Combining Asynchronous and Synchronous Byzantine Agreement: The Best of Both Worlds

Julian Loss[1] and Tal Moran[2]

[1] Ruhr University Bochum, Germany
julian.loss@rub.de

[2] IDC Herzliya, Israel
talm@idc.ac.il

June 18, 2018

Abstract

In the problem of byzantine agreement (BA), a set of n parties wishes to agree on a value v by jointly running a distributed protocol. The protocol is deemed secure if it achieves this goal in spite of a malicious adversary that corrupts a certain fraction of the parties and can make them behave in arbitrarily malicious ways. Since its first formalization by Lamport et al. (TOPLAS '82), the problem of BA has been extensively studied in the literature under many different assumptions. One common way to classify protocols for BA is by their synchrony and network assumptions. For example, some protocols offer resilience against up to $f < \frac{n}{2}$ many corrupted parties by assuming a synchronized, but possibly slow network, in which parties share a global clock and messages are guaranteed to arrive after a given time Δ. By comparison, other protocols achieve much higher efficiency and work without these assumptions, but can tolerate only $f < \frac{n}{3}$ many corrupted parties. A natural question is whether it is possible to combine protocols from these two regimes to achieve the "best of both worlds": protocols that are both efficient *and* robust. In this work, we answer this question in the affirmative. Concretely, we make the following contributions:

- We give the first generic compilers that combine BA protocols under different network and synchrony assumptions and preserve both the efficiency *and* robustness of their building blocks. Our constructions are simple and rely solely on a secure signature scheme.
- We prove that our constructions achieve optimal corruption bounds.
- Finally, we give the first efficient protocol for (binary) asynchronous byzantine agreement (ABA) which tolerates *adaptive* corruptions and matches the communication complexity of the best protocols in the static case.

1 Introduction

One of the most fundamental problems in distributed computing and cryptography is the problem of *byzantine agreement* (BA). In this problem, a set of n parties, each holding an input v_i, aims to agree on a value v by jointly running a distributed protocol. Their task is complicated by malicious parties trying to prevent agreement by deviating from the protocol description in arbitrary ways. Byzantine agreement has countless practical and theoretical applications. Most commonly, it is used as a building block to design more complex systems which should satisfy strong consistency guarantees, e.g. databases, replicated services, or secure voting mechanisms. The related (slightly easier) problem of *broadcast* (BC) also has many applications to secure multi party computation (MPC).

Formally, a protocol for BA must satisfy the following three properties. *Termination:* Every honest party P_i eventually terminates the protocol with some output v_i'. *Consistency:* All honest parties output the same value v'. *Validity:* If all honest parties input $v_i = v$ then every honest party outputs v.

The problem of BA was first introduced in the seminal work of Lamport et al. [25] and has since been extensively studied for almost four decades under various assumptions. Very roughly speaking, protocols from the literature can be separated into two classes.

- *Synchronous Protocols:* These protocols require synchronization in the form of a global clock shared among the parties. Protocols in the synchronous model are round-based and crucially rely on a network that guarantees the delivery of messages within some a priori known time bound Δ. Protocols in this regime can tolerate up to $f < \frac{n}{2}$ maliciously corrupted parties.

- *Asynchronous Protocols:* This type of protocols does not require the above assumptions. In particular, protocols in this setting achieve byzantine agreement in spite of arbitrary (but finite) message delays. The main challenge in this setting is to distinguish between a party whose message is merely delayed by the network and one that has "failed" (and did not send a message at all). Asynchronous protocols for byzantine agreement (ABA) can tolerate at most $f < \frac{n}{3}$ maliciously corrupted parties.

In order to guarantee message delivery even for remote parties that suffer from a poor connection to the network, the parameter Δ is chosen as an upper bound on the real network delay δ. Typically, Δ is chosen rather pessimistically, i.e, $\Delta \gg \delta$. Therefore, synchronous protocols are usually employed whenever robust protocols with a high tolerance for corruptions are needed and efficiency takes only second priority. On the other hand, for many applications, efficiency is more important than robustness. In such a setting, asynchronous protocols are preferable to their synchronous counterparts, because they do not require a priori bounds and thus parties can take full advantage of a fast network. In line with [30], we will call protocols with this property *responsive protocols*. A natural question that arises from the above discussion is whether it is possible to combine protocols under different synchrony assumptions to obtain a hybrid protocol with best-of-both-worlds properties in terms of robustness and efficiency.

1.1 Our Results

In this work, we present novel constructions that achieve precisely such guarantees by compiling existing protocols under different synchrony assumptions into a new protocol that boasts the beneficial properties of both synchronous and asynchronous protocols.

Best-of-both-worlds compilers Concretely, our generic compiler combines protocols Π_{ABA} and Π_{SBA} for asynchronous and synchronous byzantine agreement, respectively, and leads to a hybrid protocol Π_{HBA} for byzantine agreement with the following properties.

- For all $f_{\mathsf{AR}} \leq \frac{1}{4}$, if Π_{ABA} achieves byzantine agreement, given that less than an f_{AR}-fraction of the parties are corrupted, then Π_{HBA} is responsive in the following sense: If the network is fast and less than an f_{AR}-fraction of the parties are corrupted, then every honest party can produce *output* in Π_{HBA} within a time that depends only on the network delay δ. We refer to this property as *output responsiveness.*

- For all $f_{\mathsf{AV}} \leq \frac{1}{2}$, if Π_{ABA} satisfies validity, given that less than an f_{AV}-fraction of the parties are corrupted, Π_{HBA} also satisfies validity under the same condition.

- If Π_{SBA} achieves byzantine agreement in time t_{SBA}, given that less than half of the parties are corrupted, then Π_{HBA} also achieves $\frac{1}{2}$-consistency.
- Π_{HBA} is guaranteed to terminate by time $t_{\mathsf{out}}+\Delta+t_{\mathsf{SBA}}$, where t_{out} is a time-out parameter that can be chosen arbitrarily in Π_{HBA}. In particular, if $t_{\mathsf{SBA}} = t_{\mathsf{start}} + O(\Delta)$ (where t_{start} is the protocol starting time), then choosing $t_{\mathsf{out}} = O(\Delta)$ implies that Π_{HBA} runs in $O(1)$ synchronous rounds.

We present Π_{HBA} in section 4.1, with an informal analysis. The main properties achieved by Π_{HBA} are stated in theorem 4.10. In section 4.3, we also give an alternative compiler which leads to a responsive hybrid protocol Π_{ETHBA} in which parties can *terminate* immediately after outputting and within a time that depends only on the network delay δ. We refer to this property simply as *responsiveness.* In addition, Π_{ETHBA} satisfies the same security guarantees as Π_{HBA}, but incurs a worst-case overhead in running time of $O(n)$ synchronous rounds if either the network is slow or too many parties are corrupted. The properties of Π_{ETHBA} are summed up in Theorem 4.24.

Security against *adaptive* adversaries Protocols obtained via our compilers preserve security guarantees against adaptive adversaries offered by the components Π_{ABA} and Π_{SBA}. In particular, the responsiveness guarantees offered by our hybrid protocols do not degrade under adaptive corruptions. This is an important improvement over previous works such as [31] that offer security and responsiveness only in the weaker model of *mildly adaptive corruptions*, which take a short while to become active.

More generally, our protocol improves upon *optimistic protocols*, which immediately lose all of their responsiveness properties under adaptive corruptions. We provide a more detailed comparison of such protocols with ours in section 1.3.

Optimality of our construction In Section 4.2, we prove that for the parameters $f_{\mathsf{AR}}, f_{\mathsf{AV}}$ such that $f_{\mathsf{AV}} \leq \frac{1}{2}(1 - f_{\mathsf{AR}})$, our compilers are optimal. Namely, no protocol Π_{HBA} can achieve *both* output-responsiveness when less than an f_{AR}-fraction of parties is corrupted and validity when at least an $\frac{1}{2}(1 - f_{\mathsf{AR}})$-fraction of parties is corrupted.

Since existing Π_{ABA} protocols do not offer validity above $\frac{n}{3}$ corrupted parties, they do not give the optimal parameters when plugged into our transformation. However, our transformation does not require consistency of Π_{ABA} above an f_{AR}-fraction of corrupted parties. We make use of this by constructing a second compiler that converts any Π_{ABA} protocol achieving termination, validity, and consistency for less than an f_{AR}-fraction of corrupted parties into a new Π_{ABA} protocol that attains the desired properties.

Concretely, this means that the new protocol achieves termination given that less than an f_{AR}-fraction of parties is corrupted and validity, given that less than a $\frac{1}{2}(1 - f_{\mathsf{AR}})$-fraction is corrupted, but may violate consistency, given that at least an f_{AR}-fraction of parties is corrupted. Combined with our compilers from above, we therefore show how to obtain optimal parameters for our compilers from any given Π_{ABA} protocol that achieves byzantine agreement, given that less than a $\frac{1}{4}$-fraction of the parties is corrupted.

Communication-efficient ABA with adaptive security In Section 6, we present a novel common coin protocol that leads to a new, highly efficient protocol for binary ABA (BABA) which achieves security for up to $f < \frac{n}{3}$ *adaptive* corruptions. This protocol has an overall communication complexity of $O(n^2)$, in line with the state-of-the-art for the best adaptively-secure *synchronous* protocols. Plugging this into our best-of-both-worlds compiler, the resulting hybrid protocol can also achieve the best of both worlds in terms of communication complexity.

Of independent interest, our result resolves the long-standing open question of obtaining an efficient BABA protocol that tolerates adaptive corruptions and presents a significant improvement over the best known solution in this setting (due to [9]), which requires $O(\kappa n^4)$ total communication complexity (here, κ denotes a security parameter).

1.2 Overview of Our Compiler

At a high level, our compiler uses the synchronous protocol as a slow, but robust fallback path in case the asynchronous protocol fails to reach agreement within a reasonable amount of time.

When combining protocols for BA for different synchrony assumptions, the main technical difficulty comes from the fact that some 'early' parties may obtain an output in the asynchronous path of the protocol, while for other 'late' parties, either the network was running very slow or the adversary has corrupted sufficiently many parties to control the outcome of the protocol at will. In this case, the consistency property of the hybrid protocol demands that the output of the 'late' parties be equal to the output of the 'early' parties. Thus, Π_{HBA} must ensure that the late parties do not re-agree on a value that is inconsistent with the early parties' output as otherwise, it would not make any improvements over a synchronous protocol.

Here, we rely on ideas from the recent work of Pass and Shi [31]. In essence, their protocol lets an honest party output a value v, if it sees that at least $\frac{3n}{4}$ parties have signed it. This makes it impossible for an adversary controlling less than $\frac{n}{2}$ parties to split the honest parties' view, as it cannot generate sufficiently many signatures on distinct values v', v.

On the other hand, an adversary that controls $\frac{n}{4}$ or more parties may succeed in violating the *validity* property by making parties accept a message $v' \neq v$ in the case where every honest party has input v to Π_{HBA}. To prevent this from happening, we rely on the validity property of Π_{ABA}: namely, we are guaranteed that as long as less than $n \cdot f_{\mathsf{AV}}$ parties are corrupted, validity is achieved in Π_{ABA}. Therefore, if every honest party inputs v to Π_{ABA}, then every honest party that terminates its execution of Π_{ABA} must output v.

We can use this property as follows. Every party in Π_{HBA} first runs Π_{ABA} with its input to Π_{HBA}. Then, it signs its output v from Π_{ABA} and broadcasts it to everybody. It outputs v, as soon as it obtains $\frac{3n}{4}$ signatures on v. This ensures our 'early output' property (output-responsiveness) in case sufficiently many parties are honest.

Since no honest party ever broadcasts a value other than v, no adversary controlling less than $\frac{n}{2}$ parties can produce $\frac{3n}{4}$ valid signatures on a value other than v. Furthermore, if at least one honest party *does* output v, then it will broadcast the entire list of $\frac{3n}{4}$ signatures to the network. This ensures that every other honest party obtains v along with a valid proof of $\frac{3n}{4}$ signatures.

Finally, the parties can run Π_{SBA}, using either their initial input or the unique value that they have obtained together with a proof from another party. Our argument now ensures that if every honest party has input v to Π_{HBA}, then every honest party will also input v to Π_{SBA}, i.e., the input v of an honest party to Π_{SBA} is preserved by the above procedure. Therefore, by validity of Π_{SBA}, every honest party outputs v and terminates the protocol.

1.2.1 Naïve Solutions Don't Work

One might wonder whether the same type of guarantees could also be obtained by simply running a constant round asynchronous protocol Π_{ABA} in the synchronous model. However, as we sketch in section 3.1, this can actually lead to a protocol which runs in $O(n)$ synchronous rounds *despite* tolerating only $f < \frac{n}{3}$ corrupted parties.

In comparison, the protocols Π_{HBA} and Π_{ETHBA} we have sketched above can tolerate up to $\frac{1}{2}(1 - f_{\mathsf{AR}}) \leq \frac{3}{8}$ corruptions, given suitable subcomponents and *always* run in a number of synchronous rounds that depends on the worst case running time of Π_{SBA}.

Moreover, the naïve solution does not allow for early termination, i.e., responsiveness, of the parties. All bets are off if, say, the parties run Π_{ABA} and terminate immediately after obtaining output. Namely, a party that participates honestly in Π_{ABA} is considered malicious if it does not complete the protocol with the remaining parties that have not yet obtained output. On the other hand, if the parties simply run Π_{ABA} in a synchronous network then responsiveness is immediately lost, because the time until termination now depends on the parameter Δ.

1.3 Related Work

Owing to its importance, there is a vast body of literature on the problem of byzantine agreement and related problems. We focus here on closely related work.

1.3.1 Optimistic Protocols and Their Limitations

A common paradigm in the literature to obtain protocols with high efficiency is to take an *optimistic approach.* Protocols of this type try to reach agreement by optimistically implementing an efficient strategy that works as long as the adversary does not carry out a specific attack.

For example, a widely implemented strategy is to elect a leader who distributes messages among the parties to prevent expensive all-to-all communication. As long as the leader is not corrupted, the protocol keeps running at a very efficient rate. On the other hand, the honest parties can use time-outs to detect when the leader becomes unavailable or acts maliciously to prevent agreement for a prolonged period of time, and eventually switch to a new leader.

This approach has been most widely used in the related (harder) problem of *state-machine-replication* (SMR) in which the parties agree instead on an *ordered log of values.* SMR protocols that use this approach include for example the well known PBFT protocol due to Castro and Liskov [15] as well as the works of [22, 3, 34, 35]. Another example of an optimistic protocol is considered in the elegant work of [24], which makes fast progress as long as no party behaves maliciously and switches to a pessimistic, more robust fallback mode otherwise. Interestingly, contrary to our approach, the work of [24] considers an optimistic case with a fast *synchronous* network and uses an asynchronous fallback. Another protocol that loosely fits this category is the Algorand BA protocol by Chen et al. [16], which also optimistically relies on a leader to reach fast agreement, but also has a fallback strategy which still guarantees agreement within a constant number of rounds if the leader is malicious. Their protocol does not require synchronization between the parties and merely requires that time passes at the same rate for all of them and that the network has a bounded delay. In contrast, our compilers do rely on a mild synchronisation between the parties (see section 3). However, their protocol requires a $\frac{2}{3}$ honest majority whereas our compilers lead to protocols that can tolerate even (up to) $\frac{1}{2}$ honest majority. Aside from this, our work aims to achieve *generic* compilers for BA, whereas the work of [16] presents a *specific* approach to achieve BA.

Optimistic protocols behave very well in the common case where corruptions occur infrequently or according to a fixed distribution. Indeed, these assumptions appear to be justified for many practical applications. However, one of the most important applications of BA protocols is their use as subcomponents to cryptographic protocols, which typically consider a much more powerful adversary that can corrupt parties also in a maliciously predetermined or even adaptive fashion. In such cases, optimistic protocols such as the above tend to fare poorly.

As an important example, BA protocols have recently enjoyed renewed interest from the cryptographic community in the design of cryptocurrency protocols. Here, the use of optimistic BA and SMR protocols can be somewhat problematic since an adaptive adversary can, for example, launch a Denial-of-Service attack to prevent the parties from making progress.

1.3.2 Comparison with Thunderella

An interesting example in this area is the recent work of Pass and Shi [31], which we have already mentioned. In their protocol for SMR, they use a designated party called the *accelerator* to stamp transactions with increasing sequence numbers and distribute them to the network. Once a party sees a stamped transaction, it signs the transaction and broadcasts it to the network. When a party garners $\frac{3n}{4}$ signatures from the parties on a single transaction, it accepts it.

As long as the accelerator and at least $\frac{3n}{4}$ parties are honest, this strategy guarantees that per sequence number, only a single transaction is accepted by the honest parties. Moreover, since the above steps can be carried out in a fully asynchronous manner, the above protocol has the responsiveness property.

If the accelerator or more than $\frac{n}{4}$ parties become corrupted, the protocol uses an underlying synchronous SMR protocol to detect that progress is no longer being made. In this case, the parties agree to fall back to the synchronous protocol for a while, until they later restart to run the optimistic strategy by electing a new accelerator. Importantly, their protocol tolerates $f < \frac{n}{2}$ corruptions in its fallback mode, whereas all of the above protocols fail whenever $f \geq \frac{n}{3}$.

On the downside however, their protocol can easily be degraded to a slow, fully synchronous SMR protocol by an adaptive adversary that immediately corrupts the accelerator after its election. Thus, their protocol suffers from the same weaknesses as the aforementioned works when confronted with an adaptive adversary. More importantly however, their approach seems to be inherently limited to the realm of SMR protocols. Though generic transformations from SMR to BA exist, it is unclear how their optimistic properties would translate to the case of BA. Furthermore, these transformations are not efficient, as they require to run the the SMR protocol for $O(n)$ rounds in order to achieve BA even once.

1.3.3 Previous Work On Combining Asynchronous and Synchronous Protocols

In a related, but different line of work, two previous works study the question of how much initial synchronous computation is needed to be able to switch to fully asynchronous computation afterwards. Concretely, the work of Beerilova et al. [4] shows that one initial round of synchronous broadcast is enough to perform asynchronous multi-party computation against an $\frac{n}{2}$-minority of malicious parties. Fitzi and Nielsen [20] showed that for the case of BA (and without a broadcast channel available during the synchronous rounds), $\frac{3f}{2} - \frac{n}{2} + O(1)$ initial synchronous rounds are sufficient in order to switch to fully asynchronous communication afterward (where again $f < n$ denotes the number of malicious parties).

2 Preliminaries and Notation

In this section, we recall some basic notation and definitions.

2.1 Notation

We denote algorithms with serif-free letters A. We use the standard probabilistic polynomial time efficiency and negligibility notions with respect to some security parameter λ. We write $x \leftarrow S$ to denote that variable x is sampled uniformly at random from set S. We write $(y_1, y_2...) \leftarrow \mathsf{A}(x_1, x_2...)$ to denote that algorithm A produces outputs $y_1, y_2...$ when run on inputs $x_1, x_2...$ We write $[n]$ to denote the integers $\{1, ..., n\}$.

3 Model

In this work, we consider the problem of byzantine agreement among a set of parties $P_1, ..., P_n$.

Definition 3.1 (Byzantine Agreement)**.** A distributed protocol Π among n parties $P_1, ..., P_n$ where party P_i initially holds input v_i achieves *byzantine agreement* if the following three properties are satisfied and the randomness is taken over the coins of the honest parties.

- Validity: If for every honest party P_i, $v_i = v$, then every honest party outputs v with overwhelming probability.
- Consistency: Every honest party outputs the same value v with overwhelming probability.
- p-Termination: Every honest party eventually outputs some value with probability at least p.

We consider the following setting:

- *Network assumptions*: We assume that the parties are connected via pairwise, reliable channels. In particular, any message that is sent over a channel is guaranteed to arrive after at most time Δ. For simplicity, we also assume that the channels are authentic (this is implied by the assumption of a public key infrastructure). Other than this, the adversary has full control over the network: It has the power to delay messages arbitrarily up to Δ time steps, it can reorder messages, and it can make some messages arrive multiple times at its intended recipient.
- *Synchronous Model*: In our protocols, we assume that the parties are in lockstep. This means that they proceed rounds of fixed length Δ which they enter at most some bounded number of (real) time steps apart. However, as shown in the recent work of Abraham et al. [1], bounded delay and local clocks with *bounded drift* (i.e., difference in the clock rates) are sufficient to achieve lockstep synchrony. For simplicity, we use the term 'round' and 'time' interchangeably (these are equivalent when clocks are globally synchronized).
- *Setup assumptions*: Parties initially share a *public key infrastructure* that is set up by a trusted dealer before the start of the protocol. We denote by (sk_i, pk_i) the secret/public key pair of party P_i. Throughout the following sections, we assume the existence of a signature scheme that satisfies the standard security notion of *unforgeabitliy under chosen message attacks*. We write $\sigma_i \leftarrow \mathsf{Sign}(v, sk_i)$ to denote that a party computes a signature on v using its secret key sk_i. σ_i can in turn be verified using the corresponding public key, pk_i.
- *Adversarial Model*: We consider a malicious, fully adaptive adversary that can corrupt any party at any given point in time. A malicious adversary in this setting is typically referred to in the literature as 'byzantine'. A party corrupted in a byzantine fashion can deviate arbitrarily from the protocol description, for example by not participating or equivocating to different parties. Upon corruption of a party P, the adversary learns the entire internal state of P. In particular, the adversary knows the initial state of all parties that are corrupted at the beginning of the protocol. However, the adversary does not know the internal state of the honest parties, which includes any secret values that they obtain from the honest dealer at the beginning of the protocol.

As already pointed out, all entities that we consider, i.e., the adversary, the honest parties, and the honest dealer, are assumed to be PPT algorithms.

3.1 Running Asynchronous Protocols in a Synchronous Network

Getting the most out of a fast network. It is important to note that even though messages in our model can be delayed by at most Δ time steps, it is possible that they arrive much faster, i.e. within some time $\delta \ll \Delta$. In this case, fully synchronous protocols could run very slowly, since they pessimistically proceed in rounds of a priori bounded length. Therefore, when a $\frac{2n}{3}$ majority can be ensured, it is often preferable to use an asynchronous protocol even if the parties share a global clock.

Caveat: Asynchronous rounds might blow up. One might be tempted to say that the asynchronous protocol would *always* be preferable in this case, since at worst it would devolve to a synchronous protocol. Somewhat surprisingly, however, this argument doesn't hold when the network is slow. In this case, simply running an asynchronous protocol in a synchronized, i.e., round-based fashion, may incur an overhead of $O(n)$ synchronous rounds until every party has terminated—even if the asynchronous protocol has $O(1)$ (asynchronous) rounds. In appendix A.2, we sketch how this blow-up can occur if the reliable broadcast protocol of Bracha [8] is naively run in a round-based fashion, i.e., when the parties proceed in synchronized rounds of length Δ.

Using time outs. To mitigate this blow up in round complexity, one can use the parameter Δ to define time-outs in our protocol. At a high level, this means that if a party has waited for some sufficiently long time $t(\Delta)$ without making progress in the asynchronous protocol, then it can proceed with the next step. Indeed, time-outs have been used in a model that sometimes is referred to as the *partially synchronous* model. Protocols in this model typically rely on a *leader* (sometimes called the *primary*) to ensure progress. If the leader becomes unresponsive, the protocol executes a leader replacement subprotocol called a *view change protocol.* The main issue with known protocols in this model is that once the leader is known to all parties, an adaptive adversary can immediately corrupt it and thus force the protocol to repeatedly execute expensive view changes without making progress.

Our approach. We circumvent this problem by showing a different strategy that combines an asynchronous protocol with a synchronous one *without the use of a leader.* Our protocol has the useful property that it runs at the network's speed when more than $\frac{3n}{4}$ of the parties are honest but can tolerate up to $\frac{3n}{8}$ corrupted parties while still requiring only a constant amount of synchronous rounds. We then show in Section 4.2 that the parameters in our transformation are optimal.

3.2 Composition of Hybrid Protocols

All our results are proven in the standalone model, as is common for works in the area of byzantine agreement. This means that our protocols do not necessarily remain secure when composed (sequentially or in parallel) or used without care as subcomponents within larger protocols. In section 5, we show how to sequentially compose HBA protocols so that the composed protocol is also responsive and secure—as long as the component HBA protocols remain secure under composition. It remains an interesting open question to formalize the notion of hybrid protocols for BA (in our sense) in the UC framework [13] and to prove our compilers secure with respect to such a formalization.

4 Generic Compilers for Byzantine Agreement Protocols

In this section, we propose solutions to the byzantine agreement problem that obtain 'best of both worlds guarantees.' More specifically, our protocol has the efficiency of an asynchronous protocol if the network is fast and sufficiently many parties are honest, but preserves the worst-case guarantees of a synchronous protocol if the network is slow or up to $f < \frac{3n}{8}$ parties are dishonest.

The solution that we present generically interleaves a synchronous protocol with an asynchronous one to achieve this goal. The idea is to use the synchronous protocol as a slow, but robust fallback path in case the asynchronous protocol fails to reach agreement within a reasonable amount of time. The main challenge is to ensure that if an honest party obtains an output in the asynchronous protocol, it can directly output this value without having to wait for the synchronous protocol to terminate–otherwise, our protocol would make no improvement over the synchronous protocol.

We define the following properties (and abbreviations) for a byzantine agreement protocol Π_{BA}.

Definition 4.1. Let Π_{BA} be a protocol which achieves byzantine agreement among n parties. In the following, the probability is taken over the random coins of the honest parties and p is a non-negligible value.

- Π_{BA} is said to be (p, f_{AT})-*terminating* if, with probability p, every honest party terminates the protocol, given that less than an f_{AT}-fraction of the parties is dishonest.
- Π_{BA} is said to be (p, f_{AR})-*responsive* if, with probability at least p, every honest party terminates within some time that does not depend on Δ, given that less than an f_{AR}-fraction of the parties is dishonest. Note that (p, f_{AR})-*responsiveness* implies (p, f_{AR})-termination.
- Π_{BA} is said to be (p, f_{AR})-*output-responsive* if, with probability at least p, every honest party outputs a value within some time that does not depend on Δ, given that less than an f_{AR}-fraction of the parties is dishonest.
- Π_{BA} is said to be f_{AV}-*valid* if it has the validity property, given that less than an f_{AV}-fraction of the parties is dishonest.
- Π_{BA} is said to be f_{AC}-*consistent* if it has the consistency property, given that less than an f_{AC}-fraction of the parties are dishonest.

As a special case of our generic transform, we obtain a protocol for byzantine agreement that is *output responsive* as long as less than $\frac{n}{4}$ parties are corrupted and still guarantees termination, consistency, and validity in a constant number of synchronous rounds if less than $\frac{3n}{8}$ of the parties are corrupted. Interestingly, termination and consistency are preserved even up to a bound of $f < \frac{n}{2}$ corrupted parties. In other words, when less than $\frac{n}{4}$ parties are corrupted, the time until agreement is reached depends only on the actual speed of the network and not on some a priori established upper bound on the network delay. However, even if the network is slow and at most $\frac{3n}{8}$ parties are corrupted, our protocol still manages to guarantee agreement within a constant amount of synchronous rounds. We then show that these parameters are optimal in Section 4.2.

4.1 Output-Responsive Hybrid Byzantine Agreement

In the following, we describe our construction for *Hybrid Byzantine Agreement*, which we denote as Π_{HBA}. Every execution of Π_{HBA} is parameterized by a timeout parameter, t_{out}, which is shared by all honest parties. Note that we specify the timeout as an absolute time to allow the parties to start protocol execution at different times (with the restriction that they start the protocol before t_{out}). Allowing non-synchronized starting points is standard in the asynchronous model, and is required for responsiveness in our sequential composition. Note that if all parties *are* guaranteed to start together at some time t_{start}, they can define the timeout to be $t_{\mathsf{out}} = t_{\mathsf{start}} + t_{\mathrm{rel}}$ for some interval length t_{rel}.

Definition 4.2 (Hybrid BA)**.** We say a BA protocol Π_{HBA} is an (f_{AR}, p)-output-responsive,f_{AV}-valid,f_{AC}-consistent *hybrid* BA with running-time t_{HBA} iff:

- Π_{HBA} is an (f_{AR}, p)-output-responsive BA,
- Π_{HBA} is f_{AV}-valid,
- Π_{HBA} is f_{AC}-consistent,
- Every honest party executing $\Pi_{\mathsf{HBA}}(t_{\mathsf{out}})$ is guaranteed to terminate and produce output by time $t_{\mathsf{out}} + t_{\mathsf{HBA}}$.

Our Π_{HBA} construction makes black-box use of two subprotocols: an asynchronous protocol for byzantine agreement, Π_{ABA}, and a synchronous protocol for byzantine agreement, Π_{SBA}. We denote the running time of Π_{SBA} by t_{SBA}. For simplicity of notation, we will sometimes use Π_{HBA} as a short hand notation for $\Pi_{\mathsf{HBA}}(t_{\mathsf{out}})$ in the subsequent sections. In the following, assume that:

- Π_{ABA} is an asynchronous protocol for byzantine agreement that guarantees validity, consistency, and p-termination if less than nf_{AR} parties are dishonest and satisfies validity if less than nf_{AV} parties are dishonest.
- Π_{SBA} is a synchronous protocol for byzantine agreement that guarantees validity and consistency, given that less than $\frac{n}{2}$ parties are corrupted. Π_{SBA} runs in time t_{SBA}.
- Every honest party P_i starts the protocol Π_{HBA} at time $t^i_{\mathsf{start}} < t_{\mathsf{out}}$.
- $\frac{1}{2} > f_{\mathsf{AV}} \geq f_{\mathsf{AR}}$.

Figure 4.1 contains the view of party P_i for protocol Π_{HBA}.

4.1.1 Informal Description and Security Analysis

The idea of Π_{HBA} is as follows. Parties first run Π_{ABA} with their input to Π_{HBA}. Upon obtaining output from Π_{ABA}, they sign it and broadcast the signature to every party. If a party P_i obtains $\frac{3n}{4}$ signatures on any value v before time t_{out}, then it outputs v and broadcasts v along with a proof L_i containing the signatures. (Intuitively, L_i is a proof that v was a correct output of the Π_{ABA}.)

If a party did not terminate the Π_{ABA} until time t_{out}, it waits for another Δ interval to ensure that all messages that were sent prior to t_{out} have been received. It then participates in a run of Π_{SBA}, using either its initial input v_i as input to Π_{SBA} or any value upon which it has received $\frac{3n}{4}$ valid signatures after time t_{out} (there can only be one such value).

Since Π_{ABA} ensures termination for $nf_{\mathsf{AR}} < \frac{n}{4}$ corrupted parties, the honest parties can obtain the necessary $\frac{3n}{4}$ signatures for termination at a speed that depends only on the actual network delay whenever less than nf_{AR} parties are dishonest. In this way, Π_{HBA} guarantees f_{AR}-output responsiveness.

On the other hand, if the parties do not all terminate Π_{ABA}, it is impossible that two honest parties output different values v' and v, as this would imply that both of these values were signed at least $\frac{3n}{4}$ times. (this would lead to a contradiction, because it implies that more than half the parties signed *both* values, contradicting the assumption that more than half the parties are honest).

If at least one honest party P_i obtains such a list on a value v before time t_{out} (and therefore outputs v), every other honest party is ensured to receive the same list by time $t_{\mathsf{out}} + \Delta$ (since it was broadcast P_i at time t_{out}. Therefore, in this case, all honest parties use v as their input to Π_{SBA}. Validity of Π_{SBA} now ensures that all the parties agree on v and terminate.

Figure 4.1: $\Pi_{\mathsf{HBA}}(t_{\mathsf{out}})$ protocol (view of P_i)

- Let v_i denote the input of party P_i.
- P_i starts to execute Π_{ABA} with input v_i (note that parties might start the Π_{ABA} at different times).
- Initialize $v^* \leftarrow v_i$.
- Party P_i runs Π_{ABA} until it terminates Π_{ABA} or until time t_{out} (whichever comes first).
- If party P_i's view of Π_{ABA} has terminated with output v at time $t' < t_{\mathsf{out}}$, it computes a signature $\sigma_i \leftarrow \mathsf{Sign}(v, sk_i)$. It broadcasts (i, v, σ_i) to every party (including itself).
- Upon, receiving at least $\frac{3n}{4}$ valid signatures (from different parties) on a single value v' at time $t' < t_{\mathsf{out}}$, P_i sets $v^* \leftarrow v'$ outputs v^* and broadcasts (i, v^*, L_i), where L_i denotes a list containing these signatures. Note that this instruction may also be triggered upon receiving a correctly formed tuple (j, v_j, L_j) from party P_j.

---

time t_{out} (by shared, global clock)

- Upon receiving at least $\frac{3n}{4}$ valid signatures (from different parties) on a single value v' at time $t', t_{\mathsf{out}} \leq t' \leq t_{\mathsf{out}} + \Delta$, P_i sets $v^* \leftarrow v'$ (but does not output yet).

---

time $t_{\mathsf{out}} + \Delta$

- At time $t_{\mathsf{out}} + \Delta$, P_i participates in a run of Π_{SBA}, using v^* as its input. It outputs the output of Π_{SBA} (if it hasn't output anything yet) and terminates.

If no party outputs before running Π_{SBA}, then consistency trivially follows from the consistency of Π_{SBA}. Therefore, Π_{HBA} satisfies $\frac{n}{2}$-consistency. What remains to show is that Π_{HBA} also satisfies validity. Here, the idea is the following: If all parties initially hold v, then validity of Π_{ABA} ensures that every honest party either terminates Π_{ABA} with v or does not terminate Π_{ABA} at all. In either case, no party will ever sign a value other than v, which ensures that only proofs (lists of signatures) on v can be valid proofs. On the other hand, a party that never receives a proof during the protocol (before time $t_{\mathsf{out}} + \Delta$) runs Π_{SBA} with its initial input, which is v. The validity of Π_{HBA} now follows from the validity of Π_{SBA}.

4.1.2 Formal Analysis: Output-Responsiveness, Validity and Consistency

Lemma 4.3. *Let t_{SBA} be the execution time of the underlying Π_{SBA} protocol and t^i_{start} the time at which party i starts the Π_{HBA} protocol. If*

- $f_{\mathsf{AR}} \leq \frac{1}{4}$,
- $\Delta \leq t_{\mathsf{out}} - t^i_{\mathsf{start}}$ *and*
- $t_{\mathsf{SBA}} \leq f(n) \cdot \Delta$, *where f is a function that does not depend on Δ*

Then Π_{HBA} is (p, f_{AR})-output responsive.

Proof. Suppose less than $nf_{\mathsf{AR}} \leq \frac{n}{4}$ parties are dishonest. By the consistency property of Π_{ABA}, every honest party that outputs a value in Π_{ABA}, outputs the same value v. Furthermore, by the p-termination property of Π_{ABA}, with probability at least p, every honest party eventually delivers the value v in Π_{ABA}.

Denote t_{ABA} the maximum (over the honest parties) of the time to execute the Π_{ABA} protocol (in executions where every honest party does deliver output). Note that since Π_{ABA} is an asynchronous protocol, t_{ABA} does not depend on Δ. We now consider two cases:

Case 1: $t_{\mathsf{ABA}} < t_{\mathsf{out}} - t^i_{\mathsf{start}}$. In this case, with probability at least p, every honest party in Π_{ABA} terminates and outputs v. Subsequently, every honest party broadcasts v along with a valid signature. This ensures that with probability at least p, every honest party P_i obtains at least $\frac{3n}{4}$ valid signatures on the value v by time $t_{\mathsf{ABA}} + \delta$. In this case, P_i immediately outputs v. Hence, all honest parties receive output by time $t_{\mathsf{ABA}} + \delta \leq (2 + f(n))t_{\mathsf{ABA}}$.

Case 2: $t_{\mathsf{ABA}} \geq t_{\mathsf{out}} - t^i_{\mathsf{start}}$. In this case, by the definition of t_{ABA}, at least one honest party did not receive output before time t_{out}. However, in any case, all honest parties are guaranteed to terminate after the Π_{SBA} protocol terminates, thus the total execution time for any honest party is bounded by

$$\begin{aligned} t_{\mathsf{out}} - t^i_{\mathsf{start}} + \Delta + t_{\mathsf{SBA}} &\leq t_{\mathsf{out}} - t^i_{\mathsf{start}} + \Delta + f(n) \cdot \Delta \\ &= t_{\mathsf{out}} - t^i_{\mathsf{start}} + (1 + f(n))\Delta \\ &\leq (2 + f(n))(t_{\mathsf{out}} - t^i_{\mathsf{start}}) \leq (2 + f(n))t_{\mathsf{ABA}} \end{aligned}$$

Thus, in both cases, with probability at least p the time to receive output is bounded by $(2 + f(n))t_{\mathsf{ABA}}$. Since this expression does not depend on Δ, Π_{HBA} is (p, f_{AR})-output responsive. □

For the remainder of the following sections, let us call a message (i, v, L) *correctly formed*, if it L contains at least $\frac{3n}{4}$ valid signatures on v from distinct parties.

Lemma 4.4. *Suppose that less than a $\frac{1}{2}$-fraction of the parties is dishonest. If P_i and P_j broadcast correctly formed messages (i, v, L_i) and (j, v', L_j), respectively, in Π_{HBA} at time $t' < t_{\mathsf{out}}$, then $v = v'$.*

Proof. Let $\epsilon > 0$ and suppose that $n(\frac{1}{2} - \epsilon)$ parties are dishonest. By assumption, L_i and L_j each contain at least $\frac{3n}{4}$ valid signatures on v and v', respectively. This means, that L_i and L_j each contain $\frac{3n}{4} - n(\frac{1}{2} - \epsilon) = \frac{n}{4} + \epsilon$ signatures from honest parties on v and v', respectively (since signatures are unforgeable). Since no honest party signs distinct messages v and v', there must be at least $2(\frac{n}{4} + \epsilon) = n(\frac{1}{2} + 2\epsilon)$ many honest parties. This is a contradiction, since by assumption, there are $n(\frac{1}{2} + \epsilon) < n(\frac{1}{2} + 2\epsilon)$ many honest parties. □

Lemma 4.5. *Suppose that less than an $\frac{1}{2}$-fraction of the parties is dishonest and let P_i be the first honest party that outputs v in Π_{HBA} at time $t' < t_{\mathsf{out}}$. Then all honest parties output v in Π_{HBA}.*

Proof. Since P_i outputs v at time $t' < t_{\mathsf{out}}$, it has sent a valid message of the form (j, v, L_j) to all parties by time t_{out}. Thus, all honest parties receive this message by time $t_{\mathsf{out}} + \Delta$, and set their inputs to Π_{SBA} to v (by lemma 4.4, no party P_k broadcasts a correctly formed message (k, v', L_k), s.t. $v' \neq v$.). Now the validity property of Π_{SBA} ensures that every honest party outputs v at the end of Π_{HBA}. □

Corollary 4.6. Π_{HBA} *is $\frac{1}{2}$-consistent.*

Proof. Lemma 4.5 ensures consistency in the case where an honest party outputs at time $t' < t_{\mathsf{out}}$. It remains to show that consistency also holds when no honest party outputs before time t_{out}. However, this trivially follows from the fact that now, every honest party will output whatever they obtain from running Π_{SBA}. Thus, consistency follows from the consistency property of Π_{SBA}. □

Lemma 4.7. *Suppose that less than an f_{AV}-fraction of the parties is dishonest and all honest parties input v to Π_{HBA}. Let P_i be an honest party that outputs in Π_{HBA} at time $t' < t_{\mathsf{out}}$. Then P_i outputs v.*

Proof. By validity of Π_{ABA}, every honest party that delivers a value in Π_{ABA}, delivers v. Therefore, no honest party P_j broadcasts a message of the form $(j, v', \sigma'_j), v \neq v'$. This ensures that no party can obtain a list L of $\frac{3n}{4}$ valid signatures on a value other than v (since less than $nf_{\mathsf{AV}} < \frac{n}{2}$ parties are dishonest and signatures are unforgeable). Therefore, P_i outputs v. □

Lemma 4.8. *Suppose that less than an f_{AV}-fraction of the parties is dishonest and all honest parties input v to Π_{HBA}. Further, suppose that no honest party outputs in Π_{HBA} at time $t' < t_{\mathsf{out}}$. Then every honest party outputs v in Π_{HBA} at time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBA}}$.*

Proof. By validity of Π_{ABA}, every honest party that delivers a value in Π_{ABA}, delivers v. Thus, no honest party P_j broadcasts a message of the form $(j, v', \sigma'_j), v \neq v'$. This ensures that no honest party will ever see $\frac{3n}{4}$ valid signatures on a value other than v (since less than $nf_{\mathsf{AV}} < \frac{n}{2}$ parties are dishonest and signatures are unforgeable). In particular, an honest party P_i will never set v^* to any value other than v, since v^* is initially set to $v_i = v$. This ensures that every honest party inputs v to Π_{SBA} at time $t_{\mathsf{out}} + \Delta$. Now, validity of Π_{SBA} guarantees that every honest party outputs v in Π_{HBA} at time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBA}}$. □

Corollary 4.9. Π_{HBA} *is f_{AV}-valid.*

Proof. Combining lemma 4.7 with corollary 4.20, if an honest party outputs v before time t_{out}, then every other honest party also outputs v. This ensures validity in the case where an honest party outputs before time t_{out}. On the other hand, if no party outputs before this time, then validity is ensured by lemma 4.8. □

We sum up the properties of Π_{HBA} in the following theorem.

Theorem 4.10. *Assume that:*

- Π_{ABA} *is an asynchronous protocol for byzantine agreement that guarantees validity, consistency, and p-termination if less than nf_{AR} parties are dishonest and satisfies validity if less than nf_{AV} parties are dishonest.*
- Π_{SBA} *is a synchronous protocol for byzantine agreement that guarantees validity and consistency, given that less than $\frac{n}{2}$ parties are corrupted.* Π_{SBA} *runs in time $t_{\mathsf{SBA}} \leq f(n)\Delta$ for some function $f(n)$ that does not depend on Δ.*
- $\frac{1}{2} > f_{\mathsf{AV}} \geq f_{\mathsf{AR}}$.

Then the following statements are true:

- *If $f_{\mathsf{AR}} \leq \frac{1}{4}$ and for all P_i, $t_{\mathsf{out}} - t^i_{\mathsf{start}} \geq \Delta$ then* Π_{HBA} *is (p, f_{AR})-output responsive.*
- Π_{HBA} *is $\frac{1}{2}$-consistent.*
- Π_{HBA} *is f_{AV}-valid.*
- Π_{HBA} *terminates at time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBA}}$.*

4.2 Optimality of Π_{HBA} and Π_{ETHBA}

In this section, we show that Π_{HBA} and Π_{ETHBA} (presented in the following section) achieve optimal parameters. Concretely, we show that it is not possible to obtain a hybrid protocol which achieves (p, f_{AR})-output responsiveness *and* f_{AV}-validity if $f_{\mathsf{AV}} > \frac{1}{2}(1 - f_{\mathsf{AR}})$. We also show that it is possible to convert any protocol Π_{ABA} which achieves binary byzantine agreement (with p-termination) when less than $f_{\mathsf{AR}} \leq \frac{n}{4}$ parties are corrupted into a protocol $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ that achieves (p, f_{AR})-termination, f_{AR}-consistency, and $\frac{1}{2}(1-f_{\mathsf{AR}})$-validity. This transformation can be used to transform an existing BABA protocol into a protocol that gives optimal parameters when plugged into Π_{HBA} or Π_{ETHBA}. Depending on whether (output)-responsiveness or validity are prioritized, it is possible to set the parameters in $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ accordingly (by setting f_{AR} in $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ to the desired value). The transformation $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ is described in Figure 4.2.

Figure 4.2: $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ protocol (view of P_i), with parameter f_{AR}.

1. Let b_i denote the input of party P_i.
2. P_i computes a signature $\sigma_i \leftarrow \mathsf{Sign}(b_i, sk_i)$ and broadcasts (i, b_i, σ_i) to every party (including itself).
3. P_i wait until it obtains $n(1 - f_{\mathsf{AR}})$ valid messages (i.e., with a valid signature of b_i under pk_i) of the form (i, b_i, σ_i) (from $n(1 - f_{\mathsf{AR}})$ different parties).
4. Let b denote the majority bit among the valid messages that P_i received. P_i broadcasts a message of the form (i, b, L_i) to every party (including itself), where the list L_i contains all the valid signatures that P_i received on b.
5. P_i runs Π_{ABA} with input b. Let b^* denote the output of Π_{ABA}.
6. Upon receiving a valid message of the form (j, b^*, L_j) (i.e., where L_i contains at least $n(1 - f_{\mathsf{AR}})$ valid signatures on b^* from different parties), P_i terminates the protocol with output b^*.

Lemma 4.11. *Let $f_{\mathsf{AR}} \leq \frac{n}{4}$ and let Π_{ABA} be a protocol for BABA that achieves (p, f_{AR})-termination, f_{AR}-validity, and f_{AR}-consistency. Then protocol $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ achieves (p, f_{AR})-termination, f_{AR}-consistency, and $\frac{1}{2}(1 - f_{\mathsf{AR}})$-validity.*

Proof. We proceed by proving the properties of $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ separately.

- (p, f_{AR})-*termination:* Assume that less than an f_{AR} fraction of the parties are corrupted. In this case, every honest party obtains at least $n(1 - f_{\mathsf{AR}})$ valid messages of the form (i, b_i, σ_i) in the third step of $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$ and subsequently broadcasts a valid message of the form (i, b, L_i), where b is the majority bit it has computed from these messages. It then runs Π_{ABA} on the bit b. By (p, f_{AR})-termination and f_{AR}-consistency of Π_{ABA}, with probability p, every honest party terminates Π_{ABA} in step five the with same bit b^* (note that if a party *does terminate* Π_{ABA}, then it terminates with b^*). By validity of Π_{ABA}, at least one honest party P_j has input b^* to Π_{ABA} and broadcasted a valid message of the form (j, b^*, L_j) in step 4. Thus, with probability p, every honest party will eventually obtain the message (j, b^*, L_j) and terminate. This ensures (p, f_{AR})-termination of $\Pi^{\mathsf{opt}}_{\mathsf{BABA}}$.
- f_{AR}-*consistency:*. Follows easily from f_{AR}-consistency of Π_{ABA}, since every party outputs b only if it has previously seen b as output from Π_{ABA}.

- $\frac{1}{2}(1-f_{\mathsf{AR}})$-*validity:* Assume that every honest party inputs b to $\Pi_{\mathsf{BABA}}^{\mathsf{opt}}$ and less than an $\frac{1}{2}(1-f_{\mathsf{AR}})$-fraction of the parties is corrupted. Therefore, if *any* party obtains $n(1-f_{\mathsf{AR}})$ valid messages of the form (i, b_i, σ_i) in step three, it obtains strictly more than $\frac{n}{2}(1-f_{\mathsf{AR}})$ such messages from honest parties. The majority bit computed from these messages is b, since by assumption, every honest party P_i has sent (i, b, σ_i) in step 2 (and signatures are unforgeable). It now follows that every valid message obtained by an honest party in the final step of the protocol must be of the form (i, b, L_i). Therefore, in the final step, every party either terminates with output b upon receiving a valid message of the form (j, b, L_j) or does not terminate (in case it received $1-b$ as output from Π_{ABA} in the previous step).

□

The dual-threshold structure of $\Pi_{\mathsf{BABA}}^{\mathsf{opt}}$ is reminiscent of the work of Fitzi et al. [19] who considered broadcast protocols (see section 4.3) with a two-threshold structure. In their protocols, either validity or consistency is lost when nf_1 or more parties are corrupted, but the second property is preserved until nf_2 or more parties are corrupted, where $f_1 < f_2$ and $2f_1 + f_2 < 1$. However, their work considers the notion of information theoretic broadcast in the synchronous model, whereas our results consider byzantine agreement in the asynchronous model with a computationally bounded adversary.

We now show that our construction for Π_{HBA}, combined with $\Pi_{\mathsf{BABA}}^{\mathsf{opt}}$, achieves optimal corruption bounds.

Lemma 4.12. *Let Π_{ABA} be a protocol for byzantine agreement which achieves validity, consistency, and p-termination when less than an f_{AR}-fraction of the parties are dishonest. Then Π_{ABA} does not satisfy validity if the fraction of corrupt parties $f_{\mathsf{AV}} \geq \frac{1}{2}(1-f_{\mathsf{AR}})$. Moreover, there exists an adversary controlling a $\frac{1}{2}(1-f_{\mathsf{AR}})$-fraction of the parties that violates validity and ensures p-termination for all honest parties.*

Proof. Let Π_{ABA} be a protocol for byzantine agreement that achieves validity, consistency, and p-termination in the asynchronous setting when less than nf_{AR} parties are dishonest. Let $\mathcal{H}$ denote the set of honest parties. It suffices to show that Π_{ABA} does not satisfy validity if exactly $\frac{n}{2}(1-f_{\mathsf{AR}})$ parties are dishonest. For this purpose, let $\ell > n(1-f_{\mathsf{AR}})$ be the minimum number of honest parties for which Π_{ABA} is still guaranteed to terminate for all honest parties with probability at least p.

Let $\mathcal{S}$ be the set of partitions of $[n]$ into three sets, S_0, S_1, S_X such that $|S_X| < n \cdot f_{\mathsf{AR}}$. We define $f : \mathcal{S} \times \mathcal{R}^{\ell} \to \{0, 1, \perp\}$ to be a randomized function. For $(S_0, S_1, S_X) \in \mathcal{S}$, the distribution of $f(S_0, S_1, S_X)$ is induced via Π_{ABA} as follows.

- Parties in S_0 have input 0, parties in S_1 have input 1.
- The messages of parties in S_X in Π_{ABA} are indefinitely delayed.
- Once every honest party in $S_0 \cup S_1$ has output a value, all messages from parties in S_X are delivered.
- The output of f is defined as v, if every honest party terminates Π_{ABA} with output v and as $\perp$ otherwise.

Note that by f_{AR}-consistency of Π_{ABA}, every honest party outputs the same value v or does not terminate Π_{ABA}. Since v cannot depend on messages from parties in S_X, the output distribution of f is always well defined for these inputs. Furthermore, observe that by p-termination of Π_{ABA}, $\Pr\{f(S_0, S_1, S_X) \neq \perp\} \geq p$.

For every partition $\bar{S} = (S_0, S_1, S_X) \in \mathcal{S}$, we can construct an adversary $\mathsf{A}_{\bar{S}}$ that corrupts at most $\max(|S_0|, |S_1|)$ parties and a set of inputs to the honest. We show in the following how this results in a violation of validity.

1. Let b be a bit such that $\Pr\{f(S_0, S_1, S_X) = b\} \geq \frac{p}{2}$ (this must hold for either $b = 0$ or $b = 1$).

2. Let the parties in S_X have input $1 - b$.

3. $\mathsf{A}_{\bar{S}}$ corrupts the parties in S_b, and instructs them to behave honestly.

By our definition of f, in an execution of the Π_{ABA} protocol in the presence of $\mathsf{A}_{\bar{S}}$, the honest parties output b with probability at least $\frac{p}{2}$. However, this is a violation of validity since all honest parties have input $1 - b$ and $\frac{p}{2}$ is non-negligible.

To compute the optimal parameters for the attack, we need to find a partition that minimizes $\max(|S_0|, |S_1|)$. This happens when $n - |S_X|$ is minimized and $|S_0| = |S_1| = \frac{1}{2}(n - |S_X|)$.

In a protocol that's valid as long as less than an f_{AV}-fraction of the parties are corrupted, validity should hold for every number of parties n as long as less than $n \cdot f_{\mathsf{AV}}$ parties are corrupted. Thus, to show validity is violated we can pick n such that $(1 - f_{\mathsf{AR}})$ divides n, $n - |S_X| = n(1 - f_{\mathsf{AR}})$ and $n - |S_X|$ is even.

Then $|S_0| = |S_1| = n \cdot \frac{1}{2}(1 - f_{\mathsf{AR}})$.

$n - |S_X| \geq n(1 - f_{\mathsf{AR}})$, so we can always set $|S_X|$ such that $n - |S_X| = \lfloor n(1 - f_{\mathsf{AR}}) \rfloor + 1 \leq n(1 - f_{\mathsf{AR}}) + 1$ When $n - |S_X|$ is odd, then we can split almost evenly, so in any case

$$\max(|S_0|, |S_1|) \leq \left\lceil \frac{1}{2}(n - |S_X|) \right\rceil \leq \frac{1}{2}(n - |S_X|) + 1 \tag{4.1}$$

$$\leq \frac{1}{2}(n(1 - f_{\mathsf{AR}}) + 1) + 1 = \frac{n}{2}(1 - f_{\mathsf{AR}}) + \frac{3}{2}. \tag{4.2}$$

□

Corollary 4.13. *If Π_{HBA} is both (p, f_{AR})-responsive and f_{AV}-valid, then $f_{\mathsf{AV}} < \frac{1}{2}(1 - f_{\mathsf{AR}})$.*

Proof. We prove the statement by contradiction. Thus, assume that Π_{HBA} is both (p, f_{AR})-responsive and f_{AV}-valid. Assume further that $f_{\mathsf{AV}} \geq \frac{1}{2}(1 - f_{\mathsf{AR}})$. We show that either (p, f_{AR})-responsiveness or f_{AV}-validity must be violated in this case. To see this, note that Π_{HBA} is also an *asynchronous* BA protocol when less than an f_{AR} fraction of the parties are corrupted and $t_{\mathsf{out}} = \infty$ (since in this case Π_{HBA} guarantees early termination for all honest parties with probability at least p). Thus, by lemma 4.12, when $t_{\mathsf{out}} = \infty$, Π_{HBA} either violates (p, f_{AR})-termination or f_{AV}-validity. However, setting $t_{\mathsf{out}} = \infty$ is equivalent to reducing the real network delays to 0 (or arbitrarily close). Thus, if Π_{HBA} must violate either (p, f_{AR})-responsiveness (if it violates termination when $t_{\mathsf{out}} = \infty$) or f_{AV}-validity otherwise. □

ABA with probabilistic termination In both of our compilers, the termination property of Π_{ABA} may be *probabilistic*, i.e., parties may (all) terminate only with some probability p. In this case, the responsiveness properties for Π_{HBA} are achieved also with probability p, whereas validity, consistency, and termination of Π_{HBA} are preserved.

It is not known how to obtain an Π_{ABA} protocol which terminates for all parties with overwhelming probability, given an Π_{ABA} protocol which terminates only with some constant probability p. Because of this, such protocols have not received much (if any) attention in the literature. However, since termination is one of the hardest properties to achieve in an ABA protocol, it may be much easier to design highly efficient protocols for Π_{ABA} which terminate only with some non-negligible (or constant) probability.

Combined with our compilers, this may lead to very efficient tradeoffs between responsiveness properties and communication efficiency. To the best of our knowledge, this presents the first clear motivation for designing ABA protocols which are not guaranteed to terminate with overwhelming probability.

4.3 A Protocol With Early Termination Support

In this section, we present a second variant of our compiler which offers *early termination*, i.e, responsiveness, under the same conditions in which Π_{HBA} achieves early output. As we will see, our protocol incurs an overhead of $O(n)$ synchronous in the worst case. We first define the notion of *broadcast*.

Definition 4.14 (Broadcast)**.** A distributed protocol Π among n parties $P_1, ..., P_n$ where a designated sender P_s initially holds input v achieves *broadcast* if the following two properties are satisfied at the end of the protocol.

- Termination: Every honest party terminates the protocol.
- Validity: If P_s is honest upon terminating, every honest party outputs v.
- Consistency: Every honest party outputs the same value v'.

For this subsection, we make use of an additional protocol Π_{SBC} which has the following properties:

- Π_{SBC} achieves broadcast with honest termination validity for any number $f < n$ of dishonest parties.
- In the first round of Π_{SBC}, only the sender sends a message.
- The sender in Π_{SBC} terminates directly after sending its first message.
- The protocol is secure against a *rushing* adversary (who receives all messages sent in a round before sending its own messages).

The variant of the classical Dolev-Strong protocol [18] that appears in the thesis of Kumaresan [23] satisfies the aforementioned properties. It was shown in [21] that any broadcast protocol with the above properties runs in $O(f)$ rounds in the worst-case. We note that if Π_{SBC} is meant to be executed at t_{out}, the sender can send its first message at any time $t' \leq t_{\mathsf{out}}$, since this message will be received by all honest parties by the end of the first round, and Π_{SBC} is secure against a rushing adversary. Moreover, the sender can terminate immediately after sending its message, since Π_{SBC} specifies that the sender terminates after sending a message in the first round.

Definition 4.15 (All-to-All Broadcast)**.** A distributed protocol Π among n parties $P_1, ..., P_n$ where party P_i holds input v_i and all parties output a vector $\vec{o} = (o_1, \ldots, o_n)$ achieves *all-to-all broadcast* if the following properties are satisfied.

- Termination: Every honest party terminates the protocol.
- Validity: If P_i is honest on terminating, the output vector of every honest party that did not terminate before giving output satisfies $o_i = v_i$.
- Consistency: Every honest party that did not terminate early outputs the same vector value $\vec{o}$.

In the following, we will denote $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$ as the parallel composition of n independent executions of the protocol Π_{SBC} at time step t_{out}, where for execution i (denoted as Π_{SBC}^{i}), P_i acts as the sender. We will denote the output of $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$ as an n-tuple $(v_1, ..., v_n)$, where v_i denotes the output of the ith run of Π_{SBC}. In Supplementary Material A.1, we prove the UC-Security for any number $t < n$ of malicious parties for Kumaresan's variant of the Dolev-Strong protocol

$\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ with early termination for the sender. By the UC composition theorem, this easily implies that $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$ satisfies definition 4.15.

We now argue that the sender P_i in execution i may send its first message at any time $t' \leq t_{\mathsf{out}}$ and terminate right afterward in $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$, without compromising its security properties. Let $\Pi_{\mathsf{SBC}}^{\mathsf{par\text{-}ea\text{-}}S}$ be the protocol $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$ in which a subset of honest parties $\{P_i\}_{i \in S}$ abort (without output) after sending their first message.

Lemma 4.16. *For every $S \subseteq [n]$, $\Pi_{\mathsf{SBC}}^{\mathsf{par\text{-}ea\text{-}}S}$ satisfies definition 4.15.*

Proof. Let H be the set of all honest parties. To satisfy validity, note that for all $i \in H$, the output of all honest parties in Π_{SBC}^{i} is guaranteed to be v_i even when P_i terminates after sending its first message (by the validity of Π_{SBC} and the composition theorem). Thus, validity (according to definition 4.15) is guaranteed for $\Pi_{\mathsf{SBC}}^{\mathsf{par\text{-}ea\text{-}}S}$ for any set S. Within any of the remaining executions of Π_{SBC} within $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$. Note that in $\Pi_{\mathsf{SBC}}^{j}, j \neq i$, party P_i will be counted as a malicious party if it terminates after its first message. However, since Π_{SBC} tolerates any number of malicious parties $t < n$, all parties in $H \setminus S$ (i.e., all honest parties that give output) are guaranteed to have consistent output on Π_{SBC}^{j}. Thus, by the composition theorem, $\Pi_{\mathsf{SBC}}^{\mathsf{par\text{-}ea\text{-}}S}$ is consistent according to definition 4.15. $\square$

We are now ready to present our second transformation Π_{ETHBA} which is depicted in Figure 4.3.

Figure 4.3: Π_{ETHBA} protocol (view of P_i)

- Let v_i denote the input of P_i
- P_i starts to execution of Π_{ABA} with input v_i.
- Initialize $v^* \leftarrow v_i$.
- Party P_i runs Π_{ABA} until it terminates or until time t_{out} (whichever comes first).
- If party P_i's view of Π_{ABA} has terminated with output v at time $t' < t_{\mathsf{out}}$, it computes a signature $\sigma_i \leftarrow \mathsf{Sign}(v, sk_i)$. It broadcasts (i, v, σ_i) to every party (including itself).
- Upon, receiving at least $\frac{3n}{4}$ valid signatures (from different parties) on a single value v' at time $t' < t_{\mathsf{out}}$, P_i sets $v^* \leftarrow v'$, outputs v^* and broadcasts (i, v^*, L_i) in $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$, where L_i denotes a list containing these signatures. Then it terminates.

---

time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBC}}$ (by shared, global clock)

- At time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBC}}$, if P_i has not terminated and it receives a valid message (j, v, L_j) over $\Pi_{\mathsf{SBC}}^{\mathsf{par}}$, it outputs v and terminates. Otherwise, it participates in a run of Π_{SBA}, using v^* as its input. It outputs the output of Π_{SBA} and terminates.

Lemma 4.17. *Let t_{start}^{i} denote the starting time of party P_i. If*

- $f_{\mathsf{AR}} \leq \frac{1}{4}$,
- *for all honest parties P_i, $t_{\mathsf{out}} - t_{\mathsf{start}}^{i} \geq \Delta$ and*
- $t_{\mathsf{SBA}} \leq f(n)\Delta$.

then Π_{ETHBA} *is* (p, f_{AR})*-responsive.*

Proof. Suppose less than $nf_{\mathsf{AR}} \leq \frac{n}{4}$ parties are dishonest. By the consistency property of Π_{ABA}, every honest party that outputs a value in Π_{ABA}, outputs the same value v. Furthermore, by the p-termination property of Π_{ABA}, with probability at least p, every honest party eventually delivers the value v in Π_{ABA}

Denote t^i_{ABA} the time it took P_i to execute the Π_{ABA} protocol and $t_{\mathsf{ABA}} = \max_i t^i_{\mathsf{ABA}}$ the maximum (over the honest parties) of the time to execute the Π_{ABA} protocol (in executions where every honest party does deliver output). Note that since Π_{ABA} is an asynchronous protocol, t_{ABA} does not depend on Δ. As in lemma 4.3, we consider two cases:

Case 1: For all honest P_i, $t^i_{\mathsf{ABA}} < t_{\mathsf{out}} - t^i_{\mathsf{start}}$. In this case, with probability at least p, every honest party in Π_{ABA} terminates and outputs v. Subsequently, every honest party broadcasts v along with a valid signature. This ensures that with probability at least p, every honest party P_i obtains at least $\frac{3n}{4}$ valid signatures on the value v by time $t_{\mathsf{ABA}} + \delta$. In this case, P_i immediately outputs v and broadcasts a message of the form (i, v, L_i) to every party via $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$. Then, it terminates. Hence, all honest parties receive output and terminate by time $t_{\mathsf{ABA}} + \delta \leq (2 + n + f(n))t_{\mathsf{ABA}}$.

Case 2: There exists an honest P_i s.t. $t^i_{\mathsf{ABA}} \geq t_{\mathsf{out}} - t^i_{\mathsf{start}}$. In this case, at least one honest party did not receive output before time t_{out}. However, in any case, all honest parties are guaranteed to terminate after the the Π_{SBC} and then the Π_{SBA} protocols terminates, thus the total execution time for every honest party is bounded by

$$\begin{aligned}
t_{\mathsf{out}} - t^i_{\mathsf{start}} + \Delta + t_{\mathsf{SBC}} + t_{\mathsf{SBA}} &\leq t_{\mathsf{out}} - t^i_{\mathsf{start}} + \Delta + n\Delta + f(n) \cdot \Delta \\
&= t_{\mathsf{out}} - t^i_{\mathsf{start}} + (1 + n + f(n))\Delta \\
&\leq (2 + n + f(n))(t_{\mathsf{out}} - t^i_{\mathsf{start}}) \leq (2 + n + f(n))t^i_{\mathsf{ABA}} \\
&\leq (2 + n + f(n))t_{\mathsf{ABA}}
\end{aligned}$$

Thus, in both cases, with probability at least p the total execution time is bounded by $(2 + n + f(n))t_{\mathsf{ABA}}$. Since this expression does not depend on Δ, Π_{ETHBA} is (p, f_{AR})-output responsive. □

Lemma 4.18. *Suppose that less than $\frac{n}{2}$ parties are dishonest and suppose that the output for every honest party in $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$ is $\vec{x}$. If for some $i \neq j$, $x_i = (i, v, L_i)$ and $x_j = (j, v', L_j)$ are correctly formed messages, then $v' = v$.*

Proof. This statement can be proved in the same way as Lemma 4.4. □

Lemma 4.19. *Suppose that less than $\frac{n}{2}$ parties are dishonest and let P_i be the first honest party that terminates with output v in Π_{ETHBA} at time $t' < t_{\mathsf{out}}$. Then all honest parties output v in Π_{ETHBA}.*

Proof. Since P_i terminates with output v at time $t' < t_{\mathsf{out}}$, it broadcasts a valid message of the form (i, v, L_i) to all parties via $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$. By the properties of $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$, all honest parties receive this message by time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBC}}$, and output v. Lastly, the value v is unique, as is ensured by Lemma 4.18. Namely, no party P_k can ever collect sufficiently many signatures to correctly form a message (k, v', L_k), s.t. $v' \neq v$. □

Corollary 4.20. Π_{ETHBA} *is* $\frac{1}{2}$*-consistent.*

Proof. Lemma 4.19 ensures consistency in the case where an honest party outputs at time $t' < t_{\mathsf{out}}$. It remains to show that consistency also holds when no honest party outputs before time t_{out}. This can be seen as follows. Either, there is a dishonest party P_i that broadcasts a valid message of the form (i, v, L_i) at time $t' < t_{\mathsf{out}}$ via $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$ by honestly participating in $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$. In this case, lemma 4.18 ensures that no party P_j broadcasts a correctly formed message (j, v', L_j), s.t. $v' \neq v$. Thus, at time $t_{\mathsf{out}} + \Delta + t_{\mathsf{SBC}}$, every honest party outputs v. Otherwise, every honest party outputs the output that it obtains from running Π_{SBA}. Thus, consistency follows from the consistency property of Π_{SBA}. □

Lemma 4.21. *Suppose that less than nf_{AV} parties are dishonest and all honest parties input v to Π_{ETHBA}. Let P_i be an honest party that outputs in Π_{ETHBA} at time $t' < t_{\mathsf{out}}$. Then P_i outputs v.*

Proof. Follows analogously to the proof of lemma 4.7. □

Lemma 4.22. *Suppose that less than nf_{AV} parties are dishonest and all honest parties input v to Π_{ETHBA}. Further, suppose that no honest party outputs in Π_{ETHBA} at time $t' < t_{\mathsf{out}}$. Then every honest party outputs v in Π_{ETHBA} at time $t_{\mathsf{out}} + t_{\mathsf{SBC}} + t_{\mathsf{SBA}} + \Delta$.*

Proof. By validity of Π_{ABA}, every honest party that delivers a value in Π_{ABA}, delivers v. Thus, no honest party P_j signs a message of the form $(j, v', \sigma'_j), v \neq v'$. This ensures that no party will ever see $\frac{3n}{4}$ valid signatures on a value other than v (since less than $nf_{\mathsf{AV}} < \frac{n}{2}$ parties are dishonest). In particular, an honest party P_i will never set v^* to any value other than v, since v^* is initially set to $v_i = v$. This ensures that every honest party inputs v to Π_{SBA} at time $t_{\mathsf{out}} + t_{\mathsf{SBC}} + \Delta$, unless some dishonest party P_k broadcasts a valid message (k, v, L_k) (via an honest participation in $\Pi^{\mathsf{par}}_{\mathsf{SBC}}$). In this case, every honest party is ensured to output v at time $t_{\mathsf{out}} + t_{\mathsf{SBC}} + \Delta$, and thus validity is guaranteed. Otherwise, validity of Π_{SBA} guarantees that every honest party outputs v in Π_{ETHBA} at time $t_{\mathsf{out}} + t_{\mathsf{SBC}} + t_{\mathsf{SBA}} + \Delta$. □

Corollary 4.23. *Π_{ETHBA} is f_{AV}-valid.*

Proof. Follows from lemma 4.21 and lemma 4.22 in the same way that corollary 4.9 follows from lemma 4.7 and lemma 4.8. □

Theorem 4.24. *Assume that:*

- *Π_{ABA} is an asynchronous protocol for byzantine agreement that guarantees validity, consistency, and p-termination if less than nf_{AR} parties are dishonest and satisfies validity if less than nf_{AV} parties are dishonest.*
- *Π_{SBA} is a synchronous protocol for byzantine agreement that guarantees validity and consistency, given that less than $\frac{n}{2}$ parties are corrupted. Π_{SBA} runs in time $t_{\mathsf{SBA}} \leq f(n)\Delta$ for some function f that does not depend on Δ.*
- *$\Pi^{\mathsf{par}}_{\mathsf{SBC}}$ is a synchronous protocol for all-to-all byzantine broadcast that runs in time (at most) t_{SBC}.*
- *$\frac{1}{2} > f_{\mathsf{AV}} \geq f_{\mathsf{AR}}$.*

Then the following statements are true:

- *If $f_{\mathsf{AR}} \leq \frac{1}{4}$ and for all honest P_i, $t_{\mathsf{out}} - t^i_{\mathsf{start}} > \Delta$ then Π_{HBA} is (p, f_{AR})-responsive.*
- *Π_{ETHBA} is $\frac{n}{2}$-consistent.*
- *Π_{ETHBA} is f_{AV}-valid.*
- *Π_{ETHBA} terminates at time $t \leq t_{\mathsf{out}} + t_{\mathsf{SBC}} + t_{\mathsf{SBA}} + \Delta$.*

5 Sequential Composition of Hybrid BAs

In many cases, a "one-shot" execution of a BA protocol is not sufficient; for example, in a state-machine replication protocol, we generally require many sequential executions of BA protocols, where the inputs to each protocol can depend on the outputs of previous executions. In this section we show how to sequentially compose multiple hybrid protocols for byzantine agreement such that all of the properties of every individual component in the composition, in particular output-responsivess, are preserved. We define a sequence of probabilistic protocols $\Pi^1, ..., \Pi^\ell$ in the following manner:

Definition 5.1 (Sequential Composition)**.** ℓ protocols $\Pi^1, ..., \Pi^\ell$ are said to be *run in sequence* if there exists a set of *input derivation functions* $\{f_{i,r}\}_{i\in[n],r\in[\ell]}$ together with a set of *inputs* $(v_i^1, ..., v_i^\ell)$ for each $i \in [n]$ s.t.:

- P_i runs Π^1 with input v_i^1
- For $r \in [\ell]$, P_i computes the input u_i^r to Π^r as $u_i^r = f_{i,r}(v_i^1, ..., v_i^{r-1}, o_i^1, ..., o_i^{r-1})$ where $o_i^k, k < r$, denotes the output of Π^k.

The resulting protocol is called the *sequential composition* of protocols $\Pi^1, ..., \Pi^\ell$.

Next, we define output responsiveness for a sequential composition of protocols

Definition 5.2 (Sequential Output Responsiveness)**.** A sequential composition Π of ℓ probabilistic protocols $\Pi^1, ..., \Pi^\ell$ is said to be *sequentially output responsive* if, for every $r \in [\ell]$ and $i \in [n]$, if P_i is honest then the time until P_i receives the output o_i^r in Π does not depend on Δ.

In the remainder of this section, we will study a sequential composition of ℓ hybrid BA protocols. Our protocol achieving this composition is denoted as Π_{SHBA}.

Figure 5.1: $\Pi_{\mathsf{SHBA}}(t_{\mathsf{out}})$ protocol (view of P_i)

- Let t_{sync} be the execution time of the synchronous portion Π_{HBA}, i.e., the time it takes to execute the remainder of Π_{HBA} after it times out.
- Let Π^r_{HBA} denote instance r of Π_{HBA}.
- Let v_i^r denote the input of party P_i at round r.
- For $r = 1$ to ℓ, P_i repeats the following steps:
  - Wait for output from Π^{r-1}_{HBA}.
  - Compute $u_i^r = f_{i,r}(v_i^1, ..., v_i^{r-1}, o_i^1, ..., o_i^{r-1})$ where $o_k, k < r$ denotes P_i's output from Π^k_{HBA}.
  - Begin execution of Π^r_{HBA} with input u_i^r and timeout parameter $t_{\mathsf{out}} + r \cdot t_{\mathsf{sync}}$.

Assume in the following that Π_{HBA} is a hybrid protocol for BA (as defined in the previous sections) and has f_{AR}-output responsiveness, f_{AV}-validity, and $\frac{1}{2}$-consistency, where as before $f_{\mathsf{AR}} \leq \frac{1}{4}$ and $f_{\mathsf{AV}} \leq \frac{1}{2}(1 - f_{\mathsf{AR}})$.

Lemma 5.3. Π_{SHBA} *is* f_{AR}*-output responsive.*

Proof. Note that a party runs any subprotocol $\Pi^1_{\mathsf{HBA}}, ..., \Pi^\ell_{\mathsf{HBA}}$ as soon as it finishes computing its input. Since the input to Π^k_{HBA} depends only on outputs of protocols $\Pi^1_{\mathsf{HBA}}, ..., \Pi^{k-1}_{\mathsf{HBA}}$, f_{AR}-output responsiveness of Π_{SHBA} follows directly from f_{AR}-output responsiveness of Π_{HBA}. □

Lemma 5.4. *All honest parties start execution of* Π^r_{HBA} *by time* $t_{\mathsf{out}} + r \cdot t_{\mathsf{sync}}$.

Proof. We argue by induction over r. Clearly, the claim holds true for $r = 1$, since u^1_i depends only on v^1_i and can therefore be immediately computed upon start of the protocol. Now suppose the claim holds for $r-1$. Then P_i can run the synchronous part of Π^{r-1}_{HBA} at time $t_{\mathsf{out}}+(r-1)t_{\mathsf{sync}}$ and obtains o_r at time $t_{\mathsf{out}} + rt_{\mathsf{sync}}$. From this it can immediately compute v^r_i. Therefore, u^r_i is well defined for every honest party by time $t_{\mathsf{out}} + rt_{\mathsf{sync}}$ (hence it will start execution). □

Correctness To argue correctness, we generalize BA consistency and validity to a sequence of executions:

- Consistency: For every $r \in [\ell]$, if every honest party outputs the same value o^r_i with overwhelming probability.
- Validity: For every $r \in [\ell]$, if every honest party had the same input u^r_i, then $o^r_i = u^r_i$ with overwhelming probability.

Since we assume the Π_{HBA} protocols remain secure under composition, these properties follow immediately from the security of the Π_{HBA} protocols.

6 An Efficient Common-Coin Protocol with Adaptive Security

In this section, we present a new, efficient common-coin protocol with security against adaptive adversaries. Coupling the BABA protocol from [29] with our common-coin protocol from Section 6.6, we can obtain a new protocol for BABA with security against adaptive adversaries that may corrupt at most $f < \frac{n}{3}$ parties. The message- and communication complexities of this protocol are $O(n^2)$. Notably, this complexity matches the best known algorithms for the static case as well as the complexity for the best known *synchronous* BA protocols. Therefore, the common-coin protocol in this section is well-motivated by the generic compilers from the previous sections. Namely, it leads to a best-of-both worlds protocol Π_{HBA} (or Π_{ETHBA}) also in terms of communication complexity. Previously, for the case of adaptive corruptions, the most efficient solution due to [9] achieved only an impractical communication complexity of $O(\kappa n^4)$ (for a security parameter κ). At a technical level, our contribution consists mainly of the simple observation that the threshold signature scheme from [26] satisfies the uniqueness property needed for the common-coin construction of Cachin et al. [11] and proving this property under the double pairing assumption, which we state below.

BABA serves as a core building block to more complex protocols such as protocols for multi-valued BA [17], asynchronous common subset [7, 10, 12, 28] and state-machine replication (SMR)/atomic broadcast [15, 17, 28]. Many of the these protocols use the statically secure BABA protocol presented in the work of Cachin et al. [11] as a subcomponent due to its low communication complexity. This holds true in particular for the highly efficient SMR protocol presented in [28]. Therefore, the new protocol for BABA immediately implies adaptive security for many of the aforementioned constructions essentially 'for free'.

6.1 Existing Protocols for Asynchronous Byzantine Agreement

The literate of existing protocols for ABA is very rich. We focus on the binary case and restrict our discussion to solutions which handle the maximal corruption bound of $\frac{n}{3}$ corrupted parties and require a polynomial amount of running time. Also, we focus on solutions which do not require set up beyond the assumption of a trusted dealer who distributes public keys to the parties before the protocol.

The problem of BABA was first solved independently by Ben-Or and Rabin [6, 33] (albeit, not with optimal resilience). To circumvent the well-known impossibility for deterministic solutions to the BABA problem by Fischer et al. [27], they were the first to harness the power of randomness in the form of a *common coin*, that is available to all parties, but is not predictable for the adversary. Most subsequent solutions to the BABA problem use the abstraction of a common coin.

The first (polynomial time) protocol to implement a common coin without any set-up assumptions is the beautiful work by Canetti and Rabin [14]. However, their protocol uses an expensive variant of asynchronous verifiable secret sharing (AVSS) which renders their protocol completely impractical. Their common coin construction was subsequently improved by Abraham et al. [2] and Choudhury et al. [32]. However, their solutions are still far beyond the scope of practicality.

Cachin et al. [11] gave the first efficient solution to the BABA problem which has a communication complexity of $O(n^2\ell)$, where ℓ denotes the size of an RSA signature. Their protocol is based on a threshold signature scheme and thus achieves security only against a bounded adversary, whereas the protocols in [14, 2, 32] can tolerate even unbounded adversaries, given that private channels are available for free. As already pointed out earlier, another difference of [11] to the aforementioned protocols is that the latter tolerates only *static corruptions*. The protocol's weakness against adaptive corruptions is inherited from their common coin protocol, which tolerates only static corruptions.

Most recently, Mostéfaoui et al. [29] gave an improvement over the protocol of [11], which reduces the communication complexity to $O(n^2)$ when using the common coin from [11]. In theory, their protocol could also be instantiated with the AVSS-based common coin protocols which would improve its security to the adaptive case. We use this observation and instantiate the common coin in their protocol with an efficient protocol that attains adaptive security and is based on the work of [26]. In this way, we obtain the first adaptively secure BABA protocol which runs in $O(n^2)$ communication complexity.

Using the generic transformation from [17], we immediately obtain an improved protocol for asynchronous multivalued byzantine agreement with optimal resilience tolerating adaptive corruptions. The communication complexity of this protocol is is $O(n^3)$.

6.2 Weak Common Coin Protocols

Definition 6.1 ((p,t)-Weak Common Coin Protocol)**.** A (p,t)-*weak common coin protocol* is a distributed protocol with a subroutine $\mathsf{GetCoin}()$ that takes as input a session identifier $\mathtt{sid}$ and outputs a bit $b \in \{0,1\}$. Furthermore, for any value of $\mathtt{sid}$, it satisfies the following three properties if at most t parties are dishonest. Here, the probability is taken over the random coins of the honest parties.

- Termination: Once every honest party has locally called $\mathsf{GetCoin}(\mathtt{sid})$, the protocol is guaranteed to terminate for every honest party (except with negligible probablity).
- Fairness: Every honest party outputs 0 with probability at least p and 1 with probability at least p.
- Unpredictability: No efficient adversary can predict the outcome of $\mathsf{GetCoin}(\mathtt{sid})$ with probability better than $1-p+\eta$ before the first honest party calls $\mathsf{GetCoin}(\mathtt{sid})$ (where η is a negligible function of the security parameter).

6.3 Random Oracle Model

Our results are stated in the random oracle model [5]. In this model, all hash functions are modeled as an oracle H, which is defined in the following manner. H keeps tracks of all queries that it answers. On input x, H first checks whether $H(x)$ has previously been defined, i.e., whether it has previously answered a query on the value x. If so, it replies with $H(x)$. If not, it samples a value s uniformly at random in the domain of H and returns s.

6.4 Pairing Groups

Let $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ be cyclic groups of prime order p with generators $g, \hat{g}$, and g_T, respectively. We assume a bilinear map $e \colon \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. For this work, we assume a type 3 setting, i.e., there is no efficiently computable isomorphism that maps from $\hat{\mathbb{G}}$ to $\mathbb{G}$. We use the following hardness assumptions.

Definition 6.2 (Decision Diffie-Hellman Assumption)**.** We say that the *Decision Diffie-Hellman Assumption* (DDH) holds with respect to $\mathbb{G}$, if every efficient adversary A has negligible advantage in the distinguishing the distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^{ab}), where $a, b, c \leftarrow \mathbb{Z}$.

In the type 3 setting, we can also make the following stronger assumption, which states that the DDH assumptions holds for both $\mathbb{G}$ and $\mathbb{G}_T$.

Definition 6.3 (Symmetric eXternal Diffie-Hellman Assumption)**.** We say that the *Symmetric eXternal Diffie-Hellman Problem* (SXDH) holds with respect to $\mathbb{G}$ and $\hat{\mathbb{G}}$, if the DDH problem is hard in both $\mathbb{G}$ and $\hat{\mathbb{G}}$.

For convenience, we also state the so-called *Double Pairing* (DP) assumption, which is implied by the DDH assumption in group $\mathbb{G}_T$.

Definition 6.4 (Double Pairing Assumption.)**.** We say that the *Double Pairing Assumption* (DP) holds with respect to $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$, if given $(\hat{g}_z, \hat{g}_r) \leftarrow \hat{\mathbb{G}}^2$, every efficient algorithm A has negligible success probability in finding a non-trivial pair $(z, r) \notin \mathbb{G}^2 \setminus \{(1_{\mathbb{G}}, 1_{\mathbb{G}})\}$ such that $e(z, \hat{g}_z)e(r, \hat{g}_r) = 1_T$.

6.5 Threshold Signature Schemes

In this subsection, we formally introduce (non-interactive) threshold signature schemes along with their security properties. We implicitly assume a message space $\mathcal{M}$ and a signature space $\mathcal{S}$.

Definition 6.5 (Threshold Signature Scheme)**.** Let $0 \leq t \leq n$. A (t, n)*-non-interactive threshold signature scheme* is a tuple of efficient algorithms

$$\mathsf{Sig} = (\mathsf{KeyGen}_{\mathsf{TS}}, \mathsf{Sign}_{\mathsf{TS}}, \mathsf{ShareVerify}_{\mathsf{TS}}, \mathsf{Verify}_{\mathsf{TS}}, \mathsf{Combine}_{\mathsf{TS}})$$

with the following properties.

- The randomized *key generation* algorithm $\mathsf{KeyGen}_{\mathsf{TS}}$ takes a security parameter λ and outputs a tuple $(sk_1, ..., sk_n)$ of secret keys, a tuple $(pk_1, ..., pk_n)$ of public keys, and a special public key pk.
- The deterministic *signing* algorithm $\mathsf{Sign}_{\mathsf{TS}}$ takes as input a secret key sk_i and message $m \in \mathcal{M}$. It outputs a signature share σ_i on m.
- The deterministic *share verification* algorithm takes as input a public key pk_i, a signature share σ_i and a tuple (i, m), where $i \in [n]$. It outputs a bit $b \in \{0, 1\}$, indicating whether σ_i is a valid signature share on m under secret key sk_i. We assume *correctness*, i.e., for all tuples $(pk_1, ...pk_n)$ and $(sk_1, ..., sk_n)$ output by $\mathsf{KeyGen}_{\mathsf{TS}}$, all $m \in \mathcal{M}$, and all $i \in [n]$, we have that $\mathsf{ShareVerify}_{\mathsf{TS}}(pk_i, \mathsf{Sign}_{\mathsf{TS}}(sk_i, m), i, m) = 1$.
- The deterministic *combining* algorithm $\mathsf{Combine}_{\mathsf{TS}}$ takes as input a tuple of public keys $(pk_1, ...pk_n)$, a message m, and a list of pairs $\{(i, \sigma_i)\}_{i \in S}$, where $S \subset [n]$ is of size $t + 1$. It outputs either a signature σ on m or $\perp$, if $\{(i, \sigma_i)\}_{i \in S}$ contains ill-formed signature shares. We will omit the public keys in the input to $\mathsf{Combine}_{\mathsf{TS}}$ when we can ensure that all the shares given as input to it are valid.

- The deterministic *verification* algorithm $\mathsf{Verify_{TS}}$ takes as input a signature σ, a message m and a special public key pk. It outputs a bit $b \in \{0,1\}$ indicating whether σ is a valid signature on m. We again require correctness; for all tuples $(pk_1, ...pk_n)$ and $(sk_1, ..., sk_n)$ output by $\mathsf{KeyGen_{TS}}$, all $m \in \mathcal{M}$, and $\mathcal{S}' = \{(i, \sigma_i)\}_{i \in S}$, where $S \subset [n]$ is of size $t+1$ and $\sigma_i = \mathsf{Sign_{TS}}(sk_i, m)$, we have that $\mathsf{Verify_{TS}}(pk, \mathsf{Combine_{TS}}(\mathcal{S}', (pk_1, ...pk_n), m), m) = 1$.

We next state the definition of unforgeability under chosen message attacks. Our definition is inspired by the work of [26], but instead assumes that the scheme uses a trusted dealer to set up the public key infrastructure, rather than the parties agreeing on the structure in a fully distributed fashion, thus emulating the trusted dealer used in our setting.

Definition 6.6 (Unforgeability Under Chosen Message Attacks)**.** A (t, n)-non-interactive threshold signature scheme satisfies *unforgebility under chosen message attacks* if every efficient algorithm A has negligible advantage in the following game.

- The challenger computes $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk) \leftarrow \mathsf{KeyGen_{TS}}(\lambda)$ and gives $pk_1, ..., pk_n, pk$ to A. Throughout the game, the challenger maintains a list $C \subseteq [n]$.
- A may ask the following two types of queries:
  - Corruption Queries: A submits an index $i \in [n]$ to the challenger. The challenger returns sk_i and sets $C = C \cup \{i\}$.
  - Signing Queries: A submits a pair (i, m) to the challenger. The challenger computes $\sigma_i \leftarrow \mathsf{Sign_{TS}}(sk_i, m)$ and returns σ_i.
- A outputs a pair (m^*, σ^*). Let $S \subset [n]$ be the list of values for which A made a signing query of the form (i, m^*). A wins if $\mathsf{Verify_{TS}}(pk, m^*, \sigma^*) = 1$ and $|S \cup C| \leq t$.

For this work, we will consider the (t, n)-non-interactive threshold signature scheme from [26]. Figure 6.1 presents a simplified version of their scheme which assumes that a trusted dealer computes the secret keys and public keys of the parties.

Lemma 6.7 ([26])**.** *The scheme in Figure 6.1 provides unforgeability against chosen message attacks in the random oracle model and under the SXDH assumption.*

Lemma 6.8. *Suppose that* $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk)$ *are generated as described above. Let* $m \in \mathcal{M}, (h_1, h_2) \leftarrow H(m)$, *and let* $\sigma = (h_1^{-A_1[0]} h_2^{-A_2[0]}, h_1^{-B_1[0]} h_2^{-B_2[0]})$. *If the DP assumptions holds with respect to* $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$, *no efficient algorithm can come up with* $\sigma' \neq \sigma$ *such that* $\mathsf{Verify_{TS}}(\sigma', m, pk) = 1$ *with non-negligible probability, even when given* $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk)$.

Proof. Let A be an algorithm that, with non-negligible probability, on input $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk)$ outputs $\sigma' \neq \sigma$ such that $\mathsf{Verify_{TS}}(\sigma', m, pk) = 1$. We show how to construct an equally efficient algorithm B that breaks the DP assumption. On input $\hat{g}_z, \hat{g}_r \leftarrow \hat{\mathbb{G}}$, B works as follows. It simulates $\mathsf{KeyGen_{TS}}$ using the values $\hat{g}_z, \hat{g}_r$ for its simulation. At the end of the simulation, it gives $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk)$ to A. Clearly, this simulation is perfect since the values $\hat{g}_z, \hat{g}_r$ are uniformly distributed and thus have the same distribution as if they were sampled in $\mathsf{KeyGen_{TS}}$. It simulates the random oracle H to A in the straightforward way. Once A returns σ', B constructs a solution to the DP problem as follows. It parses σ' as $\sigma' = (z', r')$. We write $\sigma = (z, r) = (h_1^{-A_1[0]} h_2^{-A_2[0]}, h_1^{-B_1[0]} h_2^{-B_2[0]})$. W.l.o.g. assume that $z \neq z'$. This implies that $z' h_1^{A_1[0]} h_2^{A_2[0]} \neq 1_{\mathbb{G}}$. On the other hand, $\mathsf{Verify_{TS}}(\sigma', m, pk) = e(z', \hat{g}_z) e(r', \hat{g}_r) \prod_{k=1}^{2} e(h_k, \hat{g}_k) = e(z', \hat{g}_z) e(r', \hat{g}_r) \prod_{k=1}^{2} e(h_k, \hat{g}_z^{A_k[0]} \hat{g}_r^{B_k[0]})$. Thus, expanding terms yields

$$\begin{aligned} \mathsf{Verify_{TS}}(\sigma', m, pk) &= e(z', \hat{g}_z) e(r', \hat{g}_r) \prod_{k=1}^{2} e(h_k, \hat{g}_z^{A_k[0]} \hat{g}_r^{B_k[0]}) && (6.1) \\ &= e(z' h_1^{A_1[0]} h_2^{A_2[0]}, \hat{g}_z) e(r' h_1^{B_1[0]} h_2^{B_2[0]}, \hat{g}_r) = 1_T. && (6.2) \end{aligned}$$

Figure 6.1: LJY Threshold Signature Scheme

- $\mathsf{KeyGen}_{\mathsf{TS}}(\lambda)$: Choose bilinear groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order $p > 2^\lambda$. Sample $\hat{g}_z, \hat{g}_r \leftarrow \hat{\mathbb{G}}$ and values $a_{ik}, b_{ik} \leftarrow \mathbb{F}_p$ where $i \in \{0, ..., n\}$ and $k \in \{1, 2\}$. For $k \in [2]$, set $A_k[X] = \sum_{i=0}^t a_{ik} X^i$ and $B_k[X] = \sum_{i=0}^t b_{ik} X^i$. Compute $sk_i = \{(A_k[i], B_k[i])\}_{k=1}^2$ and $pk_i = (\hat{g}_z^{A_1[i]} \hat{g}_r^{B_1[i]}, \hat{g}_z^{A_2[i]} \hat{g}_r^{B_2[i]})$. Compute $\{\hat{g}_k\}_{k=1}^2$ as $\hat{g}_k = \hat{g}_z^{A_k[0]} \hat{g}_r^{B_k[0]}$ and set $pk = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, \hat{g}_z, \hat{g}_r, \hat{g}_1, \hat{g}_2)$.

- $\mathsf{Sign}_{\mathsf{TS}}(sk_i, m)$: Compute $(h_1, h_2) \leftarrow H(m) \in \mathbb{G}^2$. Use $sk_i = \{(A_k[i], B_k[i])\}_{k=1}^2$ to compute $(z_i, r_i) \in \mathbb{G}^2$ as $z_i = \prod_{k=1}^2 h_k^{-A_k[i]}, r_i = \prod_{k=1}^2 h_k^{-B_k[i]}$

- $\mathsf{ShareVerify}_{\mathsf{TS}}(pk_i, m, \sigma_i)$: Parse σ_i as $\sigma_i = (z_i, r_i)$ and pk_i as $pk_i = (\hat{v}_{1,i}, \hat{v}_{2,i})$. Compute $(h_1, h_2) \leftarrow H(m) \in \mathbb{G}^2$. Return 1 if $e(z_i, \hat{g}_z) e(r_i, \hat{g}_r) \prod_{k=1}^2 e(h_k, \hat{v}_{k,i}) = 1_T$.

- $\mathsf{Combine}_{\mathsf{TS}}(m, pk_1, ..., pk_n, \{(i, \sigma_i)\}_{i \in S})$: For each pair (i, σ_i), run $\mathsf{ShareVerify}_{\mathsf{TS}}(pk_i, m, \sigma_i)$. Return $\perp$ if $|S| \leq t + 1$ or for less than $t + 1$ values of i, $\mathsf{ShareVerify}_{\mathsf{TS}}(pk_i, m, \sigma_i) = 0$. Otherwise, parse σ_i as $\sigma_i = (z_i, r_i) \in \mathbb{G}^2$ and compute $(z, r) = (\prod_{i \in S} z_i^{\Delta_{i,S}(0)}, \prod_{i \in S} r_i^{\Delta_{i,S}[])})$ by using Lagrange interpolation in the exponent, i.e., $\Delta_{i,S}$ denotes the Lagrange polynomial corresponding to party $i \in S$. Return (z, r).

- $\mathsf{Verify}_{\mathsf{TS}}(pk, m, \sigma)$: Parse σ as $\sigma = (z, r) \in \mathbb{G}^2$. Compute $(h_1, h_2) \leftarrow H(m) \in \mathbb{G}^2$ and return 1 iff $e(z, \hat{g}_z) e(r, \hat{g}_r) e(h_1, \hat{g}_1) e(h_2, \hat{g}_2) = 1_T$.

Thus, $(z' h_1^{A_1[0]} h_2^{A_2[0]}, r' h_1^{B_1[0]} h_2^{B_2[0]})$ is a solution to the DP problem. □

6.6 Putting Things Together: The Common-Coin Protocol

We use the common coin protocol by Cachin et al [11]. The idea of their protocol is very simple. All parties share each others' public keys from the (t, n)-threshold signature scheme described in Figure 6.1. To produce a common coin on `sid`, every party P_i produces locally a signature share $\sigma_i = \mathsf{Sign}_{\mathsf{TS}}(\mathtt{sid}, sk_i)$ and broadcasts it. Once P_i obtains $t + 1$ valid signature shares on `sid`, it uses $\mathsf{Combine}_{\mathsf{TS}}$ to combine them into signature σ. By lemma 6.8, the DP assumption assures that any set of $t+1$ shares uniquely determines σ. The parties can now use the random oracle $H' : \mathbb{G}_T \to \{0, 1\}$ to convert the signature into an unbiased and unpredictable bit b. The coin-tossing protocol is described in Protocol 1.

Lemma 6.9. *For $t < \frac{1}{2}$, Protocol 1 is a $(\frac{1}{2}, t)$-weak common coin protocol under the DP assumption.*

Proof. We prove that Protocol 1 satisfies termination, fairness, and unpredictability. The termination property is easily seen to be true; since $t < \frac{n}{2}$, once every honest party has broadcast its share σ_i on `sid`, every party will eventually receive $t+1$ valid signature shares and thus will terminate the protocol. Fairness is ensured by Lemma 6.8, which can be seen as follows. It is clear that if every honest party obtains the same signature σ on `sid` by combining shares via $\mathsf{Combine}_{\mathsf{TS}}$, then $H'(\sigma)$ is a random bit, where the randomness is over the random coins that determine the secret keys of the honest parties. On the other hand, any efficient adversary that can make two honest parties combine their shares to distinct signatures σ and σ' can clearly be used to break the DP assumption by Lemma 6.8. It remains to argue about unpredictability. This property is ensured by Lemma 6.7. Namely, any (unbounded) adversary has negligible

Protocol 1 Common coin protocol $\mathsf{CoinToss}$ from [11]

procedure $\mathsf{KeyGen}_{\mathsf{Coin}}(\lambda)$ // Execute only once
    $(sk_1, ..., sk_n, pk_1, ..., pk_n, pk) \leftarrow \mathsf{KeyGen}_{\mathsf{TS}}(\lambda)$
    for all $i \in [n]$ **do**
        Send $(pk_1, ..., pk_n, pk, sk_i)$ to P_i
    end for
end procedure

procedure $\mathsf{GetCoin}(\mathtt{sid})$ // For party P_i
    $\sigma_i \leftarrow \mathsf{Sign}_{\mathsf{TS}}(sk_i, \mathtt{sid})$
    Broadcast σ_i
    upon receiving a set S of $t+1$ valid signature shares on $\mathtt{sid}$
        Compute $\sigma \leftarrow \mathsf{Combine}_{\mathsf{TS}}(pk, \mathtt{sid}, S)$
    return $H'(\sigma)$
end procedure

advantage in predicting the value of $H'(\sigma)$ if it does not query H' on σ. However, for any efficient adversary that controls at most t parties, Lemma 6.7 ensures that it is computationally infeasible to come up with the value of σ given $\mathtt{sid}$, before the first honest party P_i broadcasts its signature share $\sigma_i = \mathsf{Sign}_{\mathsf{TS}}(sk_i, \mathtt{sid})$ and if the SXDH assumption holds. This concludes the proof. $\square$

References

[1] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Efficient synchronous byzantine consensus. Cryptology ePrint Archive, Report 2017/307, 2017. `https://eprint.iacr.org/2017/307`.

[2] I. Abraham, D. Dolev, and J. Y. Halpern. An almost-surely terminating polynomial protocol forasynchronous byzantine agreement with optimal resilience. In R. A. Bazzi and B. Patt-Shamir, editors, *27th ACM PODC*, pages 405–414. ACM, Aug. 2008.

[3] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage. Steward: Scaling byzantine fault-tolerant replication to wide area networks. In *IEEE Transactions on Dependable and Secure Computing.*

[4] Z. Beerliová-Trubíniová, M. Hirt, and J. B. Nielsen. On the theoretical gap between synchronous and asynchronous MPC protocols. In A. W. Richa and R. Guerraoui, editors, *29th ACM PODC*, pages 211–218. ACM, July 2010.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.

[6] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In R. L. Probert, N. A. Lynch, and N. Santoro, editors, *2nd ACM PODC*, pages 27–30. ACM, Aug. 1983.

[7] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In J. Anderson and S. Toueg, editors, *13th ACM PODC*, pages 183–192. ACM, Aug. 1994.

[8] G. Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75:130–143, 1987.

[9] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In V. Atluri, editor, *ACM CCS 02*, pages 88–97. ACM Press, Nov. 2002.

[10] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 524–541. Springer, Heidelberg, Aug. 2001.

[11] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, July 2005.

[12] C. Cachin and J. Poritz. Secure intrusion-tolerant replication on the internet. In *DSN*, 2002.

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

[14] R. Canetti and T. Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *25th ACM STOC*, pages 42–51. ACM Press, May 1993.

[15] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186, 1999.

[16] J. Chen, S. Gorbunov, S. Micali, and G. Vlachos. Algorand agreement super fast and parition resilient byzantine agreement. *EPRINT*, 2018.

[17] M. Correia, N. F. Neves, and P. Verissimo. From consensus to atomic broadcast: Time-free byzantine-resistant protocols without signatures. *The Computer Journal*, 2006.

[18] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[19] M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-threshold broadcast and detectable multi-party computation. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 51–67. Springer, Heidelberg, May 2003.

[20] M. Fitzi and J. B. Nielsen. On the number of synchronous rounds required for byzantine agreement. In *DISC*, 2009.

[21] J. A. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, Oct. 2007.

[22] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: speculative byzantine fault tolerance. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 45–48, 2007.

[23] R. Kumaresan. *Broadcast and verifiable secret sharing: New security models and round optimal constructions.* PhD thesis, 2012.

[24] K. Kursawe. Optimistic byzantine agreement. In *SRDS*, pages 262–267, October 2002.

[25] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(382–401), July 1982.

[26] B. Libert, M. Joye, and M. Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In M. M. Halldórsson and S. Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.

[27] M.Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[28] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of BFT protocols. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 31–42. ACM Press, Oct. 2016.

[29] A. Mostéfaoui, M. Hamouma, and M. Raynal. Signature-free asynchronous byzantine consensus with $t < n/3$ and $O(n^2)$ messages. In M. M. Halldórsson and S. Dolev, editors, *33rd ACM PODC*, pages 2–9. ACM, July 2014.

[30] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, 2017.

[31] R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In *EUROCRYPT*, 2018.

[32] A. Patra, A. Choudhary, and C. P. Rangan. Simple and efficient asynchronous byzantine agreement with optimal resilience. In S. Tirthapura and L. Alvisi, editors, *28th ACM PODC*, pages 92–101. ACM, Aug. 2009.

[33] M. O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409. IEEE Computer Society Press, Nov. 1983.

[34] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung. Spin one's wheels? byzantine fault tolerance with a spinning primary. In *SRDS*, 2009.

[35] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung. Ebawa: Efficient byzantine agreement for wide-area networks. In *HASE*, pages 10–19, 2010.

A Supplementary Material

A.1 UC Security of Dolev-Strong Protocol With Early Termination

Figure A.1: Functionality $\mathcal{F}_{wbc}$ from [23]

$\mathcal{F}_{wbc}$ interacts with an adversary S and a set of parties $\{P_1, ..., P_n\}$ and a designated sender P_s.

1. **upon** receiving $(\texttt{bcast}, \texttt{sid}, v)$ from P_s, send $(\texttt{bcast}, \texttt{sid}, P_s, v)$ to S.
2. **upon** receiving v' from S:
   - If P_s is corrupted, broadcast $(\texttt{bcast}, \texttt{sid}, P_s, v')$.
   - Otherwise, broadcast $(\texttt{bcast}, \texttt{sid}, P_s, v)$.

Lemma A.1. *For $t < n$ dishonest parties, $\Pi^{\mathsf{DS,t}}_{\mathsf{SBC}}$ securely realizes $\mathcal{F}_{wbc}$ in the $\mathcal{F}_{sig}$-hybrid model, where signatures are replaced by calls to $\mathcal{F}_{sig}$.*

Protocol 2 Dolev-Strong protocol $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ [18] for synchronous byzantine broadcast, adapted from [23]

Let v be the input of dealer P_s and let pk_i, sk_i denote the public/secret key of party i. We say that a set SET is *r-valid* for a value v', if it contains at least r valid signatures from distinct parties on v'. Finally, denote by σ_i^v a (valid) signature on v under public key pk_i.

Broadcast (v, σ_s). Output v and terminate. // Only P_s
Set $\mathsf{ACC}_i = \mathsf{SET}_i = \emptyset$. // Every party P_i
For rounds $r = 1, ..., t$:
upon receiving (v', SET) from P_j, if SET is r-valid, set $\mathsf{ACC}_i \leftarrow \mathsf{ACC}_i \cup \{v'\}, \mathsf{SET}_i \leftarrow \mathsf{SET}_i \cup \mathsf{SET}$.
If a value v' was *newly* added to ACC_i in round $r-1$, broadcast $(v', \mathsf{SET}_i \cup \{\sigma_i^{v'}\}$.
In round $t+1$*:*
if $\mathsf{ACC}_i = \{v'\}$ for some v' **then**
    return v'
end if
Let v' denote the first element in lexicographic order of ACC_i
return v'

Proof. (sketch) Let A be an adversary that interacts with the parties running the protocol $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ in the $\mathcal{F}_{sig}$-hybrid model. We construct a simulator S that runs in the idealized model and interacts with $\mathcal{F}_{wbc}$. We show that no efficient environment Z can distinguish whether it is interacting with A and the parties running $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ in the $\mathcal{F}_{sig}$-hybrid model or S interacting with dummy parties and accessing $\mathcal{F}_{wbc}$. S acts as follows.

1. S waits until either P_s is corrupted or it receives $(\mathtt{bcast}, \mathtt{sid}, P_s, v)$ from $\mathcal{F}_{wbc}$.
2. S simulates the honest parties in $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$. This is easily seen to be possible, because $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ is deterministic and S knows the inputs of P_s (which is the only party with input). Namely, either P_s is corrupted in which case S knows the input of P_s or P_s is honest, and we have argued that S learns the input of P_s from $\mathcal{F}_{wbc}$. Lastly $\mathcal{F}_{sig}$ can efficiently be simulated.
3. If A wishes to corrupt some party P_i, S corrupts P_i and simulates P_i's internal state to A. Again, this can be done efficiently, because $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ is deterministic and parties' internal states are public.
4. Upon completing the simulation of the protocol, suppose that some honest party P_i outputs v' in the simulation of $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$. S now sends v' to $\mathcal{F}_{wbc}$ and terminates.

This simulation is perfect, since S knows the inputs of all honest parties. Secondly, by the consistency property of $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$, every honest party outputs v' at the end of the simulation of $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$. Furthermore, by honest termination validity of $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$, $v' = v$ in the simulation of $\Pi_{\mathsf{SBC}}^{\mathsf{DS,t}}$ if P_s was not corrupt upon terminating. Therefore, the parties in the ideal world will output the same as the parties in the real world. □

A.2 Bracha's Protocol in the Synchronous Model

Consider the reliable broadcast protocol by Bracha [8] depicted in Protocol 3. This protocol runs in a constant number of asynchronous rounds, but we describe in the following an attack strategy of a malicious sender that causes the protocol to run in $\Omega(n)$ synchronous rounds until every party terminates, when the protocol is translated to the synchronous setting naively.

The parties proceed to run the protocol in synchronized rounds of length Δ. In the first round, the sender P_s broadcasts $(\mathtt{send}, m)$ to $\lceil \frac{n+t+1}{2} \rceil - 1$ honest parties. Thus, after Δ time,

Protocol 3 Protocol RBC for reliable broadcast [8] with sender P_s and input m.

```
Broadcast message (send, m) // For party P_s only
upon receiving a message (send, m) from P_s
Broadcast message (echo, m)
upon receiving ⌈(n+t+1)/2⌉ messages (echo, m), if ready message was not previously sent:
Broadcast (ready, m)
upon receiving t + 1 messages (ready, m), if ready message was not previously sent:
Broadcast (ready, m)
upon receiving 2t + 1 messages (ready, m):
Output m and terminate
```

each of these honest parties receives $(\texttt{send}, m)$ and in turn broadcast a total of $\lceil \frac{n+t+1}{2} \rceil - 1$ $(\texttt{echo}, m)$ messages in the second round. Let $P_1, ..., P_{t+1}$ denote some set of t honest parties. In the second round P_s, sends an additional $(\texttt{echo}, m)$ *exclusively* to the party P_{t+1}. Thus, P_{t+1} broadcast $(\texttt{ready}, m)$ at the end of the second round as the only honest party to do so. At the end of round two, the adversary also sends t $(\texttt{ready}, m)$ messages to P_t, $t-1$ such messages to P_{t-1},..., and one such message to P_1. It sends nothing to the other honest parties. Thus, P_t broadcast $(\texttt{ready}, m)$ at the end of the third round, having now received a total of $t+1$ messages of the form $(\texttt{ready}, m)$. This in turn causes party P_{t-1} to receive a total of $t+1$ `ready` messages by the end of the fourth round. Continuing this argument, P_1 receives $t+1$ `ready` messages by the end of round $2+t$ and in turn broadcasts $(\texttt{ready}, m)$. Now, every remaining honest party has received $t+1$ `ready` messages by the end of round $3+t$ and broadcasts $(\texttt{ready}, m)$. Thus, by the end of round $4+t$, every honest party has received $2t+1$ `ready` messages, and the protocol finally terminates.