Byzantine Consensus in the Partially Authenticated Setting

Christoph Lenzen
christophlen@ru.is
Aalto University
Espoo, Finland
Reykjavik University
Reykjavík, Iceland

Julian Loss
lossjulian@gmail.com
Ruhr University Bochum
Bochum, Germany

Kecheng Shi
kcshi97@gmail.com
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
Saarland University
Saarbrücken, Germany

Benedikt Wagner
benedikt.wagner@ethereum.org
Ethereum Foundation
Zug, Switzerland

Abstract

Byzantine Agreement and Broadcast are traditionally studied in one of two extremes: the authenticated setting, where a public key infrastructure (PKI) enables universally verifiable signatures and yields higher fault tolerance, and the unauthenticated setting, where no PKI is available and resilience necessarily drops. Motivated by Proof-of-Stake blockchains, where only a stable subset of participants (e.g., validators) have registered long-term keys while others do not, we initiate a systematic study of consensus in the *partially authenticated* setting, where a subset of parties are *registered* in a PKI and the remaining parties are *unregistered*.

We provide a nearly complete feasibility characterization of the resilience as a function of the number s of registered parties among n total parties. First, we show that Byzantine Agreement or Byzantine Broadcast with an *unregistered* sender is possible if and only if $t \leq \max\{\lceil s/2 \rceil, \lceil n/3 \rceil\} - 1$, matching a simple protocol and an impossibility bound. Second, for Byzantine Broadcast with a *registered* sender, we give a deterministic synchronous broadcast protocol tolerating up to $t \leq s + \lceil (n-s)/3 \rceil - 1$ Byzantine faults (equivalently, $3t < n + 2s$); while we present the binary case in the main body for clarity, our techniques extend to an efficient multivalued protocol. We complement this with a matching lower bound in a strengthened leakage model in which the adversary learns each party's private state at the end of every round, ruling out both deterministic protocols and randomized protocols that rely only on short-lived secrets and the basic signing/verification interface.

CCS Concepts

• **Theory of computation**;

Keywords

Byzantine broadcast, public key infrastructure, synchronous protocols

ACM Reference Format:
Christoph Lenzen, Julian Loss, Kecheng Shi, and Benedikt Wagner. 2026. Byzantine Consensus in the Partially Authenticated Setting. In *ACM Symposium on Principles of Distributed Computing (PODC '26), July 6–10, 2026, Egham, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3796701.3815929

1 Introduction

Byzantine Agreement and Broadcast [23] are fundamental problems in distributed computing that ask n parties to agree on a common and non-trivial output by running a joint protocol Π. Crucially, Π should remain secure and live even in the presence of up to $t < n$ *Byzantine* parties that may deviate from the protocol arbitrarily.

A long line of literature has studied protocols for the *synchronous model* in which all messages are delivered within a known upper bound Δ on network delay and protocols proceed in lock-step *rounds*. Existing work mainly considers two types of protocols:

Authenticated protocols. In the authenticated setting, parties share a pre-established public key infrastructure (PKI) that assigns to every party P_i a public key pk_i. Using its corresponding secret key sk_i, P_i can generate a signature that everyone can efficiently verify using pk_i. Under these conditions, it is well-known that deterministic synchronous Byzantine Agreement is efficiently solvable tolerating t corruptions if and only if $2t < n$ and deterministic synchronous Byzantine Broadcast is solvable for any $t < n$ corruptions [10, 20].

Unauthenticated protocols. In the unauthenticated setting, parties do not have access to a PKI and it is assumed only that parties are connected via pairwise authenticated channels. It is well-known that deterministic synchronous Byzantine Agreement and Broadcast are possible if and only if $3t < n$ [5, 20].

Partially Authenticated Protocols. A natural question is what resilience a protocol can achieve when a subset $\mathcal{S} \subseteq \mathcal{P}$ of *registered* parties holds a public key, whereas the remaining parties $\mathcal{N} := \mathcal{P} \setminus \mathcal{S}$ are *unregistered* parties and have no registered key. We refer to this setting as the *partially authenticated setting*. Indeed, this setting was

The full version of this paper [21] is available at https://eprint.iacr.org/2026/470.

PODC '26, Egham, United Kingdom

ACM ISBN 979-8-4007-2512-8/26/07
https://doi.org/10.1145/3796701.3815929

already explored in the pioneering work of Dolev and Strong [10], who showed that broadcast is possible if $|\mathcal{S}| \geq t+1$. It immediately follows that Byzantine Agreement is possible if $|\mathcal{S}| \geq t+1$ and $2t < n$, and the latter condition is trivially necessary. Protocols that assume such a *partially authenticated PKI* have recently come into focus due to practical motivation from the blockchain space. While dynamic participation models capture the temporal fluctuations of permissionless networks, they often abstract away the cryptographic asymmetry inherent in Proof-of-Stake blockchains such as, e.g., the Ethereum blockchain. Such systems typically distinguish two types of nodes: 1) *validator nodes* who participate in proposing and voting on blocks and hold *registered* long-term public keys[1] and 2) *non-validator nodes* who may join and leave freely without *registering* a public key. This shows that in reality, participation is not just about being "online"; it is also about which parties have keys and which parties do not. In this work, we give a nearly complete characterization of Byzantine Agreement and Byzantine Broadcast in the partially authenticated setting with $s = |\mathcal{S}|$ by showing the following results.

- Byzantine Agreement or Byzantine Broadcast with an *unregistered sender* $P_S \in \mathcal{N}$ is possible if and only if $t \leq \max\{\lceil s/2 \rceil, \lceil n/3 \rceil\} - 1$. Our upper bound is simple (see section 5): if the maximum is attained by $\lceil n/3 \rceil$, we can run a standard protocol ignoring the PKI. Otherwise, registered parties in $\mathcal{S}$ decide their output by running a consensus protocol among themselves. They then send their output to unregistered parties in $\mathcal{N}$ who decide their output by taking a majority vote on the messages they received from registered parties in $\mathcal{S}$. Our lower bound shows that this is the tight protocol for this setting.
- We then turn our attention to the much more interesting and challenging problem of Byzantine Broadcast with a registered sender $P_S \in \mathcal{S}$. On the positive side, we give a broadcast protocol for a registered sender $P_S \in \mathcal{S}$ that is secure against up to $t \leq s + \lceil (n-s)/3 \rceil - 1$ Byzantine corruptions. We complement this result with a matching lower bound for a setting where the private state of a party is leaked to the adversary at the end of each round r (including random coins sampled in round r). Since long-term keys do not exist in this model, signatures are instead provided exclusively through a suitable idealized functionality $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}$. Our bound rules out deterministic protocols as well as randomized R-round ones that (i) succeed with probability $1 - o(1/R)$ and (ii) rely only on short-term secrets and the basic properties of a signature scheme, i.e., the ability to sign a message and verify a signature. Interestingly, this probability threshold is asymptotically tight: we show that a simple protocol achieves success probability $1 - O(1/R)$ based on a shared coin.

1.1 Technical Overview

As already explained above, we focus here exclusively on the interesting case of Byzantine Broadcast where the sender P_S is registered, i.e., $P_S \in \mathcal{S}$. We first present a simplified version of our protocol for the case of exactly $t = s$ many corruptions, then explain how it can be extended to $t \leq s + \lceil (n-s)/3 \rceil - 1$ corruptions (as well as the special cases $t \in \{s+1, s+2\}$).

Recap: The Dolev-Strong Protocol. Our starting point is the classic Dolev-Strong authenticated broadcast protocol [10], which is secure against up to t Byzantine corruptions, where $t < n$. This deterministic broadcast protocol has $t+1$ rounds and it tolerates any $t < n$ Byzantine corruptions. In each round $k \leq t+1$, each party extends (by signing it) and forwards any valid $(k-1)$-signature chain (or simply chain, for short) that it has received in round $k-1$, to all other parties. Here, a k-chain is defined to be valid in the view of a party P_i on a value v if the chain contains valid signatures of k different parties that do not include P_i and the first signer on the chain is the sender P_S. At the end of round $t+1$, each party either outputs the unique value corresponding to the (only) valid signature chain it received during the course of the protocol or it outputs a default value in case such a unique value/signature chain does not exist. Validity, i.e., that for honest sender only the sender's value is received, follows immediately from unforgeability of the underlying digital signature scheme. Consistency, i.e., identical output at honest parties, follows from the fact that for rounds $k < t+1$, a party can be sure that if it extends and forwards an accepted chain, then all other parties will either do the same in the next round, or they have already accepted it. If a party receives a $(t+1)$-chain in round $t+1$, then that chain must include at least one honest signer who would have forwarded it to all other parties in the previous round t.

The main obstacle in transferring this protocol to our setting is that if an unregistered party $P_i \in \mathcal{N}$ receives and accepts a k-chain in some round k, then it cannot extend it so that other parties will accept the extended chain in the next round. As already observed by Dolev and Strong, this issue is easy to overcome as long as $s \geq t+1$. In this case, the protocol can proceed in $t+1$ super-rounds which each consist of two actual rounds (referred to as *phases* below). During the first phase of super round k, an unregistered party $P_i \in \mathcal{N}$ receiving a signature chain of length k accepts it without reservation. It then forwards the chain to all registered parties in $\mathcal{S}$ in the second phase of this super round. Hence, registered parties in $\mathcal{S}$ can receive and accept chains in the second phase of a super round k and extend them in the same manner as they would in the regular Dolev-Strong protocol. This allows to carry over the same reasoning for validity and consistency to this protocol as well. Unfortunately, this approach crucially relies on there being at least one honest registered party $P_j \in \mathcal{S}$ to extend any chain that an unregistered party $P_i \in \mathcal{N}$ accepts during the protocol. Thus, for $|\mathcal{S}| \leq t$, a new insight is required.

A protocol for $s = t$. We start by adding a third phase to each super round. Moreover, we will require a total of only $s = t$ super rounds as opposed to $t+1$ of them, as described above. We structure each super round k as follows. In the first phase, registered parties in $\mathcal{S}$ extend and forward any chain that they accepted in the previous super round (they can accept a chain during any phase of a super round). In the second phase, unregistered parties in $\mathcal{N}$ accept and echo any k-chain received in the first phase from a registered party. Finally, in the third phase, unregistered parties accept any k-chain that they were forwarded by another unregistered party during the previous phase and once again forward it to registered parties. But an unregistered party ignores k-chains it receives during this phase of super round k. For super rounds $k < t$, the logic is relatively

[1] This is usually achieved by validators staking funds.

straight forward. If a registered party P_i accepts a chain, every other party will accept it in the next super round $k+1$ upon P_i extending and forwarding it during phase 1 of that super round.

If an unregistered party P_i accepts a k-chain in phase 2, then every other unregistered party P_j accepts it in phase 3 of super round k. Now, for super round t, our key insight is the following: if an unregistered party P_j accepts a t-chain in phase 3 of super round t from another unregistered party P_i, it can be certain that all parties on this chain are corrupt, as otherwise, P_j would have received this chain in phase 1 of super round t. But this implies that all unregistered parties must be honest (as there are exactly $t = s$ corruptions). Hence, P_i is honest and would have forwarded the t-chain to all other unregistered parties as well, who will reach the same conclusion. Therefore, it is safe for P_j to accept this chain from P_i.

A protocol for $t = s + \lceil (n-s)/3 \rceil - 1$ (or $t = n$ if $t \leq s+2$). We observe that the above protocol can be viewed as a specific instantiation of a more general protocol blueprint. Namely, we can view the decision taken by unregistered parties in phase 3 of super round t as an instance of consensus on whether or not a specific chain should be accepted in this super round. We focus here on the binary case; hence assume that we simply run two separate agreement routines for each bit among the unregistered parties. It is clear that if all s registered parties are corrupted, then at most $\lceil (n-s)/3 \rceil - 1$ are corrupted among unregistered parties. This would allow unregistered parties to reach a decision on whether or not to accept a specific chain via an unauthenticated agreement protocol Π with security for up to $\lceil (n-s)/3 \rceil - 1$ many parties. Unfortunately, unregistered parties cannot know whether $\mathcal{S}$ is fully corrupted. This poses an additional issue compared to our simple protocol from above, since no guarantee on Π's behaviour can be made in case $\lceil (n-s)/3 \rceil$ or more parties are corrupted. For this reason, our construction follows a more modular and nuanced design pattern. We first build a special *gradecast* protocol in which parties additionally output a *grade* $g \in \{0,1\}$ to indicate their confidence in their output v. If an honest party outputs v with $g = 1$, then all honest parties output v (although possibly with grade $g = 0$). In addition to the common definition of validity, graded consistency and termination [11], our protocol has a special property: it guarantees that if *any* registered party is honest, then all honest parties will output a value v with the high-confidence grade $g = 1$. Our routine follows the 3-phase super round pattern from above, but introduces some additional protocol logic to set the grades correctly. We stress that this turns out to be particularly subtle for the multivalued case in the full version of the paper [21] and constitutes the main technical challenge of our protocol.

With this gradecast in place, unregistered parties can conclude the protocol by inputting their output v from the graded consensus routine to a single instance of the unauthenticated agreement protocol Π. A party that outputs v with grade $g = 1$ in gradecast simply ignores the output of Π and outputs v as the final output of the broadcast protocol. However, all parties participate in Π so as to help parties who output in gradecast with grade 0. These low-confidence parties will then decide their output of the broadcast protocol as the output of Π. By the above discussion, if there exists at least one honest registered party, all unregistered parties ignore the output of Π and decide on a consistent output for the broadcast protocol. On the other hand, if the set of registered parties is fully corrupt, then the parties in $\mathcal{N}$ can successfully run Π. Since a single honest party with grade $g = 1$ guarantees that all parties input the same value v to Π in this case, validity of Π guarantees that all honest registered parties output consistently.

Note that this implies that t must be such that the Π has resilience $t - s$. This results in the classic resilience of strictly less than $|\mathcal{N}|/3$—except when $|\mathcal{N}| < 3$. While of course a single party can trivially solve consensus, the case of $|\mathcal{N}| = 2$ breaks the pattern due to a subtlety. Since the inputs to Π are signed by the sender, *any* signed value can be verified to be a feasible output, even a party did not receive it as input to Π. Thus, the two parties in $\mathcal{N}$ may exchange their inputs, if there is a unique value signed by the sender output it, and otherwise output a default value. We build a broadcast protocol in the section 6 that solves this special case.

1.2 Lower Bounds

We complete our study of the partially authenticated setting with two matching lower bounds. For the case of byzantine agreement or broadcast with an unregistered sender, our lower bound follows, more or less, the standard template of impossibility proofs in this area. To illustrate the idea for the case of broadcast, our proof can leverage that the corrupted receivers can simulate the behavior of the sender with two different input bits (as it does not have a key).

Unfortunately, this strategy fails completely once the sender is registered. To illustrate our technical contribution, we describe the difficulty our proof with two strawman approaches that do not work (or only lead to a suboptimal lower bound). As our final proof mainly goes through the case of $n = 4$, $s = 1$, and $t = 2$, which it then generalizes to all other settings, our discussion below will mainly focus on this case as well. We will assume a protocol Π that solves broadcast for this setting and try to arrive at a contradiction.

Strawman 1: As we have explained, it is not possible for a corrupted party to simulate an honest sender's behaviour on two different input bits during an execution where the sender is actually honest. However, parties need to output consistently in the protocol even if the sender has crashed. For the sake of simplicity, suppose that Π is deterministic and parties output 0 after R rounds of not hearing anything from the sender. Thus, a more promising proof strategy might be to attempt to start from an execution in which the sender P_S is honest and has input 1. Two of the other parties, P_2 and P_3, are corrupted and feign, to the remaining honest party P_1, an execution in which the sender has crashed. We then move to a second execution in which the sender is dishonest, P_1 and P_2 are honest, and P_3 is corrupted. It is easy for P_S and P_3 to behave in such a way that P_1 cannot tell these first two executions apart, and so P_1 must output 1 in this second execution as well. Hence, also P_2 outputs 1 in execution 2. By a symmetrical argument, we can extend this strategy to a third execution in which P_3 and P_2 are honest so as to force P_3 to output 1 as well. However, to arrive at a contradiction, we would need an argument for why it is not possible that in this execution 1 is the output. Certainly, as the sender pretends to crash immediately, there is no way for P_3 to conclude that 1 is a valid output? Unfortunately, in order to facilitate the deception, the sender must provide signatures to the dishonest parties, so that they may "fool" P_j, $j \in \{1,2,3\}$ into not seeing

the difference between execution j and $j+1$. As the protocol may require the sender to sign the proposed output value 1, this overly optimistic approach fails.

Strawman 2: Having illustrated the main difficulty of this scenario, we now move to a proof strategy that is significantly closer to our actual one, except that it does not cover the case of $t \leq s < \lfloor n/4 \rfloor$. We have illustrated above why it is not possible for a corrupt receiver to simulate an honest execution on behalf of P_S with input 1 when P_S has, in actuality, benignly crashed at the beginning of the execution (and it is thus impossible to discern its input). So our key idea is instead to incrementally delay the receipt of a message in the view of an honest receiver that would be consistent with such an execution in which P_S is honest with input 1 to a later and later point in the protocol. Our strategy can achieve this while keeping the view of one honest party consistent through any two neighbouring executions while eventually pushing the receipt of a signed message from P_S all the way back past round R. Once this has happened, we can easily switch to an execution in which P_S has actually crashed, since any party who has not received a message from P_S by round R would have had to terminate with output 0 already. Our strategy to achieve this works as follows: in the first execution, the sender is honest, whereas parties P_2 and P_3 are corrupt and feign an execution where they do not receive any messages from P_S through the first three rounds of the protocol. In the next execution, P_1 and P_2 are honest and P_S and P_3 create a view that is consistent with the first execution for P_1. Note that now, P_2 first receives a message consistent with an honest 1-execution that has originated from P_S only at round 2 (relayed through P_1). This implies that in the next step, an honest receiver P_3 will first receive such a message from P_S by round 3. By keeping P_S corrupted and rotating the remaining corruption through one of the three receivers, we can successively delay the receipt of a message that was sent from P_S to a later and later point in the protocol, as explained above.

Restrictions of Strawman 2: To make the above work, we have assumed that the protocol is deterministic and that parties can simulate the behaviour of honest protocol parties. This is possible only if parties do not keep long-term secrets on which their executions implicitly depend. As such, we restrict the scope of our proof strategy to protocols in which registered parties can only sign through an idealized functionality $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}$ that simultaneously allows unregistered parties to verify signatures (but not to sign them). In addition, we will assume that the adversary learns the internal state of all honest parties at the end of each round. This restricts protocol parties to rely only on short term keys and randomness. Note, however, that this still covers some very strong assumptions, like a random oracle providing unpredictable, shared randomness to all parties at the beginning of each round.

There is an additional wrinkle in this simplified strategy: it does not cover the regime where $s < \lfloor n/4 \rfloor$. To see why, we will explain the issue with translating the bound for the four party setting to the n-party setting when $s < \lfloor n/4 \rfloor$. Similar issues occur when attempting a direct proof of a bound for the n-party setting, given that $s < \lfloor n/4 \rfloor$. We assume for simplicity that all fractions in this informal discussion are integers.

The conventional template for this proof strategy is to turn an n-party protocol Π' into a 4-party protocol Π by having each of the 4 protocol parties in Π simulate $n/4$ parties in Π'. This, however, presents an immediate obstacle in the partially authenticated setting: which parties in Π' should each of the registered parties running Π simulate? Consider the following idea for the case of $s = n/4$: P_S in Π simulates P_S in Π'. All other registered parties in $\mathcal{S}$ (which are also simulated by P_S) are crashed in Π'. The remaining unregistered parties that make up $\mathcal{N}$ are divided up equally and simulated by the remaining parties $P_1, P_2, P_3 \in \mathcal{N}$ that are also running Π with P_S. Now, suppose that $s < n/4$. Hence, each of the receiver parties $P_i \in \mathcal{N}$ has to simulate $(n-s)/3 > n/4 > s$ many parties in Π'. But this poses an issue: if e.g. P_2 and P_3 become corrupted in Π (as was indeed the case for the first execution in our attack strategy described above), then there are $2(n-s)/3 > s + (n-s)/3$ many corrupted parties in Π'. But Π' is only proven secure against at most $t < s + (n-s)/3$ many corruptions. Similar issues arise for different strategies of the parties in Π simulating the parties in Π'.

Our Final Proof Strategy. Our final proof strategy follows a similar idea as above. The main technical difference is that we will start from an execution in which the sender P_S has crashed and then gradually delay the point of its crash further and further into the protocol. Meanwhile, we carefully restore the messages that an honest sender P_S would send on input 1 for those rounds where P_S is now once again sending. This argument turns out to be very delicate and requires, in some cases, to restore messages only to some of the recipients (rather than to all of them at once) and the specific round during which the sender crashes. This makes the approach work when corrupting P_S and at most one other party P_j. This covers the entire parameter regime after translating to the general case.

Failure Probability. We note that our results show that this particular broadcast problem is structurally different. In contrast to the full-PKI or no-PKI scenarios, a sender with signing capability *can* successfully broadcast with probability arbitrarily close to 1, against any number of corruptions. This comes at the cost of a large number of rounds, however: our lower bound yields a failure probability of $\Omega(1/R)$ for any R-round protocol (subject to our model assumptions), and a simple protocol given in Section 4.3 succeeds with probability $1 - O(1/R)$.

1.3 Related Work

The Byzantine Agreement and Broadcast problems have been extensively studied in the literature, both in the computational setting [10] and in the unconditional setting [5, 14], as well as in the unconditional setting with pseudo-signatures [3, 24]. We note that Dolev and Strong [10] also considered the case where only $t+1$ parties sign and disseminate messages. They give a Byzantine Broadcast protocol in the partially authenticated setting with s registered parties and up to $t = s - 1$ Byzantine parties.

There are several partially authenticated models that have been considered in the past literature. Specifically, Borcherding [7, 8] considers the case where signatures are only used in some rounds. We note that their setting still requires each party to hold a key. Gordon et al. [16] and Gupta et al. [17] consider a related setting with a partially compromised PKI in which the adversary is able to obtain secret keys or forge signatures on behalf of up to t_c of the honest parties, while corrupting another t_a many parties. Their work

shows that broadcast is possible iff $2t_a + \min\{t_a, t_c\} < n$. Bansal et al. [2] extend the partially compromised PKI setting [16, 17] to arbitrary (undirected) graphs. A key difference compared to our setting is that in their setting, the set of parties with uncompromised keys is not publicly known, that is, any honest party's key may become exposed. As will become clear in our technical overview, our protocol crucially leverages that parties know which parties are registered and which parties are not.

Fitzi et al. [12] show that Broadcast (and multi-party computation) with unanimous abort is possible for $t < n$ corruptions in the unauthenticated setting, assuming only that secure signature schemes exist. In their protocol, parties generate keys for a signature scheme on the fly, and they use them to bootstrap a broadcast routine. We remark that their protocols would not be possible in our delayed public state model with $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}$. This is because only signature keys that are originally registered with this functionality (and are therefore part of the partial PKI) remain secure from the adversary over the execution of the protocol. Another example of a popular cryptographic tool that falls outside of our model are verifiable random functions (VRFs). For practical purposes, VRFs [22] can be seen as a special type of signature scheme in which signatures produced through the signing algorithm have a random, close to uniform distribution. This can be used, e.g., to sample small sub-committees or justify a particular (random) choice in a protocol to another party after the fact. VRFs have been used to great effect in randomized protocols such as Algorand [15], as well as the protocols of Abraham et al. [1] and Blum et al. [6] to achieve $o(n^2)$ communication complexity.

It would be interesting to explore whether one can circumvent our lower bound by leveraging such tools and techniques (or prove a more general lower bound that rules them out, too). More generally, it would be interesting to see if or how the picture changes if protocols rely on long-term secrets beyond signing keys. For example, a common technique in multi-party computation protocols is to have parties publicly commit to some secret value v through a cryptographic commitment and later open it (if necessary) [4, 18]. We believe these are interesting directions for future work.

2 Preliminaries

2.1 Model

Network Model. We consider a complete network of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and write $[n] = \{1, \ldots, n\}$ for the set of party indices. We work in the synchronous model where protocol proceeds in rounds; messages sent at the beginning of a round are guaranteed to be received by all receivers at the end of the round.

Cryptographic Setup. We consider a (bulletin) PKI setup for a subset of parties. Let $\mathcal{S} \subseteq \mathcal{P}$ denote the set of parties whose public keys are registered in the PKI and let $s := |\mathcal{S}|$. Let $\mathcal{N} := \mathcal{P} \setminus \mathcal{S}$ denote the set of unregistered parties. The registered parties in $\mathcal{S}$ have public/secret key pairs for an ideal digital signature scheme, and their corresponding public keys are known to all parties (while the secret keys are known only to the respective parties). We denote a signature of the party P_i on a message m by $\mathsf{sign}(m, i)$. For simplicity, we also refer to the tuple $(m, sign(m, i))$ as a signature on m of the party P_i.

Adversary Model. We consider an adaptive adversary that can choose up to t parties to corrupt at any time and get full control of the corrupted parties. The adversary is computationally bounded, meaning that it cannot break the security of the aforementioned signature schemes and forge signatures. We note that our lower bounds even rule out protocols in the presence of a static adversary, which makes the lower bounds even stronger.

2.2 Definitions

Definition 2.1 (Byzantine Agreement (BA)). Let Π be a protocol executed among parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ where each party $P_i \in \mathcal{P}$ starts with an input value v_i and outputs a value o_i upon termination. Π solves BA against an adversary corrupting up to t parties if it satisfies the following properties:

- **Termination:** Every honest party P_i terminates.
- **Agreement:** For any two honest parties $P_i, P_j \in \mathcal{P}$ with outputs o_i and o_j, respectively, it holds that $o_i = o_j$.
- **Validity:** If each honest party P_i starts with the same input value v, then every honest party outputs $o_i = v$.

Definition 2.2 (Byzantine Broadcast). Let Π be a protocol executed among parties $\mathcal{P} = \{P_1, \ldots, P_n\}$. Denote the designated sender by P_S. In a broadcast protocol, P_S starts with an input value v and each party $P_i \in \mathcal{P}$ outputs a value v_i upon terminating. Π solves broadcast against an adversary corrupting up to t parties if it satisfies the following properties:

- **Termination:** Every honest party terminates.
- **Consistency:** For any two honest parties $P_i, P_j \in \mathcal{P}$, it holds that $v_i = v_j$.
- **Validity:** If the sender P_S is honest with input v, then every honest party outputs $v_i = v$.

Definition 2.3 (Gradecast). Let Π be a protocol executed among parties $\mathcal{P} = \{P_1, \ldots, P_n\}$. Denote the designated sender by P_S. P_S starts with an input value v, and each party $P_i \in \mathcal{P}$ outputs a value v_i upon terminating. Π solves gradecast against an adversary corrupting up to t parties if it satisfies the following properties:

- **Termination:** Every honest party terminates.
- **Graded Consistency:** If some honest party $P_i \in \mathcal{P}$ outputs (v_i, g_i) with $g_i = 1$, it holds that $v_i = v_j$ for any honest party $P_j \in \mathcal{P}$ that outputs (v_j, g_j).
- **Validity:** If the sender P_S is honest and it starts with input v, then every honest party outputs (v_i, g_i) such that $v_i = v$ and $g_i = 1$.

3 Broadcast with a Registered Sender in the Partially Authenticated Setting

In this section, we present the main positive result of our work: a broadcast protocol with a registered sender $P_S \in \mathcal{S}$ and resilience $t = s + \lceil (n - s)/3 \rceil - 1$.

3.1 Gradecast in the Partially Authenticated Setting

This subsection introduces the main primitive behind our broadcast protocol: a (binary) *gradecast* protocol Π_{GC} in the partially authenticated setting described in Fig. 1. Informally, gradecast [11]

is a weakened form of broadcast in that each party additionally outputs a *grade* reflecting its confidence. Concretely, at the end of Π_{GC}, each honest party P_i outputs a pair (v_i, g_i), where $v_i \in \{0,1\}$ is its tentative decision for the sender's bit and $g_i \in \{0,1\}$ is a grade indicating whether P_i believes that agreement on v_i holds for all honest parties. Our validity and graded consistency matches the standard gradecast definition [19]. Our Π_{GC} protocol also achieves a special conditional grade-1 property, that if there exists an honest registered party, all honest parties output with grade 1; we will show this in Lemma 3.10. Crucially, Π_{GC} works for any number of corruptions. We first recall the signature chains. Similar as in the classic Dolev-Strong authenticated broadcast protocol [10],

Definition 3.1 (Signature Chains). A *k-signature chain* C_k (for $k > 0$) with sender P_S on value v is recursively defined as follows: a 1-signature chain on v is a pair $(v, \mathsf{sign}(v, S))$. For $k > 1$, a k-signature chain on v is a pair $(C_{k-1}, \mathsf{sign}(C_{k-1}, i))$, where C_{k-1} is a $(k-1)$-signature chain on v that does not contain a signature from P_i. A k-signature chain on a value v is *valid* for P_i if the following conditions are satisfied:

- C_k begins with a signature of P_S on the value v.
- All k signers are distinct.
- C_k does not contain a signature of P_i (this trivially holds for $P_i \in \mathcal{N}$).

We first prove several useful lemmas for our protocol Π_{GC} that tolerates t Byzantine parties with resilience $t < n$.

Lemma 3.2. *There is no registered party $P_i \in \mathcal{S}$ that sets $\mathrm{acc}_i^s = 1$ in the super round s.*

Proof. P_i only accepts an s-chain C_s if C_s contains signatures from s different parties and it does not contain the signature of P_i. Since there exist at most s parties that can sign on the chain, there exists no such C_s that P_i can accept in the super round s. □

Lemma 3.3. *If there is an honest registered party $P_i \in \mathcal{S}$ that outputs (v_i, g_i) such that $v_i = 1$, then $v_j = 1$ for any honest party P_j that outputs (v_j, g_j).*

Proof. Since each registered party $P_i \in \mathcal{S}$ outputs v_i only if it has accepted a k-chain in super round k, from Lemma 3.2, the accepted chain must be a k-chain for $k < s$. Therefore, P_i creates a valid $k+1$-chain C_{k+1} and sends $(C_{k+1}, \mathsf{Extension})$ to all parties in the phase 1 in super round $k+1$. Each party P_j with $v_j = 0$ sets $v_j = 1$ in the phase 2 of super round $k+1$. Furthermore, P_j never sets $v_j = 0$ once it has $v_j = 1$ according to the protocol, so P_j must output $v_j = 1$ at the end of the protocol Π_{GC}. □

Lemma 3.4. *If there is an honest registered party $P_i \in \mathcal{S}$ that outputs (v_i, g_i) such that $v_i = 0$, then $v_j = 0$ for any honest party P_j that outputs (v_j, g_j).*

Proof. Assume $P_i \in \mathcal{S}$ outputs (v_i, g_i) such that $v_i = 0$. This means it has not accepted any chain during any super round of the protocol. Toward a contradiction, let P_j be an honest party that sets $v_j = 1$. We consider two cases: first, if P_j is registered, then by Lemma 3.3, P_i must output $v_i = 1$ at the end of the protocol, contradicting to the assumption that $v_i = 0$. In the second case, we have P_j to be unregistered. Then, P_j sets $v_j = 1$ in super round k for some $k < s$. Since the k-chain C_k that P_j received cannot contain the signature of P_i (as we have assumed that it never accepts a chain in any super round), $k < s$. There are the following sub-cases:

- If $P_j \in \mathcal{N}$ sets $v_j = 1$ in phase 2 of super round k, P_i must receive the Echo_1 message containing C_k of P_j in phase 2 of super round k. P_i sets $v_i = 1$ and outputs (v_i, g_i) with $v_i = 1$ at the end of the protocol.
- If $P_j \in \mathcal{N}$ sets $v_j = 1$ in phase 3 of super round k, P_i must receive the chain either from an Echo_2 message or Echo_3 message from P_j in phase 3 of super round k. Upon receiving the k-chain C_k as part of one of those messages, P_i sets $v_i = 1$ and outputs (v_i, g_i) with $v_i = 1$ at the end of the protocol.

(Binary) Gradecast Protocol Π_{GC}

Input and Initialization. Denote the Sender $P_S \in \mathcal{S}$. Each party P_i sets its grade $g_i = 1$ and value $v_i = 0$ at the beginning of the protocol. Each registered party P_i initializes a flag $\mathrm{acc}_i^k = 0$ to indicate whether P_i has accepted a k-chain in super round k, for $k < s$. If $P_i = P_S$ and $b = 1$, set $v_i := 1$ and $\mathrm{acc}_i^0 = 1$ (treated as accepting a valid 0-chain).

For $k = 1, \ldots, s$, execute super round k as described below:

Phase 1: PKI Extension. Each registered party P_i: if $\mathrm{acc}_i^{k-1} = 1$ (which means P_i has accepted a valid $(k-1)$-chain that does not contain the signature of P_i in super round $k-1$), create k-chain $C_k = \mathsf{sign}(C_{k-1}, i)$ and send Extension message $(C_k, \mathsf{Extension})$ to all parties.

Phase 2: Echo from Non-PKI Parties.

- Each registered party P_i that received a valid k-chain C_k in an Extension message $(C_k, \mathsf{Extension})$ in Phase 1: if $v_i = 0$, set $v_i = 1$ and $\mathrm{acc}_i^k = 1$.
- Each unregistered party P_i that received a valid k-chain C_k in an Extension message $(C_k, \mathsf{Extension})$ in Phase 1:
  - if $v_i = 0$, set $v_i = 1$ and send Echo_1 message (C_k, Echo_1) to all parties.
  - if $v_i = 1$ and $g_i = 0$, set $g_i = 1$ and send Echo_1 message (C_k, Echo_1) to all parties.

Phase 3: Chain Acceptance from $\mathcal{N}$.

(1)
- Each registered party P_i: if $v_i = 0$ and P_i received a valid k-chain C_k in an Echo_1 message (C_k, Echo_1) in Phase 2 set $v_i = 1$ and $\mathrm{acc}_i^k = 1$.
- Each unregistered party P_i: if $v_i = 0$ and it received a valid k-chain in an Echo_1 message (C_k, Echo_1) in Phase 2, set $v_i = 1$, $g_i = 0$ and send (C_k, Echo_2) to all parties.

(2)
- Each registered party P_i: if $v_i = 0$ and it received a valid k-chain C_k in an Echo_2 message in the last round, set $\mathrm{acc}_i^k = 1$ and $v_i = 1$.
- Each unregistered party P_i: if it received a valid k-chain C_k in an Echo_2 message (C_k, Echo_2) in the last round and $v_i = 0$: set $g_i = 0$ and send Echo_3 message (C_k, Echo_3) to all registered parties.

At the end of Phase 3: For each registered party P_i: if it received a valid k-chain C_k in an Echo_3 message (C_k, Echo_3) in the last round and $v_i = 0$, set $\mathrm{acc}_i^k = 1$ and $v_i = 1$.

Output Rule. At the end of super round s, for each party P_i: output (v_i, g_i).

Figure 1: (Binary) Gradecast Protocol Π_{GC}

Clearly, both of these cases lead to a contradiction with P_i outputting $(v_i = 0, g_i)$. Therefore, in conclusion, each party P_j must output (v_i, g_i) such that $v_i = 0$ at the end of the protocol. □

Lemma 3.5. *If there is an honest registered party $P_i \in \mathcal{S}$ that outputs (v_i, g_i), each honest party P_j outputs (v_j, g_j) with $v_j = v_i$.*

Proof. The proof follows directly from Lemma 3.3 and Lemma 3.4. □

Lemma 3.6. *The Gradecast protocol Π_{GC} satisfies Validity.*

Proof. If the sender P_S is honest and it starts with input 1, each honest party P_i sets $v_i = 1$ in phase 2 of super round 1 upon receiving a 1-chain from P_S. Hence, all honest parties in $\mathcal{N}$ must output $(1, 1)$ at the end of the protocol. Each honest party $P_j \in \mathcal{S}$ receives the Extension message from the leader P_S in, they must set $v_i = 1$ in the phase 2 and output $(1, 1)$ at the end of the protocol. If the sender P_S is honest and it starts with input 0, we note that in our protocol, the leader does not accept any valid 0-chain and it will not set $v_S := 1$ if $b = 0$. As a result, no honest party accepts any chain during the protocol from the unforgeability of the ideal signature scheme. In conclusion, any honest party must output $(0, 1)$ at the end of the protocol. □

Lemma 3.7. *The protocol Π_{GC} satisfies Graded Consistency.*

Proof. Assume an honest party P_i outputs (v_i, g_i) such that $g_i = 1$. If there exists an honest party P_j (P_j could be P_i) $\in \mathcal{S}$, from Lemma 3.5, all honest parties must output the same value as v_j. So assume instead that all honest parties are in the set $\mathcal{N}$. There are following different cases:

- **Case 1.** $v_i = 1$**.** In this case, $g_i = 1$ only happens if P_i sets $v_i = 1$ in phase 2 in a super round k. P_i will send an Echo_1 message to all other parties in phase 2 of super round k. Thus, each party $P_j \in \mathcal{N}$ sets $v_j = 1$ in phase 3 of super round k and output (v_j, g_j) such that $v_j = 1$.
- **Case 2.** $v_i = 0$**.** In this case, $g_i = 1$ only happens if there is no honest party P_j that sets $v_j = 1$ in any super round $k < s$. Now consider the last super round s. If any honest party $P_j \in \mathcal{N}$ sets $v_j = 1$ in phase 2 or the first round of phase 3, P_j either sends an Echo_1 message or an Echo_2 message to P_i. Thus, P_i would set $v_i = 1$ in phase 3, which is a contradiction. As a result, each honest party P_j must output (v_j, g_j) such that $v_j = 0$ at the end of the protocol.

In conclusion, the protocol Π_{GC} satisfies Graded Consistency. □

Lemma 3.8. *The protocol Π_{GC} satisfies Termination.*

Proof. Each party P_i in the protocol Π_{GC} terminates and outputs (v_i, g_i) after s super rounds where each super rounds consist of 4 rounds. □

Theorem 3.9. *The protocol Π_{GC} is a Gradecast protocol.*

Proof. Directly from Lemma 3.8, Lemma 3.7 and Lemma 3.6. □

Moreover, our Gradecast protocol Π_{GC} has a specialized grade-1 property in the partially authenticated setting, that if there exists an honest registered party $P_i \in \mathcal{S}$, each honest party P_j outputs (v_j, g_j) with grade $g_j = 1$ at the end of the protocol Π_{GC}:

Lemma 3.10. *If there exists an honest registered party $P_i \in \mathcal{S}$ (including the leader P_S), each honest party P_h outputs (v_h, g_h) such that $g_h = 1$ at the end of the protocol Π_{GC}.*

Proof. Assume an honest party $P_i \in \mathcal{S}$ outputs (v_i, g_i) at the end of the protocol. Throughout the protocol, $P_i \in \mathcal{S}$ only outputs with grade $g_i = 1$. Now we distinguish two cases for the value v_i and an arbitrary honest party $P_j \in \mathcal{N}$:

- If $v_i = 1$, from Lemma 3.2, P_i only accepts a k-chain and sets $v_i = 1$ in super round $k < s$. Since P_i only accepts a chain which it isn't itself a part of, it can extend the chain by adding its own signature to it and send an Extension message $(C_{k+1}, \text{Extension})$ for the extended $(k+1)$-chain C_{k+1} in super round $k+1$. Thus, each honest party P_j sets $v_j = 1$ and it must have $g_j = 1$ in phase 2 of super round $k+1$.
- If $v_i = 0$, assume there is an honest party $P_j \in \mathcal{N}$ that outputs (v_j, g_j) with $g_j = 0$. Since $g_i = 1$, by graded consistency (Lemma 3.7), P_j must output $v_j = v_i = 0$. P_j only sets $g_j = 0$ in phase 3 of some super round $k \le s$, upon receiving a valid k-chain C_k in an Echo_2 message. If $k = s$, the chain must contain the signature of P_i. Thus $v_i = 1$, contradicting the assumption that $v_i = 0$. If $k < s$, P_j sends an Echo_3 message containing the k-chain C_k to every party in $\mathcal{S}$, including P_i. Therefore, P_i sets $v_i = 1$ at the end of Phase 3 of super round $k < s$, contradicting the assumption that $v_i = 0$. □

3.2 Broadcast in the Partially Authenticated Setting

We are now ready to introduce our broadcast protocol Π_{BC} that builds on top of Π_{GC}, assuming a deterministic unauthenticated BA protocol Π_{unau} (with inputs that come from the outputs of Π_{GC}) among $n - s$ unregistered parties that tolerates up to $\lceil (n-s)/3 \rceil - 1$ Byzantine parties (for example, the one in [5]). In particular, for each party P_i that outputs (v_i, g_i) in the Π_{GC} protocol with $g_i = 1$, P_i outputs v_i at the end of Π_{BC}. Each unregistered party $P_i \in \mathcal{N}$ inputs its value v_i to the deterministic unauthenticated BA protocol Π_{unau}, denote its output as m_i. If $g_i = 1$, P_i outputs v_i, if $g_i = 0$, P_i outputs m_i. We note that only unregistered parties may output with grade 0 from our description of the Π_{GC} protocol.

Lemma 3.11. *The protocol Π_{BC} satisfies Validity if $t \le s + \lceil (n - s)/3 \rceil - 1$.*

Proof. Assume the sender is honest and starts with input b. According to the validity of the protocol Π_{GC}, each honest party P_j must output $(v_j, g_j) = (b, 1)$ at the end of the protocol. According to the output rule of the protocol Π_{BC}, for each honest party $P_i \in \mathcal{S}$, P_i must output $v_i = b$. For each honest party $P_j \in \mathcal{N}$, we make a case distinction:

- **Case 1.** $b = 0$**.** Since $v_j = 0$ and $g_j = 1$, P_j must output 0 according to the output rule of Π_{BC}.
- **Case 2.** $b = 1$**.** We have $v_j = 1$ and $g_j = 1$, none of the two conditions in the output rule are satisfied, and the party P_j must outputs 1.

In conclusion, each honest party outputs b. □

Broadcast Protocol Π_{BC}

Input and Initialization. Denote the sender $P_S \in \mathcal{S}$. Each party P_i sets value $v_i = 0$ at the beginning of the protocol. If $P_i = P_S$ and $b = 1$, set $v_i \coloneqq 1$.

Gradecast. Each party P_i: participate in the Gradecast protocol Π_{GC} with sender P_S and its input b. Denote the output of party P_i to be (v_i, g_i).

Consensus of Non-PKI parties.

- Each unregistered party $P_i \in \mathcal{N}$: input v_i to the deterministic unauthenticated BA protocol Π_{unau}, denote the output of P_i to be m_i.
- Each registered party P_i in $\mathcal{S}$: do not participate in the deterministic unauthenticated BA and wait for it to terminate; we use a fixed-round unauthenticated BA (e.g., the one in [5], which takes $3(\lceil (n-s)/3 \rceil - 1) + 1$ rounds).

Output Rule.

- Each registered party $P_i \in \mathcal{S}$: output v_i.
- Each unregistered party $P_j \in \mathcal{N}$:
  - If $g_j = 1$, output v_j.
  - Otherwise, output m_j.

Figure 2: Byzantine Broadcast Protocol Π_{BC} in the partially authenticated setting

Lemma 3.12. *The protocol Π_{BC} satisfies Consistency if $t \le s + \lceil (n-s)/3 \rceil - 1$.*

Proof. We consider two cases:

- **Case 1. There exists an honest party $P_i \in \mathcal{S}$.** In this case, denote its output of Π_{GC} as (v_i, g_i). From special grade-1 property (Lemma 3.10), each honest party P_h outputs (v_h, g_h) such that $g_h = 1$ at the end of the protocol Π_{GC}. From the graded consistency, each honest party P_h must output $v_h = v_i$. According to the output rule, each party $P_x \in \mathcal{S}$ therefore outputs $v_x = v_i$ and outputs v_i, and each party P_j in $\mathcal{N}$ has $g_j = 1$, so it outputs v_i from our output rule. This means that each honest party P_j outputs v_i, finishing this case.
- **Case 2. All parties in $\mathcal{S}$ are Byzantine.** Since $t \le s + \lceil (n-s)/3 \rceil - 1$, we have $3(t-s) \le 3\lceil (n-s)/3 \rceil - 3 < n - s$. If all parties in $\mathcal{S}$ are Byzantine, then there are at most $t - s$ Byzantine parties in $\mathcal{N}$, which means that the deterministic unauthenticated BA among parties in $\mathcal{N}$ must be secure. As a result, all honest parties in the set $\mathcal{N}$ must have the same output m for the unauthenticated BA.
  - **Case 2a. There exists an honest party $P_j \in \mathcal{N}$ that outputs (v_j, g_j) with $g_j = 1$.** In this case, from the graded consistency (Lemma 3.7), each honest party P_h must output (v_h, g_h) in the Gradecast protocol Π_{GC} such that $v_h = v_j$. From the validity of the unauthenticated BA, all honest parties output the same value m at the end of Π_{BC}.
  - **Case 2b. All honest parties have grade 0.** In this case, their output value is fully determined by their outputs of the unauthenticated BA, according to the output rule. As we have already established that all honest parties have the same output m in the unauthenticated BA protocol, consistency follows in this case.

□

Theorem 3.13. *The protocol Π_{BC} is a Byzantine Broadcast protocol that tolerates $t \le s + \lceil (n-s)/3 \rceil - 1$ Byzantine parties, given the condition that s parties (including the sender) are registered.*

Interestingly, in the special case when $n - s < 3$, we can even tolerate $t \le s + 1$ corruptions. We state the result here and leave proofs as well as protocol Π_{BCS} constructions (Fig. 4) in our Appendix 6.

Theorem 3.14. *If $n - s < 3$, there exists a Byzantine Broadcast protocol that tolerates any number of corruptions, given the condition that the sender is registered.*

Theorem 3.15. *The communication complexity of the Binary Byzantine Broadcast protocol Π_{BC} in Figure. 2 is $O(s \cdot n^2) + O((n-s)^3)$.*

4 Lower Bounds

4.1 Lower bounds for BA in the Partially Authenticated Setting

We now show that the resilience bound for Byzantine Agreement

$$t \le \max\left\{\lceil s/2 \rceil, \lceil n/3 \rceil\right\} - 1$$

is indeed optimal. In particular, when $s = 0$, i.e., no PKI is available, it is well-known that $t \le \lceil n/3 \rceil - 1$ is the tight bound. When $s = n$, i.e., all parties are registered with PKI setup, it is also well-known that $t \le \lceil s/2 \rceil - 1$ is the tight bound. We first prove it is optimal for broadcast with respect to an unregistered sender (recall that in the Corollary 5.2, we show there exists a Broadcast protocol that tolerates up to $t \le \max\left\{\lceil s/2 \rceil, \lceil n/3 \rceil\right\} - 1$ Byzantine parties, if the sender is unregistered).

Theorem 4.1. *Assume there exist s registered parties and up to t parties may be Byzantine. Then there is no Byzantine Broadcast protocol if*

$$t \ge \max\left\{\lceil s/2 \rceil, \lceil n/3 \rceil\right\}.$$

Corollary 4.2. *Assume there exists s registered parties and t parties can be Byzantine. Then there is no Byzantine Agreement protocol if*

$$t \ge \max\left\{\lceil s/2 \rceil, \lceil n/3 \rceil\right\}.$$

4.2 Lower bounds for BC in the Partially Authenticated Setting with Public State

We now give a lower bound for broadcast protocols in the partially authenticated setting. The bound holds in a model in which registered parties can securely use digital signatures, but otherwise all state of parties is known to the adversary, and consequently using digital signatures and related cryptographic primitives does not help parties.

The lower bound works in the same model as discussed in Section 2, with the following two modifications:

Delayed Public State Model. The adversary can base its actions in round r on the full state of all parties at the end of the preceding round $r - 1$ (or the initial state, if $r = 1$). More precisely, at the end of each round, the full internal state[2] of each party is leaked to the adversary. Parties may generate and use any kind of private or

[2]When modeling parties as interactive Turing machines, the full internal state consists of all tapes, the position of the head, its state transition function, and its state register.

shared randomness. However, any randomness generated in round r is leaked to the adversary at the end of round r as well. Thus, the adversary must choose which messages corrupted parties send in round r without knowledge of the private randomness honest parties generate in round r, but gains complete information about round r once it ends.

Partial Signing Functionality. All parties (and the adversary) have access to a special ideal functionality $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}$ parameterized by the sets $\mathcal{S}, \mathcal{N}$ (see Section 2). The ideal functionality internally maintains an initially empty set Auth and offers the following interfaces:

- $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}.\texttt{Sign}(m)$, called by Party $i \in \mathcal{S}$: Set Auth := Auth $\cup$ $\{(i, m)\}$.
- $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}.\texttt{Verify}(i, m)$, called by Party $j \in \mathcal{S} \cup \mathcal{N}$: Return 1 if $(i, m) \in$ Auth. Otherwise, return 0.

Notably, unregistered parties in $\mathcal{N}$ cannot call $\mathcal{F}_{\text{sign}}^{\mathcal{S},\mathcal{N}}.\texttt{Sign}(m)$. Intuitively, this models that unregistered parties cannot use digital signatures (in a meaningful way): while they could still sign with a digital signature scheme without calling the ideal functionality, their secret key would be accessible by the adversary as part of their state.

Remark. The way we defined the signature functionality implies that we consider protocols in which signatures are not used in any other way than for signing and verification. For instance, the model does not allow parties to encrypt or hash signatures. For our purposes, this minimalistic ideal functionality is sufficient, and we refer to [9] for a discussion on ideal functionalities for digital signatures.

Theorem 4.3. *Suppose that $t - s \geq (n - s)/3$ with $n - s \geq 3$. Then there is no broadcast protocol that terminates in R rounds and succeeds with probability $1 - 1/(8R)$, even for a registered sender, according to the model defined above.*

4.3 The Probability Bound is Tight up to Constants

It is worth noting that it is not an artifact of the proof strategy we employed in deriving Theorem 4.3 that it leaves room for a success probability of $1 - 1/\Omega(R)$. To the contrary, a very simple protocol similar to the one of Fitzi et al. [13] achieves this, granted access to a shared coin. For simplicity, we assume that the coin is unbiased; a biased coin reduces the success probability accordingly.

(1) The sender multicasts its signed input.
(2) For $R-1$ rounds, honest parties echo the first two signatures they receive (all other messages are ignored).
(3) Afterwards, a shared coin is used to pick a value $1 \leq r \leq R$ uniformly at random.
(4) Each honest party outputs the default value if it received no or two conflicting signed values by round r. Otherwise, the signed value received first is the output.

This protocol runs for $R + 1$ rounds. If the sender is honest, it succeeds deterministically. Otherwise, before the shared coin is evaluated, the adversary must commit to the rounds

Simple Byzantine Agreement Protocol Π_{SBA}
Input and Initialization. Each party $P_i \in \mathcal{P}$ has input m_i.
Consensus of Registered Parties. Each registered party P_i ($\in \mathcal{S}$): participate in a deterministic authenticated Byzantine Agreement protocol (denoted as Π_{auth}) with input m_i, denote the output of P_i as b_i.
Message Dissemination. Each registered party P_i: sign b_i and send $(b_i, \text{sign}(b_i, i))$ to each unregistered party P_j ($\in \mathcal{N}$).
Majority Output Rule.
- Each unregistered party P_j: denote the majority value (with any prefixed tie-breaking rule) received by P_j in the Message Dissemination phase to be v_j, output v_j.
- Each registered party P_i: output b_i.

Figure 3: Byzantine Agreement Protocol Π_{SBA}

- $1 \leq r_1 \leq R$ when the first honest party receives some signed value x (if there is no such round, set $r_1 := R + 1$; in case of a tie, use any value x that an honest party echoes) and
- $r_1 \leq r_2 \leq R$ in which the first honest party sees a signature on a different value than x (again, $r_2 := R + 1$ if there is no such round).

Observe that the echoing ensures that each honest party sees x by round $r_1 + 1$ and each honest party sees conflicting values by round $r_2 + 1$ (unless $r_1 = R$ or $r_2 = R$, respectively). Thus, the only two values of the shared coin for which the outputs of honest parties might not satisfy consistency are r_1 and r_2. As the probability that $r = r_1$ or $r = r_2$ is at most $2/R$, the protocol thus succeeds with probability at least $1 - 2/R$.

5 Upper Bounds for BA and BC with an Unregistered Sender in the Partially Authenticated Setting

We show a simple Byzantine Agreement protocol Π_{SBA} in the partially authenticated setting that tolerates at most $t < s/2$ Byzantine parties (Fig. 3). We note that when $s \leq 2n/3$, $t < s/2$ implies $t < n/3$. Since it is well-known that an unauthenticated Byzantine Agreement protocol suffices to solve the BA problem [5], we focus on the interesting case when $s > 2n/3$.

Theorem 5.1. *Assume that Π_{auth} is a deterministic protocol that solves authenticated Byzantine Agreement among s parties against up to t Byzantine corruptions whenever $2t < s$. Then the above protocol Π_{SBA} solves Byzantine Agreement in the partially authenticated setting against up to t Byzantine corruptions whenever $2t < s$.*

Discussion. Independently, ignoring the PKI and running any deterministic unauthenticated BA protocol among all n parties yields feasibility when $3t < n$. Combining these two approaches implies that BA is achievable for $t \leq \max\{\lceil s/2 \rceil, \lceil n/3 \rceil\} - 1$.

Corollary 5.2. *There exists a Byzantine Agreement protocol in the partially authenticated setting where s parties have PKI setups that tolerates $t \leq \max\{\lceil s/2 \rceil, \lceil n/3 \rceil\} - 1$ Byzantine parties.*

We note that the Agreement protocol Π_{SBA} can be easily transformed into a broadcast protocol with the same number of keyholders that tolerates the same number of Byzantine parties: the

(Binary) Broadcast Protocol Π_{BCS}

Input and Initialization. Denote the Sender $P_S \in \mathcal{S}$. Each party P_i sets value $v_i = 0$ at the beginning of the protocol. If $P_i = P_S$ and $b = 1$, set $v_i \coloneqq 1$.

Gradecast. Each party P_i: participate in the Gradecast protocol with input value $b \in \{0, 1\}$ iff $P_i = P_S$. Denote the output of party P_i to be (v_i, g_i).

Weak Consensus of Non-PKI parties (among at most 2 parties).

(1) Each unregistered party $P_j \in \mathcal{N}$: send v_j to all parties $\in \mathcal{N}$.
(2) Each unregistered party $P_j \in \mathcal{N}$: set $m_j = 1$ if it received at least one $v_i = 1$ from $P_i \in \mathcal{N}$ in the last round. Otherwise, set $m_j = 0$.

Output Rule.

- Each registered party $P_i \in \mathcal{S}$: output v_i.
- Each unregistered party $P_j \in \mathcal{N}$:
  - If $g_j = 1$, output v_j.
  - Otherwise, output m_j.

Figure 4: (Binary) Byzantine Broadcast Protocol Π_{BCS} in the partially authenticated setting (special case)

sender first sends its input to all parties, then all parties run this Π_{SBA} protocol and output Π_{SBA}'s output.

6 Broadcast with a Registered Sender: The Special Case of $n - s < 3$

Theorem 3.14. *If $n - s < 3$, there exists a Byzantine Broadcast protocol that tolerates any number of corruptions, given the condition that the sender is registered.*

If $s \geq n - 1$, the proof is immediate from the paper of Dolev and Strong [10]. So it remains to argue about the case of $s = n - 2$.

Lemma 6.1. *The protocol Π_{BCS} satisfies consistency if $s = n - 2$ and $t \leq n - 2$.*

Proof. Suppose there exists an honest party $P_i \in \mathcal{S}$. In this case, denote its output of Π_{GC} as (v_i, g_i). From Lemma 3.10, each honest party P_j outputs (v_j, g_j) such that $g_j = 1$ at the end of the protocol Π_{GC}. From the graded consistency, all honest parties must have $v_j = v_i$. According to the output rule, each party $P_j \in \mathcal{S}$ therefore outputs $v_j = v_i$. For each party P_j in $\mathcal{N}$, we have argued above that it has $g_j = 1$. So it outputs $v_j = v_i$ from our output rule. This means that all honest parties output v_i, completing this case.

So assume instead that all parties in $\mathcal{S}$ are Byzantine. Since $s = n - 2$, and all of those parties are corrupted, $s = t = n - 2$. So the set $\mathcal{N}$ is of size 2 and both of these parties are honest. There are two cases:

- **An honest $P_j \in \mathcal{N}$ has $(v_j, g_j = 1)$.** In this case, from the graded consistency (Lemma 3.7), each honest party P_h must output (v_h, g_h) in the gradecast protocol Π_{GC} such that $v_h = v_j$. If $v_j = 0$, each honest party P_h outputs $m_h = 0$ in the non-PKI weak consensus, all honest parties (note they all belong to $\mathcal{N}$) output $v = 0$ according to our output rule of Π_{BCS}. If $v_j = 1$, from description of the non-PKI weak consensus of the protocol Π_{BCS}, each honest party P_h outputs $m_h = 1$, all honest parties output 1 according to our output rule of Π_{BCS}.
- **All honest parties $P_j \in \mathcal{N}$ have $(v_j, g_j = 1)$.** In this case, the outputs of both honest parties are fully determined by the messages they exchange in the weak consensus phase of the protocol. As they see the same messages, they output the same value.

In conclusion, the consistency is satisfied if $s = n - 2$ and $t \leq n - 2$. □

Acknowledgments

Julian Loss was supported by the European Union, ERC-2023-STG, Project ID: 101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work was also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2019. Communication Complexity of Byzantine Agreement, Revisited. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, Peter Robinson and Faith Ellen (Eds.). ACM, New York, NY, USA, 317–326. doi:10.1145/3293611.3331629

[2] Piyush Bansal, Prasant Gopal, Anuj Gupta, Kannan Srinathan, and Pranav K. Vasishta. 2011. Byzantine Agreement Using Partial Authentication. In *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings (Lecture Notes in Computer Science)*, David Peleg (Ed.). Springer, Berlin, Heidelberg, 389–403. doi:10.1007/978-3-642-24100-0_38

[3] Birgit Baum-Waidner, Birgit Pfitzmann, and Michael Waidner. 1991. Unconditional Byzantine Agreement with Good Majority. In *STACS 91, 8th Annual Symposium on Theoretical Aspects of Computer Science, Hamburg, Germany, February 14-16, 1991, Proceedings (Lecture Notes in Computer Science)*, Christian Choffrut and Matthias Jantzen (Eds.). Springer, Berlin, Heidelberg, 285–295. doi:10.1007/BFB0020806

[4] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, Janos Simon (Ed.). ACM, New York, NY, USA, 1–10. doi:10.1145/62212.62213

[5] Piotr Berman, Juan A Garay, and Kenneth J Perry. 1992. Bit optimal distributed consensus. In *Computer science: research and applications*. Springer, Berlin, Heidelberg, 313–321.

[6] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. In *TCC 2020: 18th Theory of Cryptography Conference, Part I (Lecture Notes in Computer Science, Vol. 12550)*, Rafael Pass and Krzysztof Pietrzak (Eds.). Springer, Cham, Switzerland, Durham, NC, USA, 353–380. doi:10.1007/978-3-030-64375-1_13

[7] Malte Borcherding. 1995. On the Number of Authenticated Rounds in Byzantine Agreement. In *Distributed Algorithms, 9th International Workshop, WDAG '95, Le Mont-Saint-Michel, France, September 13-15, 1995, Proceedings (Lecture Notes in Computer Science)*, Jean-Michel Hélary and Michel Raynal (Eds.). Springer, Berlin, Heidelberg, 230–241. doi:10.1007/BFB0022150

[8] Malte Borcherding. 1996. Partially authenticated algorithms for byzantine agreement. In *ISCA: Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems*. International Society for Computers and Their Applications, Winona, MN, USA, 8–11.

[9] Ran Cohen, Jack Doerner, Eysa Lee, Anna Lysyanskaya, and Lawrence Roy. 2024. An Unstoppable Ideal Functionality for Signatures and a Modular Analysis of the Dolev-Strong Broadcast. Cryptology ePrint Archive, Report 2024/1807. https://eprint.iacr.org/2024/1807

[10] Danny Dolev and H. Raymond Strong. 1983. Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* 12, 4 (1983), 656–666.

[11] Paul Feldman and Silvio Micali. 1988. Optimal Algorithms for Byzantine Agreement. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing,*

May 2-4, 1988, Chicago, Illinois, USA, Janos Simon (Ed.). ACM, New York, NY, USA, 148–161. doi:10.1145/62212.62225

[12] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. 2002. Detectable Byzantine agreement secure against faulty majorities. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. ACM, New York, NY, USA, 118–126.

[13] Matthias Fitzi, Chen-Da Liu-Zhang, and Julian Loss. 2021. A New Way to Achieve Round-Efficient Byzantine Agreement. In *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen (Eds.). ACM, New York, NY, USA, 355–362.

[14] Juan A. Garay and Yoram Moses. 1993. Fully polynomial Byzantine agreement in t+1 rounds. In *25th Annual ACM Symposium on Theory of Computing*. ACM Press, San Diego, CA, USA, 31–41. doi:10.1145/167088.167101

[15] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. ACM, New York, NY, USA, 51–68.

[16] S. Dov Gordon, Jonathan Katz, Ranjit Kumaresan, and Arkady Yerukhimovich. 2014. Authenticated broadcast with a partially compromised public-key infrastructure. *Inf. Comput.* 234 (2014), 17–25. doi:10.1016/J.IC.2013.11.003

[17] Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. 2010. Authenticated Byzantine Generals in Dual Failure Model. In *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3-6, 2010. Proceedings (Lecture Notes in Computer Science)*, Krishna Kant, Sriram V. Pemmaraju, Krishna M. Sivalingam, and Jie Wu (Eds.). Springer, Berlin, Heidelberg, 79–91. doi:10.1007/978-3-642-11322-2_12

[18] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2007. Zero-knowledge from secure multiparty computation. In *39th Annual ACM Symposium on Theory of Computing*, David S. Johnson and Uriel Feige (Eds.). ACM Press, San Diego, CA, USA, 21–30. doi:10.1145/1250790.1250794

[19] Jonathan Katz and Chiu-Yuen Koo. 2006. On Expected Constant-Round Protocols for Byzantine Agreement. In *Advances in Cryptology – CRYPTO 2006 (Lecture Notes in Computer Science, Vol. 4117)*, Cynthia Dwork (Ed.). Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA, 445–462. doi:10.1007/11818175_27

[20] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* 4, 3 (1982), 382–401. doi:10.1145/357172.357176

[21] Christoph Lenzen, Julian Loss, Kecheng Shi, and Benedikt Wagner. 2026. Byzantine Consensus in the Partially Authenticated Setting. Cryptology ePrint Archive, Paper 2026/470. https://eprint.iacr.org/2026/470

[22] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. 1999. Verifiable Random Functions. In *40th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, New York, NY, USA, 120–130. doi:10.1109/SFFCS.1999.814584

[23] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (1980), 228–234.

[24] B Pfitzmann and M Waidner. 1996. *Information-theoretic pseudosignatures and Byzantine agreement.* Technical Report. IBM Research, Tech. Rep. RZ 2882, Rep.