sign}_q(\mathsf{id}_q \| m) \in B'$. This would imply that honest party $q$ signed $\mathsf{id}_q \| m$ prior to round $r$, yielding a contradiction, since $q$ is honest until round $r$. Therefore, by construction, $\{(q, m_q)\}_{q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}} \subseteq A^r_p \subseteq A_p$. □

To prove *consistency*, suppose that parties $p, q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ hold the sets $A^r_p$ and $A^r_q$ in round $r$ once they decide to terminate (simultaneous termination is guaranteed by Lemma 3.6). Consistency may be violated only if is there exists $(u, m) \in A^r_p \setminus A^r_q$. We prove that this cannot happen in the following lemma.

**Lemma 4.5** (Consistency)**.** *Let $r$ be a round number, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ be a party that accepts a party-value pair $(u, m)$ in round $r$. Then, every party $q \in \mathcal{P}^1_{\mathcal{E}}$ that terminates uncorrupted accepts $(u, m)$ by round $r + 1$ before terminating.*

*Proof.* We prove the statement by induction on $r$. In round 1, party $p$ accepts only $(p, m_p)$ and sends $\{\{\mathsf{Sign}_p(\mathsf{id}_p \| m_p)\}\}$. In round 2, every party $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^2_{\mathcal{E}} \setminus p$ receives $p$'s message and accepts $(p \| m_p)$, since $(p, m_p) \notin A^1_q$ and $\{\mathsf{Sign}_p(\mathsf{id}_p \| m_p)\}$ is a $(q, 2)$-valid batch of signatures for $(p, m_p)$.

For round $r = 2$, let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^2_{\mathcal{E}}$ accept $(u, m)$ in round 2 for $u \neq p$ upon receiving a $(p, 2)$-valid batch of signatures $\{\{\mathsf{Sign}_u(\mathsf{id}_u \| m)\}\}$ for $(u, m)$ in round 2. Party $p$ sends $\{\{\mathsf{Sign}_u(\mathsf{id}_u \| m)\}, \{\mathsf{Sign}_p(\mathsf{id}_u \| m)\}\}$ to all active parties. Consider a party $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^2_{\mathcal{E}}$: if $q = u$ then $q$ accepted $(u, m)$ in round 1; if $q = p$ then $q$ accepted $(u, m)$ in round 2. If $q \notin \{p, u\}$, then either $q$ accepted $(u, m)$ in round 2, or $q$ receives $p$'s message in round 3 and accepts $(u, m)$, since $(u, m) \notin A^2_q$ (since $q$ did not accept $(u, m)$ yet) and $\{\{\mathsf{Sign}_u(\mathsf{id}_u \| m)\}, \{\mathsf{Sign}_p(\mathsf{id}_u \| m)\}\}$ is a $(q, 3)$-valid batch of signatures for $(u, m)$.

Now assume $r \geq 3$ and that the claim holds for all rounds $r' < r$. Let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ accept $(u, m)$ in round $r$. Party $p$ receives a $(p, r)$-valid batch of signatures $B = \{\mathsf{Sign}_v(\mathsf{id}_u \| m)\}_{v \in V}$, with $(u, m) \notin A_p^{r-1}$ and $|V \cap S_p^{r-1} \setminus \{u\}| \geq r - 2$. Thus, there exists a set $W \subseteq V \cap S_p^{r-1} \setminus \{u\}$ with $|W| \geq r - 2$ such that $p$ accepted every party of $W$ by round $r - 1$. By unforgeability of $\mathcal{F}_{\mathsf{sig}}$ we have that $p \notin V$ and thus $p \notin W$ (since $p \in V$ would imply that $p$'s signature on $\mathsf{id}_u \| m$ was forged at some round prior to $r$).

Let $q \in \mathcal{P}_{\mathcal{E}}^1 \setminus \{p\}$ be some party that terminates uncorrupted. By the induction hypothesis, $q$ accepts all parties in $W$ by round $r$. Consider two cases:

1. If $q$ has accepted $(u, m)$ by round $r$, the claim holds directly.
2. Otherwise, $q$ did not accept $(u, m)$ by round $r$, so $q \notin W$. Let $r' \leq r$ be the round in which $q$ accepts the final party of $W$. Because $W \cup \{p, q\} \subseteq S_q^{r'}$, party $q$ observes

$$|S_q^{r''}| \geq |S_q^{r'}| \geq |W \cup \{p, q\}| \geq (r - 2) + 2 = r > r'' - 1$$

in every round $r'' \in [r', r]$. Thus, $q$ does not terminate early and advances to round $r + 1$. Since $p$ is honest in round $r$, it sends a well-formed message in round $r$ containing a batch of signatures $B' = \{\mathsf{Sign}_v(\mathsf{id}_u \| m)\}_{v \in V'}$ for $(u, m)$, with $W \cup \{p\} \subseteq V'$. Party $q$ receives $B'$ in round $r + 1$, observes $(u, m) \notin A_q^r$ and $|V' \cap S_q^r \setminus \{u\}| \geq |W \cup \{p\}| \geq r - 1$, and accepts $(u, m)$ in round $r + 1$. □

**Lemma 4.6** (Simultaneous Termination)**.** *All parties in $\mathcal{P}_{\mathcal{E}}^1$ that terminate uncorrupted terminate simultaneously in at most $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds, where $R_T$ denotes the final round in which $\mathcal{Z}$ activates new parties.*

*Proof.* Because party acceptance and termination mechanics in $\pi_{\mathsf{ic}}$ are identical to those in Protocol $\pi_{\mathsf{apsa}}$, simultaneous termination in at most $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds follows directly from Lemma 3.6 and Lemma 3.7. □

Lemmas 4.4, 4.5, and 4.6 together complete the proof of Theorem 4.3. □

# 5 Byzantine Broadcast with Unknown Participants

In this section, we present our UP broadcast protocol based on our UPIC protocol from Section 4.

**High-level idea.** In accordance with the classic model, UP broadcast can be reduced to UPIC by executing an instance of UPIC where the designated sender $s$ inputs $m_s$ while all other parties input a default value. Upon completing the UPIC execution, each party decides $m$ if $(s, m)$ is included in its output set, and outputs $\perp$ otherwise. In this way, the properties of *validity*, *consistency*, and *termination* of UPIC directly guarantee the corresponding properties for Broadcast.

We present a formal description of this protocol in Figure 3.

**Protocol $\pi_{\mathsf{bc}}$**

**Setup Parameters:** The protocol is parameterized by the identity of the sender $s$, and by the UPIC protocol $\pi_{\mathsf{ic}}$ in the CA model (Figure 2).

**UPIC:** Every party $p \in \mathcal{U}$ that is activated starts running $\pi_{\mathsf{ic}}$; the sender $s$ participates with its input $m_s$, and all other parties participate with input $\perp$.

**Output:** Upon receiving the output $A_p$ from $\pi_{\mathsf{ic}}$, every active party $p \in \mathcal{U}$ outputs $m$ if $(s, m) \in A_p$, for

some $m \in \{0,1\}^\ell$; otherwise, $p$ outputs $\perp$.

Figure 3: UP Broadcast Protocol.

**Theorem 5.1.** *Assuming the existence of EUF-CMA signatures, Protocol $\pi_{\mathsf{bc}}$ (Figure 3) is an adaptively secure UP Broadcast protocol (according to Definition 2.2) in the CA-model.*

*Moreover, in every execution $\mathcal{E}$ of $\pi_{\mathsf{bc}}$, all parties in $\mathcal{P}^1_{\mathcal{E}}$ that terminate uncorrupted, terminate simultaneously in at most $|\mathcal{P}^{R_T}_{\mathcal{E}}|$ rounds, where $R_T$ is the last round in which the environment activates new parties.*

*Proof.* Let $\mathcal{Z}$ be a dynamic PPT environment and let $\mathcal{A}$ be an adaptive PPT adversary. Let $\mathcal{E}$ be an execution of $\pi_{\mathsf{bc}}$ under $\mathcal{Z}$ and $\mathcal{A}$. Let $r$ be a round number, and let $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ be a party who terminates in round $r$ with output $m$. By construction $m \in \{0,1\}^\ell \cup \{\perp\}$.

- **Validity:** If the designated sender $s \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ is honest up to round $r$, then by Lemma 4.4 (UPIC Validity), $(s, m_s) \in A_p$, so $p$ outputs $m = m_s$. Conversely, if $s \notin \mathcal{P}^r_{\mathcal{E}}$, Lemma 4.4 guarantees that $(s, \cdot) \notin A_p$, and thus $p$ outputs $m = \perp$. Hence, the validity property of UP Broadcast holds.
- **Consistency and Simultaneous Termination:** By Lemma 4.6 (UPIC Simultaneous Termination), all parties in $\mathcal{P}^1_{\mathcal{E}}$ that terminate uncorrupted do so simultaneously in some round $r \le |\mathcal{P}^{R_T}_{\mathcal{E}}|$. Thus, every $q \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^r_{\mathcal{E}}$ terminates in round $r$. By Lemma 4.5 (UPIC Consistency), $A^r_p = A^r_q$, so $p$ and $q$ obtain identical output sets $A_p = A_q$ from $\pi_{\mathsf{ic}}$. Consequently, $p$ and $q$ output the same value $m$ by construction. □

# 6 Round-Complexity Lower Bounds

In this section, we present lower bounds for all three problems that studied in this paper: UP Broadcast, APSA, and UPIC. In particular, we show that any protocol that solves either of these problems, requires at least $|\mathcal{P}_{\mathcal{E}}|$ rounds in a worst case execution $\mathcal{E}$. To establish the strongest possible lower bounds, our proofs assume static $\mathcal{Z}$ and $\mathcal{A}$; under these conditions, the duration of an execution corresponds precisely to the number of rounds required for all honest parties to terminate and output a value. In Section 6.1 we present the lower bound for UP Broadcast, and in Section 6.2, for APSA and UPIC.

## 6.1 A Lower Bound for UP Broadcast

We start by showing that for any *randomized* protocol that solves even the binary version of UP Broadcast (where the input domain is $\{0,1\}$) in less than $|\mathcal{P}_{\mathcal{E}}|$ rounds in every execution $\mathcal{E}$, there exists a pair $(\mathcal{Z}, \mathcal{A})$ that causes the protocol to fail with non-negligible probability.

Our argument adopts the approach of Garay *et al.* [GKKO07], who showed that in the known-participants model every randomized broadcast protocol secure against $n-2$ corruptions requires at least $n-1$ rounds. We prove that in the UP setting, parties may actually need to run for at least one more round, which corresponds to $|\mathcal{P}_{\mathcal{E}}|$ rounds overall.

While in the classic model honest parties can ensure consistent views regarding the sender's behavior in $n-1$ rounds, in the UP model, where the sender might be inactive, an honest party might observe $n$ active parties (including itself) by round $n-1$ without knowing whether these parties received an input from the sender. Specifically, the party cannot distinguish between the scenario where the sender is inactive and the scenario where the sender's message was received by

all other parties except itself. Consequently, this party must run at least one more round to resolve the ambiguity.

We first present the core idea in Lemma 6.1 for the simplified case where the adversary corrupts all but two honest parties. Next, in Theorem 6.5 we extend the result to the general case.

**Lemma 6.1.** *There exist a static environment $\mathcal{Z}$ and a static adversary $\mathcal{A}$ such that the following holds for every randomized UP Broadcast protocol $\pi$: In a random execution $\mathcal{E}$ with $\mathcal{Z}$ and $\mathcal{A}$, in which $\mathcal{Z}$ activates party-set $\mathcal{P}_{\mathcal{E}}$ and $\mathcal{A}$ corrupts all but two parties, at least one honest party in $\mathcal{E}$ runs for at least $|\mathcal{P}_{\mathcal{E}}|$ rounds, except with negligible probability (where the probability is over the coins of $\mathcal{Z}$, $\mathcal{A}$, and the honest parties).*

*Proof.* Let $\pi$ be a UP Broadcast protocol and assume without loss of generality that the sender is $p_1$. Given the security parameter $\kappa$, let $\mathcal{P} = \{p_1, \ldots, p_\kappa\}$. Define two static environments:

- $\mathcal{Z}_1$ activates all parties in $\mathcal{P}$.
- $\mathcal{Z}_2$ activates the parties in $\mathcal{P} \setminus \{p_1\}$.

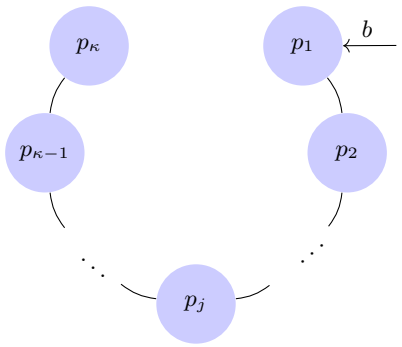
Next, define $\kappa$ adversaries, $\mathcal{A}_1, \ldots, \mathcal{A}_\kappa$, as follows:

- For $i \in [\kappa - 1]$, $\mathcal{A}_i$ corrupts all active parties in $\mathcal{P} \setminus \{p_i, p_{i+1}\}$.
- $\mathcal{A}_\kappa$ corrupts all active parties in $\mathcal{P} \setminus \{p_{\kappa-1}, p_\kappa\}$.

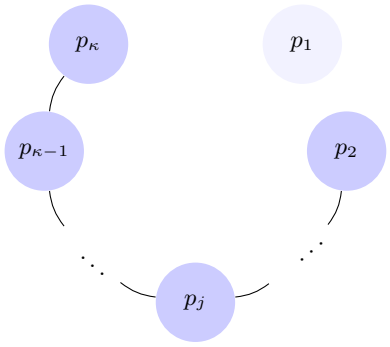
For every $i \in [\kappa]$, adversary $\mathcal{A}_i$ instructs every corrupt $p_j$ to follow $\pi$, but cuts all communications channels between $p_j$ and parties other than $p_{j-1}$ and $p_{j+1}$ (if active).[9] That is, $\mathcal{A}_i$ drops every message sent by $p_j$ to parties other than $p_{j-1}$ and $p_{j+1}$, and every message sent to $p_j$ from parties other than $p_{j-1}$ and $p_{j+1}$.

Given the above, we define the following scenarios (illustrated in Figure 4):

- For $i \in [\kappa - 1]$, scenario $S_i$ corresponds to executing $\pi$ with $\mathcal{Z}_1$ and $\mathcal{A}_i$. Observe that in every execution $\mathcal{E}$ of $S_i$, the number of active parties is $|\mathcal{P}_{\mathcal{E}}| = \kappa$, with $|\mathcal{H}_{\mathcal{E}}| = 2$ honest parties.
- Scenario $S_\kappa$ corresponds to executing $\pi$ with $\mathcal{Z}_2$ and $\mathcal{A}_\kappa$. In every execution $\mathcal{E}$ of $\pi$ of $S_\kappa$, the number of active parties is $|\mathcal{P}_{\mathcal{E}}| = \kappa - 1$, with $|\mathcal{H}_{\mathcal{E}}| = 2$ honest parties.




(a) Scenario $S_i$ (for $i \in [\kappa - 1]$) with $\mathcal{Z}_1$ and $\mathcal{A}_i$. Here only parties $p_i$ and $p_{i+1}$ are honest.




(b) Scenario $S_\kappa$ with $\mathcal{Z}_2$ and $\mathcal{A}_\kappa$. Here the sender $p_1$ is not active, and only parties $p_{\kappa-1}$ and $p_\kappa$ are honest.

Figure 4: Configurations corresponding to scenarios $S_1, \ldots, S_\kappa$. Every $p_j$ communicates only with parties $p_{j-1}$ and $p_{j+1}$ (if active).

Let $b \leftarrow \{0, 1\}$ be a uniformly distributed bit. Towards contradiction, assume that in every execution $\mathcal{E}$ of $\pi$ in scenarios $S_1, \ldots, S_\kappa$, where the sender $p_1$ holds input $b$, all honest parties terminate in at most $|\mathcal{P}_{\mathcal{E}}| - 1$ rounds.

[9] By abuse of notation, parties $p_0$ and $p_{\kappa+1}$ (which are not defined in $\mathcal{P}$) are treated as inactive.

**Claim 6.2.** *In a random execution $\mathcal{E}$ of scenario $S_{\kappa-1}$, party $p_\kappa$ outputs $b$ with overwhelming probability.*

*Proof.* In scenario $S_1$, parties $p_1$ and $p_2$ are honest; by validity, both output $b$ except for negligible probability. Next, for every $i \in \{2, \ldots, \kappa-1\}$ the view of party $p_i$ is identically distributed between scenarios $S_i$ and $S_{i-1}$, so $p_i$'s output distribution is identical across both scenarios. By consistency, in Scenario $S_i$ honest parties $p_i$ and $p_{i+1}$ output the same value except for negligible probability. By a standard hybrid argument, we conclude that in scenario $S_{\kappa-1}$, party $p_\kappa$ outputs $b$ except for negligible probability. □

**Claim 6.3.** *The output of $p_\kappa$ is identically distributed in scenarios $S_{\kappa-1}$ and $S_\kappa$.*

*Proof.* Let $\mathcal{E}_\kappa$ be a random execution of Scenario $S_\kappa$, where $|\mathcal{P}_{\mathcal{E}_\kappa}| = \kappa - 1$. By assumption, honest party $p_\kappa$ terminates in at most $|\mathcal{P}_{\mathcal{E}_\kappa}| - 1 = \kappa - 2$ rounds. In scenario $S_{\kappa-1}$, party $p_\kappa$ is $\kappa - 1$ hops from away from $p_1$. Thus, within the first $\kappa - 2$ rounds, the view of $p_\kappa$ is identically distributed in Scenarios $S_\kappa$ and $S_{\kappa-1}$. Hence, in scenario $S_{\kappa-1}$ party $p_\kappa$ also terminates in at most $\kappa - 2$ rounds, and its output distribution is identical across both scenarios. □

**Claim 6.4.** *In a random execution $\mathcal{E}$ of scenario $S_\kappa$, party $p_\kappa$ outputs $b$ with probability $1/2$.*

*Proof.* In Scenario $S_\kappa$ the sender $p_1$ is inactivate. Hence, the view of $p_\kappa$ is independent of $b$. □

Claims 6.2, 6.3, and 6.4 together yield a contradiction. This concludes the proof of Lemma 6.1. □

In the spirit of [GKKO07], we generalize this result to establish that any UP Broadcast protocol requires at least $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds when $|\mathcal{H}_\mathcal{E}| \geq 2$.

**Theorem 6.5.** *There exist a static environment $\mathcal{Z}$ and a static adversary $\mathcal{A}$ such that the following holds for every randomized UP Broadcast protocol $\pi$: In a random execution $\mathcal{E}$ with $\mathcal{Z}$ and $\mathcal{A}$, in which $\mathcal{Z}$ activates party-set $\mathcal{P}_\mathcal{E}$ and $\mathcal{A}$ corrupts all but a subset $\mathcal{H}_\mathcal{E}$, the honest parties in $\mathcal{H}_\mathcal{E}$ must run for at least $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds, except for negligible probability.*

*Proof.* Let $\pi$ be a UP Broadcast protocol with sender $p_1$. Let $f_1$ and $f_2$ be two polynomially bounded functions satisfying $f_2(\kappa) < f_1(\kappa)$ for every $\kappa \in \mathbb{N}$ (without loss of generality, assume that $f_2(\kappa)$ is even and divides $2f_1(\kappa)$). Assume that for every static environment $\mathcal{Z}$ that activates $n = f_1(\kappa)$ parties and for every static adversary $\mathcal{A}$ that corrupts all but $h = f_2(\kappa)$ parties, in every execution $\mathcal{E}$ the protocol $\pi$ terminates in less than $2n/h$ rounds.

We construct another UP Broadcast protocol $\pi_1$ from $\pi$. Let $\mathcal{Z}_1$ be a static environment activating the set of parties $\mathcal{Q} = \{q_1, \ldots, q_{n_1}\}$. Each party $q_i$ in $\pi_1$ computes $n = f_1(\kappa)$ and $h = f_2(\kappa)$ and emulates in its heed $h/2$ virtual parties in $\pi$, denoted $p_{(i-1)+1}, \ldots, p_{(i-1)+h/2}$. The sender $q_1$ sets the input bit of the virtual sender $p_1$ according to its own input. Whenever a party $q_i$ emulates a virtual party $p_j$ sending a message $m$ to virtual party $p_k$ in round $r$, party $q_i$ sends the message $(j, k, r, m)$ to the party emulating $p_k$ (and emulates it in its head if $q_i$ emulates $p_k$). Once all virtual parties emulated by $q_i$ terminate, $q_i$ outputs the value of the $p_{(i-1)+1}$ and terminates.

If $\pi$ is secure against static adversaries that corrupt up to $h = f_2(\kappa)$ parties, then $\pi_1$ is secure against static adversaries corrupting up to 2 parties. The virtual execution of $\pi$ contains $n = n_1 \cdot h/2$ parties, all but two are corrupted, so there remain at least $2 \cdot h/2 = h$ uncorrupted parties in $\pi$. By assumption, the round number in which $\pi$ terminates is less than

$$\frac{2n}{h} = \frac{2n_1 \cdot h/2}{h} = n_1.$$

Hence, $\pi_1$ executes with $n_1$ active parties (out of which all but two are corrupted) and terminates in fewer than $n_1$ rounds. This contradicts Lemma 6.1. □

## 6.2 Lower Bounds for APSA and UPIC

We now prove that the $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ lower bound also applies to APSA and UPIC protocols. The round-preserving reductions presented below establish a hierarchy among the core primitives in the UP model, illustrated in Figure 5 (where the arrows are pointing to easier problems from harder ones.)

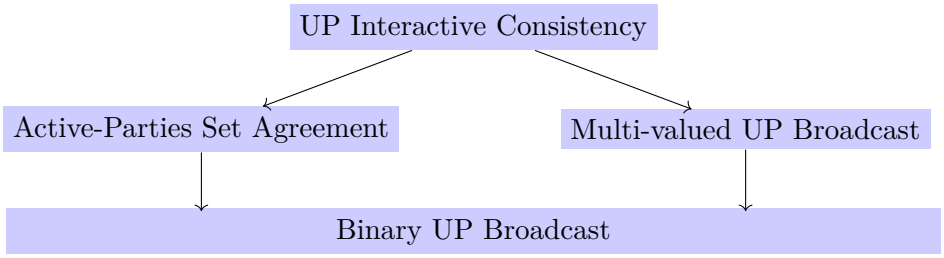



Figure 5: Hierarchy of problems in the UP model.

While the reduction from (multi-valued) UP Broadcast to UPIC is immediate, the binary version of UP Broadcast can also be directly reduced to APSA.

**Corollary 6.6.** *There does not exist any randomized APSA protocol that terminates (with overwhelming probability) in fewer than $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds in every execution $\mathcal{E}$.*

*Proof.* Assume for contradiction that there exists an APSA protocol $\pi$ that terminates with overwhelming probability in fewer than $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds. We construct a UP Broadcast protocol $\pi_{\mathsf{bc\text{-}apsa}}$ as shown in Figure 6.

**Protocol** $\pi_{\mathsf{bc\text{-}apsa}}$

**Setup Parameters:** The protocol is parameterized by the identity of the sender $s$, and by an APSA protocol $\pi$.

**Active-Parties Set Agreement:** Every active party $p \in \mathcal{U} \setminus \{s\}$ starts running $\pi$; if the sender $s$ is active and has input $b_s = 1$, then $s$ participates honestly in $\pi$, whereas if $b_s = 0$, the sender $s$ remains silent throughout the execution.

**Output:** Upon receiving the output $S_p$ from $\pi$, every active party $p \in \mathcal{U}$ outputs 1 if $s \in S_p$, and otherwise outputs 0.

Figure 6: Reduction from Binary UP Broadcast to APSA.

Notice that if $s$ is honest and has input value $b_s = 1$, it actively participates in $\pi$. By validity and consistency of APSA every honest party includes $s$ in its output set with overwhelming probability, and thus outputs 1. If $s$ is honest and has input value $b_s = 0$, then it behaves as inactive; by validity of APSA, $s$ in not included in the output of any honest party with overwhelming probability, and all honest parties outputs 0. Finally, consistency and termination in at most $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds follow directly from the corresponding properties of APSA. Hence, $\pi_{\mathsf{bc\text{-}apsa}}$ constitutes a secure UP Broadcast protocol that terminates in fewer than $\lfloor 2|\mathcal{P}_\mathcal{E}|/|\mathcal{H}_\mathcal{E}| \rfloor$ rounds, contradicting Theorem 6.5. □

The reduction from UP Broadcast to UPIC is established in the security proof of our UP Broadcast protocol in Section 5.

**Corollary 6.7.** *There does not exist any UPIC protocol terminating (with overwhelming probability) in fewer than $\lfloor 2|\mathcal{P}_{\mathcal{E}}|/|\mathcal{H}_{\mathcal{E}}|\rfloor$ rounds in every execution $\mathcal{E}$.*

*Proof.* Assume for contradiction that there exists a UPIC protocol $\pi$ that terminates with overwhelming probability in fewer than $\lfloor 2|\mathcal{P}_{\mathcal{E}}|/|\mathcal{H}_{\mathcal{E}}|\rfloor$ rounds. Then Protocol $\pi_{\mathsf{bc}}$ (Figure 3) would yield a UP Broadcast protocol terminating in fewer than $\lfloor 2|\mathcal{P}_{\mathcal{E}}|/|\mathcal{H}_{\mathcal{E}}|\rfloor$ rounds with overwhelming probability in every execution $\mathcal{E}$, contradicting Theorem 6.5. □

**Tightness of our Protocols in the UP Model.** Having established APSA, UP Broadcast, and UPIC all require at least $\lfloor 2|\mathcal{P}_{\mathcal{E}}|/|\mathcal{H}_{\mathcal{E}}|\rfloor$ rounds in the worst-case, we conclude that the protocols that presented in Sections 3 and 4 achieve optimal worst-case round complexity. Recall that all our UP protocols require at most $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds in every execution $\mathcal{E}$, matching the round complexity indicated by the lower bounds for the worst-case scenario that $|\mathcal{H}_{\mathcal{E}}^{R_T}| = 2$.

# 7 Late-Joining Parties

So far, the protocols constructed in the UP model guarantee security exclusively for honest parties activated at the beginning of the execution. A natural question is whether we can provide meaningful security guarantees for parties activated *after* the beginning of the execution. Our primary goal is to preserve the security guarantees established in previous sections with respect to the parties in the initial round $\mathcal{P}_{\mathcal{E}}^1$, while extending output guarantees to late-joining parties $\mathcal{P}_{\mathcal{E}}^r$ for $r > 1$.

We first formalize the model of UP Broadcast with late-joining receivers (Definition 7.1) and prove that this definition cannot be satisfied by any deterministic protocol (Theorem 7.2). We then consider a relaxed variant where late receivers may selectively abort, and provide a protocol with optimal worst-case round complexity (Theorem 7.4).

Similarly, for the case of a late-joining sender, we formalize UP Broadcast with a late-joining sender (Definition 7.5) and show a corresponding impossibility result for deterministic protocols (Theorem 7.6). To circumvent this limitation, we investigate broadcast within the *interacting preprocessing (IP)* model, where active parties can communicate with other prior to receiving the sender's input (Theorem 7.7). establish a round-complexity lower bound for IP Broadcast (Theorem 7.9), which directly implies a corresponding lower bound for UP Broadcast with a late-joining sender (Theorem 7.10).

## 7.1 Impossibility of UP Broadcast with Late-Joining Receivers

We analyze a strong variant of UP Broadcast with late-joining receivers, in which all parties that are honest in round $r$ and terminate in round $r$ must agree on the same output as parties activate in the first round and remain honest until throughout their execution. Ideally, late receivers should obtain this guarantee regardless of when they join, provided no party in $\mathcal{P}_{\mathcal{E}}^1$ has terminated uncorrupted before their activation.

**Definition 7.1** (UP Broadcast with Late-Joining Receivers)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$. A protocol $\pi$ is an adaptively secure, late-joining receivers UP Broadcast protocol with sender $s$, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$*

*satisfies the following: Let $r^*$ be the last round such that no party in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r^*}_{\mathcal{E}}$ has terminated by round $r^*$, then:*

- ***Validity:*** *Let $r$ be a round number, and let $p \in \mathcal{P}^{r^*}_{\mathcal{E}} \cap \mathcal{H}^{r}_{\mathcal{E}}$ be a party who terminates in round $r$ with output $m$. Then $m \in \{0,1\}^\ell \cup \{\perp\}$, where $m = m_s$ if $s \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r}_{\mathcal{E}}$, and $m = \perp$ if $s \notin \mathcal{P}^{r}_{\mathcal{E}}$.*
- ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}^{r^*}_{\mathcal{E}} \cap \mathcal{H}^{r_p}_{\mathcal{E}}$ and $q \in \mathcal{P}^{r^*}_{\mathcal{E}} \cap \mathcal{H}^{r_q}_{\mathcal{E}}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $m_p$ and $m_q$, respectively. Then $m_p = m_q$.*
- ***Termination:*** *Every party in $\mathcal{P}^{r^*}_{\mathcal{E}}$ either terminate in polynomially (in $\kappa$) many rounds, or get corrupted before terminating.*

Notice that $r^* \geq 1$; thus, the requirements for parties in $\mathcal{P}^1_{\mathcal{E}}$ that terminate uncorrupted are identical to those in standard UP Broadcast (Definition 2.2). Furthermore, these same guarantees extend to any party that joins before any of the parties in $\mathcal{P}^1_{\mathcal{E}}$ terminate uncorrupted.

We show that without placing additional bounds on the arrival times of late receivers, no deterministic protocol can achieve Definition 7.1. Specifically, if a late-joining receiver joins in round $r^*$ and terminates uncorrupted, no deterministic protocol can satisfy all three properties simultaneously.

**Lemma 7.2.** *There does not exist any deterministic late-joining receivers broadcast protocol $\pi$ in the UP setting that satisfies Definition 7.1.*

*Proof.* Assume for contradiction that there exists a deterministic late-joining receivers UP Broadcast protocol $\pi$. Let $s$ denote the sender, and let $p_1$ and $p_2$ be two distinct potential receivers. For messages $m, m' \in \{0,1\}^\ell$, consider the following executions of $\pi$:

- $\mathcal{E}_{1,m}$: Environment $\mathcal{Z}_1$ activates $\{s, p_1\}$ at the beginning with sender input $m$. Adversary $\mathcal{A}_1$ corrupts no parties. Let $R_{1,m}$ denote the round in which $p_1$ terminates.
- $\mathcal{E}_{2,m'}$: Environment $\mathcal{Z}_2$ activates $\{s, p_2\}$ at the beginning with sender input $m'$. Adversary $\mathcal{A}_2$ corrupts no parties. Let $R_{2,m'}$ denote the round in which $p_2$ terminates.

Since every execution terminates in at most $R_{\max} = \mathsf{poly}(\kappa)$ rounds, for any message length $\ell > \log_2(R_{\max})$, by the Pigeonhole Principle there exist distinct messages $m \neq m'$ such that $R_{1,m} = R_{2,m'}$; denote this round by $R$. Fix such a pair $(m, m')$ and denote $\mathcal{E}_1 = \mathcal{E}_{1,m}$ and $\mathcal{E}_2 = \mathcal{E}_{2,m'}$. By validity of $\pi$, party $p_1$ outputs $m$ in $\mathcal{E}_1$ and party $p_2$ outputs $m'$ in $\mathcal{E}_2$, both terminating in round $R$.

Fix an arbitrary party $q \in \mathcal{U} \setminus \{s, p_1, p_2\}$. Consider the following executions $\mathcal{E}_3$ and $\mathcal{E}_4$, defined under adversary-environment pairs $(\mathcal{Z}_3, \mathcal{A}_3)$ and $(\mathcal{Z}_4, \mathcal{A}_4)$, respectively:

- $\mathcal{E}_3$: Environment $\mathcal{Z}_3$ activates $\{s, p_1, p_2, q\}$ at the beginning. Adversary $\mathcal{A}_3$ immediately corrupts $\{s, p_2, q\}$ and instructs the sender $s$ to behave honestly towards $p_1$ as if its input were $m$. Simultaneously, $\mathcal{A}_3$ internally emulates an execution between $s$ and $p_2$ where $s$ holds input $m'$, dropping all messages between $p_1$ and $\{p_2, s\}$. Finally, $\mathcal{A}_3$ instructs $q$ to remain silent until round $R-2$, and then behave as a late-joining receiver that joins in round $R-1$, except that $q$ sends no messages to $p_1$.
- $\mathcal{E}_4$: Environment $\mathcal{Z}_4$ activates $\{s, p_1, p_2, q\}$ at the beginning. Adversary $\mathcal{A}_4$ immediately corrupts $\{s, p_1, q\}$ and instructs the sender $s$ to behave honestly towards $p_2$ as if its input were $m'$. Simultaneously, $\mathcal{A}_4$ internally emulates an execution between $s$ and $p_1$ where $s$ holds input $m$,

dropping all messages between $p_2$ and $\{p_1, s\}$. Finally, $\mathcal{A}_4$ instructs $q$ to remain silent until round $R-2$, and then behave as a late-joining receiver that joins in round $R-1$, except that $q$ sends no messages to $p_2$.

Party $p_1$ cannot distinguish between $\mathcal{E}_1$ and $\mathcal{E}_3$ by round $R$, and thus terminates in round $R$ with output $m$ in $\mathcal{E}_3$. Analogously, $p_2$ cannot distinguish between $\mathcal{E}_2$ and $\mathcal{E}_4$ by round $R$, terminating in round $R$ with output $m'$ in $\mathcal{E}_4$.

Now, consider executions $\mathcal{E}_5$ and $\mathcal{E}_6$, defined under $(\mathcal{Z}_5, \mathcal{A}_5)$ and $(\mathcal{Z}_6, \mathcal{A}_6)$, respectively:

- $\mathcal{E}_5$: Environment $\mathcal{Z}_5$ activates $\{s, p_1, p_2\}$ at the beginning. Adversary $\mathcal{A}_5$ immediately corrupts $\{s, p_2\}$, instructing $s$ to behave honestly towards $p_1$ with input $m$, while internally emulating $\{s, p_2\}$ with input $m'$ and dropping all messages between $p_1$ and $p_2$. Environment $\mathcal{Z}_5$ activates $q$ in round $R-1$, and $\mathcal{A}_5$ corrupts $p_1$ in round $R$, forcing it to remain silent for all rounds $r \geq R$.
- $\mathcal{E}_6$: Environment $\mathcal{Z}_6$ activates $\{s, p_1, p_2\}$ at the beginning. Adversary $\mathcal{A}_6$ immediately corrupts $\{s, p_1\}$, instructing $s$ to behave honestly towards $p_2$ with input $m'$, while internally emulating $\{s, p_1\}$ with input $m$ and dropping all messages between $p_1$ and $p_2$. Environment $\mathcal{Z}_6$ activates $q$ in round $R-1$, and $\mathcal{A}_6$ corrupts $p_2$ in round $R$, forcing it to remain silent for all rounds $r \geq R$.

In $\mathcal{E}_5$, the view of party $q$ is identically distributed as its view in $\mathcal{E}_3$ up to its output determination, so by consistency with $p_1$, party $q$ outputs $m$ in $\mathcal{E}_5$. Furthermore, party $q$ cannot distinguish between execution $\mathcal{E}_5$ and execution $\mathcal{E}_6$, which implies $q$ must also output $m$ in $\mathcal{E}_6$. However, in $\mathcal{E}_6$, honest party $p_2$ outputs $m'$, so $q$'s output of $m$ violates consistency with $p_2$, yielding a contradiction. □

### 7.2 UP Broadcast with Selective Abort for Late-Joining Receivers

We proceed to present a weaker notion of UP Broadcast, which enables late-joining receivers to abort in case they identify that an attack is taking place. Not-triviality is captured by requiring that if all parties are honest they must output the sender's input as long as they join before all parties that were active in the first round terminate. Borrowing terminology from the MPC realm [GL05, IKP10], we call this variant UP Broadcast with *selective abort*. We note that requiring security for late-joining receivers *only* when all active parties are honest, resembles *weak broadcast* [Lam83].

**Definition 7.3** (UP Broadcast with Selective Abort for Late-Joining Receivers)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$. A protocol $\pi$ is an adaptively secure, late-joining receivers UP Broadcast protocol with selective abort with sender $s$, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$ satisfies the following: Let $r^*$ be the last round such that no party in $\mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^{r^*}_\mathcal{E}$ has terminated by round $r^*$, then:*

- ***Validity:*** *Let $r$ be a round number, and let $p \in \mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^r_\mathcal{E}$ be a party who terminates in round $r$ with output $m$. Then $m \in \{0,1\}^\ell \cup \{\perp\}$, where $m = m_s$ if $s \in \mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^r_\mathcal{E}$, and $m = \perp$ if $s \notin \mathcal{P}^r_\mathcal{E}$.*
- ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^{r_p}_\mathcal{E}$ and $q \in \mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^{r_q}_\mathcal{E}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $m_p$ and $m_q$, respectively. Then $m_p = m_q$.*
- ***Late-joining consistency:*** *Let $r_p$ and $r_q$ be round numbers and let $p \in \mathcal{P}^1_\mathcal{E} \cap \mathcal{H}^{r_p}_\mathcal{E}$ and $q \in \mathcal{P}^{r^*}_\mathcal{E} \cap \mathcal{H}^{r_q}_\mathcal{E}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $m_p$ and $m_q$, respectively. Then, $m_q \in \{m_p, \perp\}$ and furthermore, if $\mathcal{P}^{r_q}_\mathcal{E} = \mathcal{H}^{r_q}_\mathcal{E}$, then $m_q = m_p$.*

– ***Termination:*** *Every party in $\mathcal{P}_{\mathcal{E}}^{r^*}$ either terminates in polynomially (in $\kappa$) many rounds or gets corrupted before terminating.*

Notice that the properties of *validity* and *consistency* are identical to those in our primary UP Broadcast definition. *Late-joining consistency* guarantees that if a late receiver joins before any of the parties in $\mathcal{P}_{\mathcal{E}}^1$ (that remain honest) has terminated, then it will either output the same value as the parties in $\mathcal{P}_{\mathcal{E}}^1$ that terminate uncorrupted, or output $\perp$.

**UP Broadcast with selective abort.** Our starting point is the UPIC protocol from Section 4. In this protocol, if all active parties remain honest throughout the execution, no communication is required after round 3: all parties send their initial messages in round 1 and forward the received signatures alongside their own in round 2. Thus, to accommodate late-joining receivers, parties activated from the beginning simply send their local set $A_p^r$ (defined in Figure 2) in every round $r$.

If a late-joining party receives $|A|$ identical copies of a set $A$, it can temporarily assume that the agreed-upon set is $A$. If no other message, revealing the presence of an active corruption is received within $|A|$ rounds from the beginning of the execution, the late-joining party can safely terminate.

Under this approach, late-joining parties remain silent and do not disrupt the ongoing UPIC instance. Consequently, if no corruptions occur, all parties terminate simultaneously in $|A| = |\mathcal{P}_{\mathcal{E}}^1|$ rounds. The only exception occurs when $|\mathcal{P}_{\mathcal{E}}^1| = 1$. In this edge case, the adversary could confuse the single party $p \in \mathcal{P}_{\mathcal{E}}^1$ and the late-joining parties by delivering a new value to $p$ just before $p$ terminates while withholding it from late-joiner, causing $p$ and the late-joiners to diverge in subsequent rounds. We circumvent this edge case by adopting the standard assumption that $|\mathcal{P}_{\mathcal{E}}^1| \geq 2$, as consistency is otherwise trivial . Our selective-abort protocol is described in Figure 7.

**Protocol** $\pi_{\mathsf{bc\text{-}sa}}$

**Setup Parameters:** The protocol is parameterized by the identity of the sender $s$ and by the UPIC protocol $\pi_{\mathsf{ic}}$ (Figure 2).

**UPIC:** In round 1, every active party $p \in \mathcal{U} \setminus \{s\}$ executes an instance of $\pi_{\mathsf{ic}}$ with input $\perp$, while $s$ participates with input $m_s$. In every round $r \geq 1$, every active party $p \in \mathcal{U}$ appends $A_p^r$ to the message it sends at the end of $r$.

**Selective Abort:** Let $p \in \mathcal{U}$ be a party activated in round $r > 1$. If $p$ receives a message $M$ in round $r+1$ that is not a set of party-value pairs, or if it receives two distinct sets $A \neq A'$, it outputs $\mathsf{abort}$.

Otherwise, let $A$ be the unique set received by $p$ in round $r+1$. If $p$ receives a message other than $A$ prior to or during round $|A|+1$, it outputs $\mathsf{abort}$.

Otherwise, $p$ terminates in round $|A|+1$, outputting $m$ if $(s, m) \in A$ for an $m \in \{0,1\}^\ell$, and $\perp$ otherwise.

Figure 7: UP Broadcast protocol with Selective Abort for Late-Joining Receivers.

Note that if the protocol terminates in round $|A|+1$, it consumes exactly $|A|$ rounds of communication. We now establish that this construction achieves UP Broadcast with selective abort for late-joining receivers.

**Theorem 7.4.** *Assuming the existence of EUF-CMA signatures, Protocol $\pi_{\mathsf{bc\text{-}sa}}(\pi_{\mathsf{ic}})$ (Figure 7) is a UP broadcast protocol with selective abort for late-joining receivers (according to Definition 7.3) in the CA-model.*

*Moreover, for every execution $\mathcal{E}$ of $\pi_{\mathsf{bc\text{-}sa}}(\pi_{\mathsf{ic}})$, letting $r^*$ be the last round such that no party in $\mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r^*}_{\mathcal{E}}$ has yet terminated, then all parties in $\mathcal{P}^{r^*}_{\mathcal{E}}$ that terminate uncorrupted, do so in at most $|\mathcal{P}^{R_T}_{\mathcal{E}}|$ rounds from the beginning of the execution.*

*Proof.* First, observe that the execution of the underlying $\pi_{\mathsf{ic}}$ instance is completely independent of the messages exchanged for late-joining receivers; hence, validity and consistency are preserved.

Consider an execution $\mathcal{E}$ where $\mathcal{H}^{|\mathcal{P}^1_{\mathcal{E}}|+1}_{\mathcal{E}} = \mathcal{P}^{|\mathcal{P}^1_{\mathcal{E}}|+1}_{\mathcal{E}}$. By the properties of $\pi_{\mathsf{ic}}$, all parties in $\mathcal{P}^1_{\mathcal{E}}$ terminate simultaneously in round $|\mathcal{P}^1_{\mathcal{E}}| + 1$. Moreover, every party $p \in \mathcal{P}^1_{\mathcal{E}}$ observes $A^r_p = A$ in every round $r \geq 2$. Thus, $A$ is the sole message received by any party $q \in \mathcal{H}^{|\mathcal{P}^1_{\mathcal{E}}|+1}_{\mathcal{E}}$ up to round $|\mathcal{P}^1_{\mathcal{E}}| + 1$, ensuring that $p$ and $q$ yield identical outputs.

Next, suppose for contradiction that there exist rounds $r_p$ and $r_q$ and uncorrupted parties $p \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r_p}_{\mathcal{E}}$ and $q \in \mathcal{P}^{r^*}_{\mathcal{E}} \cap \mathcal{H}^{r_q}_{\mathcal{E}}$ that terminate in rounds $r_p$ and $r_q$ with distinct outputs $m_p \neq m_q$, respectively. For $q$ to terminate with output $m_q$, it must have received a single set $A$ containing $(s, m_q)$ and no conflicting messages. This implies that $p$ sent $A$ in some round $r < r_p$. Because $p \in \mathcal{H}^{r_p}_{\mathcal{E}}$ outputs $m_p \neq m_q$, $p$ must have accepted a different pair $(s, m')$ in a round $r' > r$. We analyze two cases based on $r'$:

- If $r' \geq |A| + 1$, then $p$ observed $|A| - 1$ valid signatures on the newly accepted pair from parties represented in $A$. At least one such signature must originate from an honest party $w \in \mathcal{P}^1_{\mathcal{E}} \cap \mathcal{H}^{r'}_{\mathcal{E}} \setminus \{p\}$. Consequently, $q$ also receives this signature batch in round $|A| + 1$ prior to terminating.
- If $r' \leq |A|$, then $p$ broadcasts a message batch for its newly accepted pair, which $q$ receives by round $r' + 1 \leq |A|$, strictly before $q$'s termination round.

In either case, $q$ observes a message conflicting with $A$ before its decision threshold, compelling $q$ to output $\mathsf{abort}$, which is a contradiction. Therefore, late-joining consistency holds.

Finally, termination in at most $|\mathcal{P}^{R_T}_{\mathcal{E}}|$ rounds follows directly from the termination properties of Protocol $\pi_{\mathsf{ic}}$. □

## 7.3 Security with a Late-Joining Sender

Similar to our treatment of late-joining receivers, we begin by establishing in Section 7.3.1 that achieving security for a late-joining sender is impossible in the general case. In Section 7.3.2 we describe a straightforward protocol for the setting where the sender is guaranteed sufficient time to distribute its message after joining, under the assumption that a commonly agreed-upon activation round exists. In Section 7.3.3 we introduce a relaxation of the UP model permitting offline communication prior to input delivery, and in Section 7.3.4 we prove that UP Broadcast with a late-joining sender remains impossible when available time is strictly bounded, even under this relaxed setup.

### 7.3.1 Impossibility of UP Broadcast with a Late-Joining Sender

In principle, a late-joining sender UP Broadcast protocol should maintain all standard UP Broadcast guarantees (as defined in Section 2.2) regardless of how late the sender joins the execution, provided that no uncorrupted party terminates prior to its arrival. We formalize this security notion below:

**Definition 7.5** (UP Broadcast with a Late-Joining Sender)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$. A protocol $\pi$ is an adaptively secure, late-joining sender UP Broadcast protocol with sender $s$, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with*

*overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$ in the presence of $\mathcal{Z}$ and $\mathcal{A}$ satisfies the following: Let $r^*$ be the last round such that no party in $\mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{r^*}$ has terminated by round $r^*$, then:*

- ***Validity:*** *Let $r$ be a round number, and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^r$ be a party who terminates in round $r$ with output $m$. Then $m \in \{0,1\}^\ell \cup \{\perp\}$, where $m = m_s$ if $s \in \mathcal{P}_{\mathcal{E}}^{r^*} \cap \mathcal{H}_{\mathcal{E}}^r$, and $m = \perp$ if $s \notin \mathcal{P}_{\mathcal{E}}^{r-1}$.*
- ***Consistency:*** *Let $r_p$ and $r_q$ be round numbers, and let $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{r_p}$ and $q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{r_q}$ be two parties that terminate in rounds $r_p$ and $r_q$ with outputs $m_p$ and $m_q$, respectively. Then $m_p = m_q$.*
- ***Termination:*** *Every active party either terminates in polynomially (in $\kappa$) many rounds or gets corrupted before terminating.*

We now prove that no deterministic protocol can satisfy Theorem 7.5.

**Lemma 7.6.** *There exists no deterministic late-joining sender UP Broadcast protocol $\pi$ in the UP setting satisfying Theorem 7.5.*

*Proof.* Assume toward contradiction that there exists a deterministic late-joining sender UP Broadcast protocol $\pi$. Let $s = p_1$ with input $m_s$, fix $n \geq 2$, and let $G = \{p_2, \ldots, p_{n+1}\}$. Consider the following executions $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ of $\pi$, operating with $(\mathcal{Z}_1, \mathcal{A}_1)$, $(\mathcal{Z}_2, \mathcal{A}_2)$, and $(\mathcal{Z}_3, \mathcal{A}_3)$, respectively.

- $\mathcal{E}_1$: $\mathcal{Z}_1$ activates $G$ at round 1 while keeping $s$ inactive. $\mathcal{A}_1$ corrupts no parties. Let $r$ is the first round in which some party $p \in G$ terminates.
- $\mathcal{E}_2$: $\mathcal{Z}_2$ activates $G$ at round 1 and activates $s$ in round $r-1$. $\mathcal{A}_2$ corrupts only $p$ in round $r$, instructing it to remain silent thereafter.
- $\mathcal{E}_3$: $\mathcal{Z}_3$ activates $G$ at round 1 and activates $s$ in round $r-1$. $\mathcal{A}_3$ corrupts only $s$ in round $r-1$, instructing it to behave honestly towards the parties in $G \setminus \{p\}$ while not sending anything to $p$.

In $\mathcal{E}_1$, by validity, $p$ terminates with output $\perp$ in round $r$. Party $p$ cannot distinguish between $\mathcal{E}_1$ and $\mathcal{E}_3$ up to round $r$. Hence, $p$ must also terminate in round $r$ of $\mathcal{E}_3$ with output $\perp$.

Furthermore, parties in $G \setminus \{p\}$ cannot distinguish between $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ by round $r-1$. Thus, $s$ is activated in $\mathcal{E}_2$ and $\mathcal{E}_3$ before any party terminates uncorrupted in either of the two executions. By consistency of $\pi$, all parties in $G \setminus \{p\}$ must output $\perp$ in $\mathcal{E}_3$ to match $p$'s output. However, by validity, parties in $G \setminus \{p\}$ must output $m_s$ in $\mathcal{E}_2$. Because parties in $G \setminus \{p\}$ cannot distinguish between $\mathcal{E}_2$ and $\mathcal{E}_3$, they must yield identical outputs in both executions, yielding a contradiction. □

### 7.3.2 UP Broadcast with a Late-Joining Sender and Enough Time

Having established that no protocol can remain secure when the sender joins just one round prior to the uncorrupted termination of initial parties, a natural follow-up question arises: can UP Broadcast be achieved imposing an additional restriction on the joining time of the sender (apart from no party having terminated uncorrupted)?

Specifically, if we assume a known upper bound $r$ how late the sender may join (provided no initial honest party has terminated), a trivial solution exists: all parties remain idle for the first $r$ rounds, after which they execute a standard UP Broadcast protocol. This simple construction terminates in $r + |\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds, ensuring all parties in $\mathcal{P}_{\mathcal{E}}^1$ that remain honest reach consistent outputs.

This feasibility observation relies on two essential conditions:

1. A round $r \leq R$ is known by which the sender is guaranteed to be active.

2. The protocol is allowed to execute for at least $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ additional rounds following the sender's arrival.

If either condition is violated, UP Broadcast becomes impossible. If the parties do not know a priori when the sender will join, our previous lower bound (Theorem 7.6) rules out feasibility. Conversely, if the guaranteed activation round of the sender is known, but the protocol is forced to terminate in strictly fewer than $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds following that activation, a distinct barrier arises.

To illuminate the fundamental reasons underlying this second barrier, we prove that $\left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$ serves as a lower bound on the remaining round complexity, even under a significantly strengthened model. Specifically, we analyze the *interactive-preprocessing model*, where active parties are permitted to engage in arbitrary message exchanges prior to the sender receiving its input and initiating the agreement phase. We show that any lower bound established for this interactive-preprocessing model directly implies a corresponding lower bound for late-joining sender UP Broadcast protocols.

### 7.3.3 UP Broadcast with Interactive Preprocessing

The *interactive-preprocessing (IP)* model constitutes a relaxation of the UP setting described in Section 2.1. The key distinction in the IP setting is that the sender is not assigned its input value at the beginning of the execution. Instead, the environment $\mathcal{Z}$ can activate parties dynamically, allowing them to communicate freely over an arbitrary number of rounds before the actual broadcast protocol begins. At a designated round, the sender $s$ is given its input value, and all active parties are notified. This setup phase enables currently active honest parties to exchange identity information and establish coordination before any information regarding the sender's input is revealed.

Since we introduce this model primarily to establish our lower bound, we assume without loss of generality that both $\mathcal{Z}$ and $\mathcal{A}$ cease activating and corrupting parties, respectively, by round $R_T$, which is also the round in which the sender receives its input. Under these conditions, the formal definition of broadcast in the IP model directly generalizes that of UP Broadcast:

**Definition 7.7** (IP Broadcast)**.** *Let $\kappa \in \mathbb{N}$ be a security parameter, let $\mathcal{U}$ be a (possibly exponential) set of parties, and let $s \in \mathcal{U}$. A protocol $\pi$ is an adaptively secure, IP Broadcast protocol with sender $s$, if for every dynamic PPT environment $\mathcal{Z}$, every adaptive PPT adversary $\mathcal{A}$, and every input message $m_s \in \{0,1\}^\ell$ with $\ell = O(\kappa)$, with overwhelming probability, an execution $\mathcal{E}$ of $\pi$ with sender input $m_s$, where $\mathcal{Z}$ and $\mathcal{A}$ terminate by round $R_T$, has the following properties:*

- ***Validity:*** *Every party $p \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{R_T}$ terminates with output $m_p \in \{0,1\}^\ell \cup \{\perp\}$, where $m_p = m_s$ if $s \in \mathcal{H}_{\mathcal{E}}^{R_T}$ and $m_p = \perp$ if $s \notin \mathcal{P}_{\mathcal{E}}^{R_T}$.*

- ***Consistency:*** *For all parties $p, q \in \mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{R_T}$ that output $m_p$ and $m_q$, respectively, it holds that $m_p = m_q$.*

- ***Termination:*** *Every party in $\mathcal{P}_{\mathcal{E}}^1$ either terminates in polynomially (in $\kappa$) many rounds, or gets corrupted before terminating.*

We now prove that any protocol satisfying IP Broadcast requires at least $\left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$ additional rounds after the input is delivered to the sender. This argument generalizes the corresponding lower bound established for the UP model in Section 6.

**Lemma 7.8.** *There exist an adaptive environment $\mathcal{Z}$ and an adaptive adversary $\mathcal{A}$ such that the following holds for every IP Broadcast protocol $\pi$: In a random execution $\mathcal{E}$ under $\mathcal{Z}$ and $\mathcal{A}$, where $\mathcal{Z}$ activates a set of parties $\mathcal{P}_{\mathcal{E}}^{R_T}$ by round $R_T$ and $\mathcal{A}$ corrupts all but two parties, at least one honest party in $\mathcal{H}_{\mathcal{E}}^{R_T}$ must execute for at least $|\mathcal{P}_{\mathcal{E}}^{R_T}|$ rounds after $R_T$, except with negligible probability (where the probability is over the coins of the environment, the adversary, and the honest parties).*

*Proof.* Let $\pi$ be an IP Broadcast protocol and, without loss of generality, assume the sender is $p_1$. Given security parameter $\kappa$, consider the set of parties $\mathcal{P} = \{p_1, \ldots, p_\kappa\}$, and define two static environments:

- $\mathcal{Z}_1$ activates parties $\mathcal{P} \setminus \{p_1\}$ at round 1 and activates $p_1$ right before terminating in round $R_T$.
- $\mathcal{Z}_2$ activates parties $\mathcal{P} \setminus \{p_1\}$ at round 1 and terminates in round $R_T$ without activating $p_1$.

Consider the sequence of $\kappa$ adversaries $\mathcal{A}_1, \ldots, \mathcal{A}_\kappa$ and corresponding scenarios $S_1, \ldots, S_\kappa$ constructed as in the proof of Theorem 6.1. This induces execution configurations identical to those in Theorem 6.1.

Let $b \leftarrow \{0,1\}$ be a uniformly distributed bit delivered to sender $p_1$ in round $R_T$. Assuming for contradiction that all honest parties terminate within $|\mathcal{P}_{\mathcal{E}}^{R_T}| - 1$ rounds following $R_T$ across all scenarios $S_1, \ldots, S_\kappa$, then the exact indistinguishability arguments from Theorem 6.1 apply (substituting $\mathcal{P}_{\mathcal{E}}^{R_T}$ for $\mathcal{P}_{\mathcal{E}}$). Consequently, Claims 6.2, 6.3 and 6.4 hold, yielding a contradiction. □

**Theorem 7.9.** *There exist an dynamic environment $\mathcal{Z}$ and an adaptive adversary $\mathcal{A}$ such that the following holds for every IP Broadcast protocol $\pi$: In a random execution $\mathcal{E}$ under $\mathcal{Z}$ and $\mathcal{A}$, where $\mathcal{Z}$ activates party set $\mathcal{P}_{\mathcal{E}}^{R_T}$ by round $R_T$ and $\mathcal{A}$ corrupts all but an honest subset $\mathcal{H}_{\mathcal{E}}^{R_T}$, the honest parties in $\mathcal{H}_{\mathcal{E}}^{R_T}$ must run for at least $\left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$ rounds following $R_T$, except with negligible probability.*

*Proof.* Follows by an argument identical to the proof of Theorem 6.5, replacing the initial party set $\mathcal{P}_{\mathcal{E}}$ with the active set $\mathcal{P}_{\mathcal{E}}^{R_T}$ at round $R_T$. □

### 7.3.4 Late-Joining Sender UP Broadcast Lower Bound

By a straightforward reduction from IP Broadcast to late-joining sender UP Broadcast, we establish the matching lower bound for the latter setting:

**Lemma 7.10.** *Let $\pi$ be a late-joining sender UP Broadcast protocol (Theorem 7.5). There exists an execution $\mathcal{E}$ of $\pi$ where the environment activates sender $s$ in round $r \leq R_T$ (prior to any initial party terminating uncorrupted) such that uncorrupted parties in $\mathcal{P}_{\mathcal{E}}^1$ cannot all terminate before round $r + \left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$.*

*Proof.* Assume for contradiction that there exists such a late-joining sender UP Broadcast protocol $\pi_{\mathsf{ls}}$. We can construct an IP Broadcast protocol $\pi_{\mathsf{ip}}(\pi_{\mathsf{ls}})$ by executing $\pi_{\mathsf{ls}}$ where $s$ participates as a late-joining sender.

In the IP setting, both $\mathcal{Z}$ and $\mathcal{A}$ terminate in round $R_T$, which is also when sender $s$ receives its input (corresponding to joining round $r = R_T$ in $\pi_{\mathsf{ls}}$). By our assumption on $\pi_{\mathsf{ls}}$, all uncorrupted parties in $\mathcal{P}_{\mathcal{E}}^1$ would terminate strictly before round $R_T + \left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$.

Furthermore, by the validity and consistency properties of $\pi_{\mathsf{ls}}$, if $s \in \mathcal{H}_{\mathcal{E}}^{R_T}$, all uncorrupted parties in $\mathcal{P}_{\mathcal{E}}^1 \cap \mathcal{H}_{\mathcal{E}}^{R_T}$ output $m_s$, and all such parties output identical values regardless of $s$'s honesty. Consequently, $\pi_{\mathsf{ip}}$ constitutes an IP Broadcast protocol that terminates in fewer than $\left\lfloor 2|\mathcal{P}_{\mathcal{E}}^{R_T}|/|\mathcal{H}_{\mathcal{E}}^{R_T}| \right\rfloor$ rounds following $R_T$, contradicting Theorem 7.9. □

## References


[ABdSFG08] Eduardo Adílio Pelinson Alchieri, Alysson Neves Bessani, Joni da Silva Fraga, and Fabíola Greve. Byzantine consensus with unknown participants. In Theodore P. Baker, Alain Bui, and Sébastien Tixeuil, editors, *Proceedings of the 12th International Conference Principles of Distributed Systems OPODIS 2008*, volume 5401 of *Lecture Notes in Computer Science*, pages 22–40. Springer, 2008.

[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. ACM, July / August 2019.

[AD15a] Ittai Abraham and Danny Dolev. Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 605–614. ACM Press, June 2015.

[AD15b] Marcin Andrychowicz and Stefan Dziembowski. PoW-based distributed cryptography with no trusted setup. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 379–399. Springer, Berlin, Heidelberg, August 2015.

[ADD+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected $O(1)$ rounds, expected $O(n^2)$ communication, and optimal resilience. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 320–334. Springer, Cham, February 2019.

[BDDS92] Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: Changing algorithms on the fly to expedite byzantine agreement. *Inf. Comput.*, 97(2):205–233, 1992.

[Ben83] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *2nd ACM PODC*, pages 27–30. ACM, August 1983.

[BGIK16] Amos Beimel, Ariel Gabizon, Yuval Ishai, and Eyal Kushilevitz. Distribution design. In Madhu Sudan, editor, *ITCS 2016*, pages 81–92. ACM, January 2016.

[BGK+18] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, October 2018.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *30th FOCS*, pages 410–415. IEEE Computer Society Press, October / November 1989.

[BGP92] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Optimal early stopping in distributed consensus (extended abstract). In Adrian Segall and Shmuel Zaks, editors, *Distributed Algorithms, 6th International Workshop, WDAG '92, Haifa, Israel, November 2-4, 1992, Proceedings*, volume 647 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 1992.

[BIK17] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. Ad hoc PSM protocols: Secure computation without coordination. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 580–608. Springer, Cham, April / May 2017.

[BMTZ17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356. Springer, Cham, August 2017.

[Bor96] Malte Borderding. Levels of authentication in distributed agreement. In *10th International Workshop on Distributed Algorithms WDAG*, pages 40–55, 1996.

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

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[Can04] Ran Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop, (CSFW)*, pages 219–233. IEEE Computer Society, 2004.

[Can20] Ran Canetti. Universally composable security. *Journal of the ACM*, 67(5):28:1–28:94, 2020.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Berlin, Heidelberg, August 2015.

[CDL+25] Ran Cohen, Jack Doerner, Eysa Lee, Anna Lysyanskaya, and Lawrence Roy. An unstoppable ideal functionality for signatures and a modular analysis of the dolev-strong broadcast. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 675–708. Springer, Cham, December 2025.

[CDM25] Ran Cohen, Poulami Das, and Tal Moran. Setup-free committee sampling with subquadratic communication. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part II*, volume 16246 of *LNCS*, pages 273–304. Springer, Singapore, December 2025.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

[CGZ23] Ran Cohen, Juan A. Garay, and Vassilis Zikas. Completeness theorems for adaptively secure broadcast. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 3–38. Springer, Cham, August 2023.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CPS20] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 246–265. Springer, Cham, May 2020.

[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. Springer, Cham, May 2025.

[DPS19] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 23–41. Springer, Cham, February 2019.

[DRS90] Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM*, 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.

[Fit02] Matthias Fitzi. *Generalized communication and security models in Byzantine agreement*. PhD thesis, ETH Zurich, 2002.

[FLM86] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Comput.*, 1(1):26–39, 1986.

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

[FN09] Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. In Idit Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 449–463. Springer, 2009.

[GK20] Juan A. Garay and Aggelos Kiayias. SoK: A consensus taxonomy in the blockchain era. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 284–318. Springer, Cham, February 2020.

[GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, October 2007.

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

[GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 291–323. Springer, Cham, August 2017.

[GKL24] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 71(4):25:1–25:49, 2024.

[GKO+20] Juan A. Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. Resource-restricted cryptography: Revisiting MPC bounds in the proof-of-work era. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 129–158. Springer, Cham, May 2020.

[GKZ10] Juan A. Garay, Aggelos Kiayias, and Hong-Sheng Zhou. A framework for the sound specification of cryptographic tasks. In Andrew Myers and Michael Backes, editors, *CSF 2010 Computer Security Foundations Symposium*, pages 277–289. IEEE Computer Society Press, 2010.

[GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005.

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

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, Berlin, Heidelberg, August 2010.

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

[KW21] Pankaj Khanchandani and Roger Wattenhofer. Byzantine agreement with unknown participants and failures. In *35th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2021*, pages 952–961. IEEE, 2021.

[Lam83] Leslie Lamport. The weak byzantine generals problem. *Journal of the ACM*, 30(3):668–676, 1983.

[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. Springer, Cham, May 2024.

[LR23a] Andrew Lewis-Pye and Tim Roughgarden. Byzantine generals in the permissionless setting. In Foteini Baldimtsi and Christian Cachin, editors, *The 27th International Conference on Financial Cryptography and Data Security FC 2023, Part I*, volume 13950 of *Lecture Notes in Computer Science*, pages 21–37. Springer, 2023.

[LR23b] Andrew Lewis-Pye and Tim Roughgarden. Permissionless consensus. *CoRR*, abs/2304.14701, 2023.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[MMR23] Dahlia Malkhi, Atsuki Momose, and Ling Ren. Towards practical sleepy BFT. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 490–503. ACM Press, November 2023.

[MR22] Atsuki Momose and Ling Ren. Constant latency in sleepy consensus. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2295–2308. ACM Press, November 2022.

[Per85] Kenneth J. Perry. Early stopping protocols for fault-tolerant distributed agreement. Technical report, Cornell University, USA, 1985.

[PS17] Rafael Pass and Elaine Shi. The sleepy model of consensus. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 380–409. Springer, Cham, December 2017.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[PSs17] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Cham, April / May 2017.

[Rab83] Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409. IEEE Computer Society Press, November 1983.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

[WXDS20] Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi. Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 412–456. Springer, Cham, November 2020.

[WXSD20] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 381–411. Springer, Cham, November 2020.