Perfectly Secure Synchronous MPC with Asynchronous Fallback Guarantees Against General Adversaries

Ananya Appan*, Anirudh Chandramouli**, and Ashish Choudhury***

Abstract. In this work, we study *perfectly-secure multi-party computation* (MPC) against general (*non-threshold*) adversaries. Known protocols in a *synchronous* network are secure against $\mathcal{Q}^{(3)}$ adversary structures, while in an *asynchronous* network, known protocols are secure against $\mathcal{Q}^{(4)}$ adversary structures. A natural question is whether there exists a *single* protocol which remains secure against $\mathcal{Q}^{(3)}$ and $\mathcal{Q}^{(4)}$ adversary structures in a *synchronous* and in an *asynchronous* network respectively, where the parties are *not aware* of the network type. We design the *first* such *best-of-both-worlds* protocol against general adversaries. Our result generalizes the result of Appan, Chandramouli and Choudhury (PODC 2022), which presents a best-of-both-worlds perfectly-secure protocol against *threshold* adversaries.

To design our protocol, we present two important building blocks which are of independent interest. The first building block is a best-of-both-worlds perfectly-secure *Byzantine agreement* (BA) protocol for $\mathcal{Q}^{(3)}$ adversary structures, which remains secure *both* in a synchronous, as well as an asynchronous network. The second building block is a *best-of-both-worlds* perfectly-secure *verifiable secret-sharing* (VSS) protocol, which remains secure against $\mathcal{Q}^{(3)}$ and $\mathcal{Q}^{(4)}$ adversary structures in a *synchronous* network and an *asynchronous* network respectively.

1 Introduction

Secure *multi-party computation* (MPC) [31,21,7] is one of the central pillars in modern cryptography. Informally, an MPC protocol allows a set of mutually distrusting parties, $\mathcal{P} = \{P_1, \ldots, P_n\}$, to securely perform any computation over their private inputs without revealing any additional information about their inputs. In any MPC protocol, the distrust among the parties is modeled by a centralized *adversary* $\mathcal{A}$, who can corrupt and control a subset of the parties during the protocol execution. We consider *computationally unbounded*, Byzantine (malicious) adversaries. This is the most powerful form of corruption where

* The work was done when the author was a student at the International Institute of Information Technology, Bangalore India. Email: `ananya.appan@iiitb.ac.in`

** The work was done when the author was a student at the International Institute of Information Technology, Bangalore India. Email: `anirudh.c@iiitb.ac.in`

*** International Institute of Information Technology, Bangalore India. Email: `ashish.choudhury@iiitb.ac.in`.

$\mathcal{A}$ can force the corrupt parties to behave *arbitrarily* during protocol execution. Security achieved against such an adversary is called *perfect security*.

Traditionally, the *corruption capacity* of $\mathcal{A}$ is modeled through a publicly-known *threshold* t, where it is assumed that $\mathcal{A}$ can corrupt *any* t parties [7,12,30]. A more general form of corruption capacity is the *general-adversary* model (also known as the *non-threshold* setting) [22]. Here, $\mathcal{A}$ is characterized by a publicly-known *adversary structure* $\mathcal{Z} \subset 2^{\mathcal{P}}$, which enumerates *all possible* subsets of potentially corrupt parties, where $\mathcal{A}$ can select any subset from $\mathcal{Z}$ for corruption. Notice that a *threshold* adversary is a *special* type of *non-threshold* adversary, where $\mathcal{Z}$ consists of all subsets of $\mathcal{P}$ of size up to t. It is well-known that modelling $\mathcal{A}$ through $\mathcal{Z}$ allows for more flexibility, especially when $\mathcal{P}$ is small [22,23].

Our Motivation and Results: Traditionally, MPC protocols are designed assuming either a *synchronous* or *asynchronous* communication model. In a *synchronous* MPC (SMPC) protocol, the communication channels between the parties are assumed to be *synchronized*, and every message is assumed to be delivered within some *known* time. Unfortunately, maintaining such time-outs in real-world networks like the Internet is extremely challenging. Asynchronous MPC (AMPC) protocols operate assuming an *asynchronous* communication network, where the channels are not synchronized, and messages can be arbitrarily (yet finitely) delayed. Designing AMPC protocols is *more challenging* when compared to SMPC protocols. This is because, inherently, in *any* AMPC protocol, a receiver party *cannot distinguish* between a *slow sender* party (whose messages are arbitrarily delayed) and a *corrupt sender* party (who does not send any messages). Hence, to avoid an endless wait, no party can afford to receive messages from *all* the parties, as corrupt parties may never send their designated messages. So, at every step in an AMPC protocol, a receiver party can wait for messages from only a "subset" of parties, ignoring messages from the remaining parties which may be *potentially honest*. In fact, in *any* AMPC protocol, it is *impossible* to ensure that the inputs of *all* honest parties are considered for computation, since waiting for the inputs of *all* the parties may turn out to be an endless wait.

Against *threshold* adversaries, perfectly-secure SMPC can tolerate up to $t_s < n/3$ corrupt parties [7]. On the other hand, perfectly-secure AMPC can tolerate up to $t_a < n/4$ corrupt parties [6]. These impossibility results have been generalized to the following bounds against a *non-threshold* adversary: SMPC against a general-adversary is possible provided the underlying *synchronous adversary structure* $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(3)}$ condition [23]. On the other hand, AMPC against a general-adversary is possible provided the underlying *asynchronous adversary structure* $\mathcal{Z}_a$ satisfies the $\mathcal{Q}^{(4)}$ condition [25].[1]

Typically, in any MPC protocol, it is *assumed* that the parties will be knowing whether the underlying network is synchronous or asynchronous *beforehand*. We envision a scenario where the parties are *not aware* of the network type, and aim to design a *single* MPC protocol which offers the best possible security guar-

[1] An adversary structure $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(k)}$ condition if the union of any k subsets from $\mathcal{Z}$ *does not* cover $\mathcal{P}$.

antees, both in the synchronous and the asynchronous communication model. We call such a protocol as a *best-of-both-worlds* MPC protocol. In a recent work, Appan et al. [2] presented a best-of-both-worlds *perfectly-secure* MPC protocol against *threshold* adversaries which could tolerate up to t_s and t_a corruptions in a *synchronous* and *asynchronous* network respectively, for any $t_a < t_s$ where $t_a < n/4$ and $t_s < n/3$, provided $3t_s + t_a < n$ holds. We aim to generalize this result against *general* adversaries, and ask the following question:

Let $\mathcal{A}$ be an adversary, characterized by adversary structures $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in a synchronous network and asynchronous network respectively, where $\mathcal{Z}_s \neq \mathcal{Z}_a$. Then, is there a best-of-both-worlds perfectly-secure MPC protocol which is secure against $\mathcal{A}$, irrespective of the network type?

No prior work has addressed the above question. We present a best-of-both-worlds perfectly-secure MPC protocol, provided $\mathcal{Z}_s, \mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}$ condition and if every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$.[2] Note that we focus on the case where $\mathcal{Z}_s \neq \mathcal{Z}_a$ as otherwise, the question is *trivial* to solve. More specifically, if $\mathcal{Z}_s = \mathcal{Z}_a$, then the necessary condition of AMPC implies that *even* $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(4)}$ condition. Hence, one can use *any* existing perfectly-secure AMPC protocol against general-adversaries (with appropriate time-outs) [25,15,3], which will work *even* in the synchronous network, with the guarantee that the inputs of *all honest* parties are considered for the computation. Our goal, however, is to achieve security against $\mathcal{Q}^{(3)}$ adversary structures, if the underlying network is *synchronous*. For example, let $\mathcal{P} = \{P_1, \ldots, P_8\}$. Consider the adversary structures $\mathcal{Z}_s = \{\{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_3, P_4, P_5\}, \{P_4, P_5, P_6\}, \{P_7\}, \{P_8\}\}$ and $\mathcal{Z}_a = \{\{P_1, P_3\}, \{P_2, P_4\}, \{P_3, P_5\}, \{P_4, P_6\}\}$. Since $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy $\mathcal{Q}^{(3)}$ and $\mathcal{Q}^{(4)}$ conditions respectively, it follows that *existing* SMPC protocols can tolerate $\mathcal{Z}_s$, while existing AMPC protocols can tolerate $\mathcal{Z}_a$. However, we show that *even* if the parties are *not aware* of the exact network type, then using our protocol, one can *still* achieve security against $\mathcal{Z}_s$ if the network is *synchronous* or against $\mathcal{Z}_a$ if the network is *asynchronous*. The above example *also* demonstrates the flexibility offered by the non-threshold adversary model, in terms of tolerating *more* number of faults. More specifically, in the *threshold* model, using the protocol of [2], one can tolerate up to $t_s = 2$ and $t_a = 1$ faults, in a *synchronous* and *asynchronous* network respectively. However, in the *non-threshold* model, our protocol can tolerate subsets of size larger than the maximum allowed t_s and t_a in synchronous and asynchronous network.

Even though our results generalize the results of [2], our protocols are relatively simpler compared to theirs. For instance, one of the main ingredients used in their protocol is a best-of-both-worlds *verifiable secret-sharing* (VSS) protocol. Their VSS is involved and built upon another primitive called *weak polynomial-sharing* (WPS). On the contrary, our best-of-both-worlds VSS protocol is relatively *simpler* and is *not* based on any WPS protocol.

[2] $\mathcal{Z}_s, \mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(k,k')}$ condition if the union of any k and k' subsets from $\mathcal{Z}_s$ and $\mathcal{Z}_a$ respectively *does not* cover $\mathcal{P}$.

1.1 Technical Overview

Like in any generic MPC protocol, we assume that the underlying computation (which the parties want to perform securely) is modelled as some publicly-known function, abstracted by some arithmetic circuit cir, over some algebraic structure $\mathbb{K}$, consisting of linear and non-linear (multiplication) gates. The problem of secure computation then reduces to secure *circuit-evaluation*, where the parties jointly and securely "evaluate" cir in a secret-shared fashion, such that all the values during the circuit-evaluation remain *verifiably secret-shared* and where the shares of the corrupt parties *fail* to reveal the exact underlying value. The secret-sharing used is typically *linear* [16], thus allowing the parties to evaluate the linear gates *locally* (non-interactively). On the other hand, non-linear gates are evaluated by deploying the standard Beaver's method [5] using random, secret-shared *multiplication-triples* which are generated in a circuit-independent *preprocessing phase*. Then, once all the gates are securely evaluated, the parties publicly reconstruct the secret-shared circuit-output. Apart from VSS [13], the parties also need to run instances of a *Byzantine agreement* (BA) protocol [29] to ensure that all the parties are on the "same page" during the various stages of the circuit-evaluation. The above framework for shared circuit-evaluation is defacto used in *all* generic perfectly-secure SMPC and AMPC protocols. Unfortunately, there are several challenges to adapt the framework in our setting, where the parties will be *unaware* of the exact network type.

First Challenge — A Best-of-Both-Worlds BA Protocol: Informally, a BA protocol [29] allows parties with private inputs to reach agreement on a *common* output (*consistency*), such that the output is the input of *honest* parties, if all honest parties have the same input (*validity*). *Perfectly-secure* BA protocols can be designed against $\mathcal{Q}^{(3)}$ adversary structures *irrespective* of the network type [19,14]. However, the *termination* (also called *liveness*) guarantees are *different* for *synchronous* BA (SBA) and *asynchronous* BA (ABA) protocols. The (deterministic) SBA protocols ensure that all honest parties obtain their output after some fixed time (*guaranteed liveness*) [19]. On the other hand, to circumvent the FLP impossibility result [18], ABA protocols are *randomized* and provide *almost-surely liveness* [1,4,14], where the parties terminate the protocol with probability 1, if they keep on running the protocol forever. Known SBA protocols become insecure in an *asynchronous* network, while existing ABA protocols can provide *only* almost-surely liveness in a *synchronous* network.

The *first* challenge to perform shared circuit-evaluation in our setting is to get a best-of-both-worlds BA protocol which provides the security guarantees of SBA and ABA in a *synchronous* and an *asynchronous* network respectively. We are *not* aware of any such BA protocol and hence, present a BA protocol against $\mathcal{Q}^{(3)}$ adversary structures with the above properties. As our BA protocol is technical, we defer its informal discussion to Section 3.

Second Challenge — A Best-of-Both-Worlds VSS Protocol: In a VSS protocol, there exists a *dealer* D with some private input s. The protocol allows D to "verifiably" distribute shares of s to the parties, such that adversary's view remains independent of s, provided D is *honest* (*privacy*). Moreover, in a *synchronous* VSS (SVSS) protocol, every (honest) party obtains its shares after some *known* time-out (*correctness*). The *verifiability* here guarantees that even a *corrupt* D shares some value "consistently" within the known time-out (*commitment* property). Perfectly-secure SVSS against general adversaries is possible, provided the underlying adversary structure $\mathcal{Z}_s$ satisfies $\mathcal{Q}^{(3)}$ condition [27,24].

For an *asynchronous* VSS (AVSS) protocol, the *correctness* guarantees that for an *honest* D, the secret s is eventually secret-shared. However, a *corrupt* D *may not* invoke the protocol in the first place, in which case the honest parties may not obtain any shares. Hence, the *commitment* property of AVSS guarantees that if D is *corrupt and* if some honest party computes a share (implying that D has invoked the protocol), then all honest parties eventually compute their shares. Perfectly-secure AVSS against general adversaries is possible, provided the underlying adversary structure $\mathcal{Z}_a$ satisfies the $\mathcal{Q}^{(4)}$ condition [24,3].

Existing SVSS protocols become completely insecure in an asynchronous network, even if a single expected message from an *honest* party is *delayed*. On the other hand, existing AVSS protocols become insecure against $\mathcal{Q}^{(3)}$ adversary structures (which will be the case, if the network is *synchronous*). Since ,in our setting, the parties will *not* be knowing the exact network type, to maintain *privacy* during the shared circuit-evaluation, we need to ensure that each value remains secret-shared with respect to $\mathcal{Z}_s$ and *not* $\mathcal{Z}_a$, *even* if the network is *asynchronous*.[3] The *second* challenge to perform shared circuit-evaluation in our setting is to get a perfectly-secure VSS protocol which is secure with respect to $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in a *synchronous* and *asynchronous* network respectively, where *privacy always holds* with respect to $\mathcal{Z}_s$, *irrespective* of the network type. We are not aware of any VSS protocol against general adversaries with these guarantees. Hence, we present a best-of-both-worlds perfectly-secure VSS protocol satisfying the above properties. Since our VSS is slightly technical, we defer the informal discussion about the protocol to Section 4.

1.2 Other Related Work

All existing works in the domain of the best-of-both-worlds protocols focus only on *threshold* adversary model. The works of [9] and [11,17] show that the condition $2t_s + t_a < n$ is necessary and sufficient for best-of-both-worlds *cryptographically-secure* BA and MPC respectively, tolerating *computationally bounded* adversaries. Using the same condition, [10] presents a best-of-both-worlds *cryptographically-secure* atomic broadcast protocol. The work of [28] studies Byzantine fault tolerance and state machine replication protocols for multiple thresholds, including t_s and t_a. The work of [20] presents best-of-both-worlds protocol for the task of approximate agreement using the condition $2t_s + t_a < n$.

[3] Since we are assuming that every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$, the privacy will be maintained, both in the synchronous as well as asynchronous network, if each value remains secret-shared with respect to $\mathcal{Z}_s$.

2 Preliminaries and Definitions

The parties in $\mathcal{P}$ are assumed to be connected by pair-wise secure channels. The underlying communication network can be either synchronous or asynchronous, with parties being *unaware* about the exact type. In a *synchronous* network, every sent message is delivered in the same order, within time Δ. In an *asynchronous* network, messages can be delayed arbitrarily, but finitely, and *need not* be delivered in the same order. The only guarantee is that every sent message is *eventually* delivered. The distrust is modeled by a centralized *malicious* (Byzantine) adversary $\mathcal{A}$, who can corrupt a subset of the parties in $\mathcal{P}$ and force them to behave in any arbitrary fashion during the execution of a protocol. The adversary is assumed to be *static*, and decides the set of corrupt parties at the beginning of the protocol execution. The adversary $\mathcal{A}$ is characterized by a *synchronous adversary structure* $\mathcal{Z}_s \subset 2^{\mathcal{P}}$ and an *asynchronous adversary structure* $\mathcal{Z}_a \subset 2^{\mathcal{P}}$. While in a *synchronous* network, $\mathcal{A}$ can corrupt any subset of parties from $\mathcal{Z}_s$, in an *asynchronous* network, $\mathcal{A}$ can corrupt any subset from $\mathcal{Z}_a$.

Given an arbitrary $\mathcal{P}' \subseteq \mathcal{P}$, and an arbitrary adversary structure $\mathcal{Z} \subset 2^{\mathcal{P}}$, we say that $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(k)}(\mathcal{P}, \mathcal{Z})$ condition [22], if the union of *any* k subsets from $\mathcal{Z}$, *does not* cover $\mathcal{P}'$; i.e. for every $Z_{i_1}, \ldots, Z_{i_k} \in \mathcal{Z}$, the condition $\mathcal{P}' \not\subseteq Z_{i_1} \cup \ldots \cup Z_{i_k}$ holds. Given $\mathcal{Z}_s$ and $\mathcal{Z}_a$, we say that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(k,k')}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, if the union of any k subsets from $\mathcal{Z}_s$ and any k' subset from $\mathcal{Z}_a$, *does not* cover $\mathcal{P}$. That is, for every $Z_{i_1}, \ldots, Z_{i_k} \in \mathcal{Z}_s$ and every $Z_{j_1}, \ldots, Z_{j_{k'}} \in \mathcal{Z}_a$, the condition $\mathcal{P} \not\subseteq Z_{i_1} \cup \ldots \cup Z_{i_k} \cup Z_{j_1} \cup \ldots \cup Z_{j_{k'}}$ holds.

We assume that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the following conditions, which we refer throughout the paper as *conditions* Con.

Condition 1 (Con) *$\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the following conditions.*
- *$\mathcal{Z}_s \neq \mathcal{Z}_a$, and $\mathcal{Z}_s, \mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.*
- *For every subset $Z \in \mathcal{Z}_a$, there exists a subset $Z' \in \mathcal{Z}_s$, such that $Z \subseteq Z'$;*

Conditions Con imply that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathcal{Q}^{(4)}(\mathcal{P}, \mathcal{Z}_a)$ conditions respectively. In our VSS and MPC protocols, all computations are done over a finite algebraic structure $(\mathbb{K}, +, \cdot)$, which could be a ring or a field. We assume that each P_i has an input $x_i \in \mathbb{K}$, and parties want to securely compute a function $f : \mathbb{K}^n \rightarrow \mathbb{K}$. Without loss of generality, f is represented by an arithmetic circuit cir over $\mathbb{K}$, consisting of linear and non-linear (multiplication) gates, where cir has c_M multiplication gates and a multiplicative depth of D_M.

Termination Guarantees of Our Sub-Protocols: As done in [2], for simplicity, we will *not* be specifying any *termination* criteria for our sub-protocols. The parties will keep on participating in these sub-protocol instances, *even* after receiving their outputs. The termination criteria of our MPC protocol will ensure the termination of *all* underlying sub-protocol instances. We will be using an existing *randomized* ABA protocol [14] which ensures that the honest parties (eventually) obtain their respective output *almost-surely.* That is:

$$\lim_{T \rightarrow \infty} \Pr[\text{An honest } P_i \text{ obtains its output by local time } T] = 1,$$

where the probability is over the random coins of the honest parties and adversary in the protocol. The property of almost-surely obtaining an output carries over to the "higher" level protocols, where ABA is used as a building block.

We next discuss the syntax and semantics of the secret-sharing, used in our VSS and MPC protocol. The secret-sharing is based on [27], and is defined with respect to a given *sharing specification* $\mathbb{S}$, which is a tuple of subsets of $\mathcal{P}$.

Definition 1 ([27]). *Let $\mathbb{S} = (S_1, \ldots, S_{|\mathbb{S}|})$ be a sharing specification where, for $m = 1, \ldots, |\mathbb{S}|$, each set $S_m \subseteq \mathcal{P}$. Then a value $s \in \mathbb{K}$ is said to be secret-shared with respect to $\mathbb{S}$ if there exist shares $s_1, \ldots, s_{|\mathbb{S}|}$ such that $s = s_1 + \ldots + s_{|\mathbb{S}|}$ and, for $m = 1, \ldots, |\mathbb{S}|$, the share s_m is available with every (honest) party in S_m.*

A secret-sharing of s will be denoted by $[s]$, where $[s]_m$ denotes the m^{th} share. Note that each P_i holds multiple shares $\{[s]_m\}_{P_i \in S_m}$, corresponding to the sets from $\mathbb{S}$ to which it belongs. The above secret-sharing is *linear* as $[c_1 s_1 + c_2 s_2] = c_1[s_1] + c_2[s_2]$ holds for any publicly-known $c_1, c_2 \in \mathbb{K}$. Hence, the parties can *non-interactively* compute any linear function over secret-shared inputs.

For our protocols, we consider the specific sharing specification $\mathbb{S} = (S_1, \ldots, S_q)$, where, for $m = 1, \ldots, q$, the set $S_m \stackrel{def}{=} \mathcal{P} \setminus Z_m$, and where $\mathcal{Z}_s = \{Z_1, \ldots, Z_q\}$ is the *synchronous* adversary structure.

2.1 Existing Primitives

Asynchronous Reliable Broadcast (Acast): An Acast protocol allows a *sender* $\mathsf{S} \in \mathcal{P}$ to send some message $m \in \{0,1\}^\ell$ *identically* to all the parties. The work of [26] presents an Acast protocol against $\mathcal{Q}^{(3)}$ adversary structures by generalizing the classic Bracha's Acast protocol against *threshold* adversaries. While the protocol has been designed for an *asynchronous* network, it also provides certain guarantees in a *synchronous* network, as stated in Lemma 1. The Acast protocol Π_{ACast} and proof of Lemma 1 are available in Appendix A.

Lemma 1. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Then Π_{ACast} achieves the following.*

- *Asynchronous Network:* **(a) $\mathcal{Z}$-Liveness**: *If S is honest, all honest parties eventually have an output.* **(b) $\mathcal{Z}$-Validity**: *If S is honest, then each honest P_i with an output, outputs m.* **(c) $\mathcal{Z}$-Consistency**: *If S is corrupt and some honest P_i outputs $m^\star$, then all honest parties eventually output $m^\star$.*
- *Synchronous Network:* **(a) $\mathcal{Z}$-Liveness**: *If S is honest, then all honest parties obtain an output within time 3Δ.* **(b) $\mathcal{Z}$-Validity**: *If S is honest, then every honest party with an output, outputs m.* **(c) $\mathcal{Z}$-Consistency**: *If S is corrupt and some honest party outputs $m^\star$ at time T, then every honest P_i outputs $m^\star$ by the end of time $T + 2\Delta$.*
- *Communication Complexity: $\mathcal{O}(n^2\ell)$ bits are communicated by the honest parties, where S's message is of size ℓ bits.*

Terminologies for Using Π_{ACast}: We will say that P_i *Acasts* m to mean that P_i acts as a sender S and invokes an instance of Π_{ACast} with input m, and the parties participate in this instance. Similarly, we say that P_j *receives* m *from the Acast of* P_i to mean that P_j outputs m in the corresponding instance of Π_{ACast}.

Public Reconstruction of a Secret-Shared Value: Let $s \in \mathbb{K}$ be a value, which is secret-shared with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Let the goal is to publicly reconstruct s. Since $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ condition, we can use the reconstruction protocol $\Pi_{\mathsf{Rec}}(s, \mathbb{S})$ of [27] which, *irrespective* of the network type, allows the parties to robustly reconstruct s. In a *synchronous* network, the protocol will take Δ time, while in an *asynchronous* network, the parties eventually output s. The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2 \log |\mathbb{K}|)$ bits; see Appendix A for the details.

Beaver's Circuit-Randomization Method [5]: Let u and v be secret-shared among the parties. The goal is to compute a secret-sharing of $w = u \cdot v$. Moreover, let (a, b, c) be a shared *multiplication-triple* available with the parties such that $c = a \cdot b$. Then, Beaver's method allows the parties to compute a secret-sharing of w such that, if a and b are random for the adversary, then the view of the adversary remains independent of u and v. In a *synchronous* network, the parties compute $[w]$ within time Δ, while in an *asynchronous* network, the parties eventually compute $[w]$. Protocol $\Pi_{\mathsf{Beaver}}(([u], [v]), ([a], [b], [c]))$ incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2 \log |\mathbb{K}|)$ bits, and is presented in Appendix A.

3 Best-of-Both-Worlds Byzantine Agreement (BA)

We begin with the definition of BA, which is adapted from [11,2].

Definition 2 (BA). *Let Π be a protocol for $\mathcal{P}$, where every party P_i has an input $b_i \in \{0, 1\}$ and a possible output from $\{0, 1, \bot\}$. Moreover, let $\mathcal{A}$ be an adversary, characterized by adversary structure $\mathcal{Z}$, where $\mathcal{A}$ can corrupt any subset of parties from $\mathcal{Z}$ during the execution of Π.*

- **$\mathcal{Z}$-Guaranteed Liveness**: *Π has $\mathcal{Z}$-guaranteed liveness if all honest parties obtain an output.*
- **$\mathcal{Z}$-Almost-Surely Liveness**: *Π has $\mathcal{Z}$-almost-surely liveness if, almost-surely, all honest parties obtain some output.*
- **$\mathcal{Z}$-Validity**: *Π has $\mathcal{Z}$-validity if the following holds: If all honest parties have input b, then every honest party with an output, outputs b.*
- **$\mathcal{Z}$-Weak Validity**: *Π has $\mathcal{Z}$-weak validity if the following holds: If all honest parties have input b, then every honest party with an output, outputs b or $\bot$.*
- **$\mathcal{Z}$-Consistency**: *Π has $\mathcal{Z}$-consistency if all honest parties with an output, output the same value (which can be $\bot$).*
- **$\mathcal{Z}$-Weak Consistency**: *Π has $\mathcal{Z}$-weak consistency if all honest parties with an output, output either a common $v \in \{0, 1\}$ or $\bot$.*

Π is called a $\mathcal{Z}$-perfectly-secure synchronous BA (SBA) protocol if, in a synchronous network, it has $\mathcal{Z}$-guaranteed liveness, $\mathcal{Z}$-validity, and $\mathcal{Z}$-consistency.

Π is called a $\mathcal{Z}$-perfectly-secure asynchronous BA (ABA) protocol if, in an asynchronous network it has $\mathcal{Z}$-almost-surely liveness, $\mathcal{Z}$-validity and $\mathcal{Z}$-consistency.

To design our BoBW BA protocol, we will be using an *existing* perfectly-secure SBA and a perfectly-secure ABA protocol, whose properties we review next.

Existing SBA and ABA Protocols: We assume the existence of a $\mathcal{Z}$-perfectly-secure SBA protocol Π_{SBA} with $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, which *also* provides $\mathcal{Z}$-guaranteed liveness in an *asynchronous* network.[4]. For the sake of efficiency, we design a candidate for Π_{SBA} by generalizing the simple SBA protocol of [8], which was designed to tolerate $t < n/3$ corruptions. The protocol requires at most $3n$ rounds in a *synchronous* network and hence, within time $T_{\mathsf{SBA}} \stackrel{def}{=} 3n \cdot \Delta$, all honest parties will get an output in a *synchronous* network. The protocol incurs a communication of $\mathcal{O}(n^3\ell)$ bits if the inputs of the parties are of size ℓ bits. To achieve $\mathcal{Z}$-guaranteed liveness in an *asynchronous* network, the parties can run Π_{SBA} till time T_{SBA}, and then output $\perp$ if no "valid" output is computed as per the protocol at time T_{SBA}. This guarantees that even in an *asynchronous* network, all *honest* parties obtain *some* output at local time T_{SBA}. Our Π_{SBA} protocol and the proof of its properties are available in Appendix B.

For the *asynchronous* network, [14] presents a $\mathcal{Z}$-perfectly-secure ABA protocol Π_{ABA}, provided $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Protocol Π_{ABA} has the following properties in the synchronous and asynchronous network.

Lemma 2 ([14]). *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}$, satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Then, there exists a BA protocol Π_{ABA} tolerating $\mathcal{A}$ such that:*

- **Asynchronous Network***: The protocol is a $\mathcal{Z}$-perfectly-secure ABA protocol with the following liveness guarantees.*
  - *If the inputs of all honest parties are the same, then Π_{ABA} achieves $\mathcal{Z}$-guaranteed liveness. Else, Π_{ABA} achieves $\mathcal{Z}$-almost-surely liveness.*
- **Synchronous Network***: The protocol achieves $\mathcal{Z}$-validity, $\mathcal{Z}$-consistency, and the following liveness guarantees.*
  - *If the inputs of all honest parties are the same, then Π_{ABA} achieves $\mathcal{Z}$-guaranteed liveness, and all honest parties obtain their output within time $T_{\mathsf{ABA}} = k \cdot \Delta$, for some constant k.*
  - *Else, Π_{ABA} achieves $\mathcal{Z}$-almost-surely liveness and requires $\mathcal{O}(poly(n) \cdot \Delta)$ expected time to generate the output.*
- **Communication Complexity***: $\mathcal{O}(|\mathcal{Z}| \cdot n^5 \log |\mathbb{F}| + n^6 \log n)$ bits are communicated by the honest parties, if their inputs are the same. Else, Π_{ABA} incurs an expected communication of $\mathcal{O}(|\mathcal{Z}| \cdot n^7 \log |\mathbb{F}| + n^8 \log n)$ bits. Here $\mathbb{F}$ is a finite field such that $|\mathbb{F}| > n$ holds.*

We note that Π_{ABA} *cannot* be considered as a best-of-both-worlds BA protocol. This is because it achieves *$\mathcal{Z}$-guaranteed liveness* in a *synchronous* network *only* when *all* honest parties have the *same* input. If the honest parties start Π_{ABA}

[4] We *do not* require any other property from Π_{SBA} in an *asynchronous* network.

with *different* inputs, then *instead* of guaranteed liveness, the parties may keep on running the protocol forever, *without* obtaining any output (though the probability of this happening is asymptotically 0). We next design a BA protocol which gets rid of this problem, and which is secure in *any* network. To design the protocol, we need a special type of *broadcast* protocol, which we design first.

3.1 Synchronous Broadcast with Asynchronous Guarantees

We begin with the definition of broadcast, adapted from [11,2].

Definition 3 (Broadcast). *Let Π be a protocol, where a sender $\mathsf{S} \in \mathcal{P}$ has input $m \in \{0,1\}^\ell$, and parties obtain a possible output, including $\perp$. Moreover, let $\mathcal{A}$ be an adversary, characterized by adversary structure $\mathcal{Z}$, where $\mathcal{A}$ can corrupt any subset of parties from $\mathcal{Z}$ during the execution of Π.*

- $\mathcal{Z}$**-Liveness**: *Π has $\mathcal{Z}$-liveness if all honest parties obtain some output.*
- $\mathcal{Z}$**-Validity**: *Π has $\mathcal{Z}$-validity if the following holds: if S is honest, then every honest party with an output, outputs m.*
- $\mathcal{Z}$**-Weak Validity**: *Π has $\mathcal{Z}$-weak validity if the following holds: if S is honest, then every honest party with an output, outputs either m or $\perp$.*
- $\mathcal{Z}$**-Consistency**: *Π has $\mathcal{Z}$-consistency if the following holds: if S is corrupt, then every honest party with an output, outputs a common value.*
- $\mathcal{Z}$**-Weak Consistency**: *Π has $\mathcal{Z}$-weak consistency if the following holds: if S is corrupt, then every honest party with an output, outputs a common $m^\star \in \{0,1\}^\ell$ or $\perp$.*

Π is called a $\mathcal{Z}$-perfectly-secure broadcast protocol if it has $\mathcal{Z}$-Liveness, $\mathcal{Z}$-Validity, and $\mathcal{Z}$-Consistency.

We next design a special broadcast protocol Π_{BC}, which is a $\mathcal{Z}$-perfectly-secure broadcast protocol in a *synchronous* network. Additionally, in an *asynchronous* network, the protocol has $\mathcal{Z}$-Liveness, $\mathcal{Z}$-Weak Validity and $\mathcal{Z}$-Weak Consistency. Looking ahead, we will combine the protocols Π_{BC} and Π_{ABA} to get our best-of-both-worlds BA protocol. We note that the existing Acast protocol Π_{ACast} *does not* guarantee the same properties as Π_{BC} since, for a *corrupt* S, there is *no* liveness guarantee (irrespective of the network type). Moreover, in a *synchronous* network, *all* honest parties *may not* obtain an output within the *same* time, if S is *corrupt* (see Lemma 1).[5]

To design Π_{BC} (Fig 1), we generalize an idea used in [2] against *threshold* adversaries. The idea is to carefully "stitch" together protocol Π_{ACast} with the protocol Π_{SBA}. In the protocol, S first Acasts its message. If the network is *synchronous*, then at time 3Δ, all honest parties should have an output. To confirm this, the parties start participating in an instance of Π_{SBA}, where the input of each party is the output that party has obtained from S's Acast instance at time 3Δ. It is possible that a party has no output at time 3Δ (implying that either the network is *asynchronous* or S is *corrupt*), in which case the input of

[5] Looking ahead, this property from Π_{BC} will be crucial when we design our best-of-both-worlds BA protocol.

the party for Π_{SBA} will be $\perp$. Finally, at time $3\Delta + T_{\mathsf{SBA}}$, the parties output $m^\star$, if it has been received from the Acast of S *and* is the output of Π_{SBA} as well, else, they output $\perp$. It is easy to see that the protocol has *guarantees liveness* in *any* network since all parties will have some output at (local) time $3\Delta + T_{\mathsf{SBA}}$. Moreover, in a *synchronous* network, if some *honest* party has an output $m^\star \neq \perp$ at time $3\Delta + T_{\mathsf{SBA}}$, then *all* the honest parties will also have the output $m^\star$ at time $3\Delta + T_{\mathsf{SBA}}$. This is because if any *honest* party obtains an output $m^\star$, then *at least* one *honest* party must have received $m^\star$ from S's Acast by time 3Δ and so by time $3\Delta + T_{\mathsf{SBA}}$, *all* honest parties will receive $m^\star$ from S's Acast.

Eventual Consistency and Validity for Π_{BC} in Asynchronous Network: In Π_{BC}, the parties set a "time-out" of $3\Delta + T_{\mathsf{SBA}}$ to guarantee liveness. However, in this process, the protocol *only* guarantees *weak validity* and *weak consistency* in an *asynchronous* network. This is because some *honest* parties may receive S's message from the Acast of S within time $3\Delta + T_{\mathsf{SBA}}$, while others may fail to do so. The time-out is essential, as we need *liveness* from Π_{BC} (irrespective of the network type) when used later in our best-of-both-worlds BA protocol.

Looking ahead, we will use Π_{BC} in our VSS protocol for broadcasting values. However, the *weak validity* and *weak consistency* properties may lead to a situation where, in an *asynchronous* network, one subset of *honest* parties may output a value different from $\perp$ at the end of the time-out, while others may output $\perp$. For the security of the VSS protocol, we would require the latter category of parties to *eventually* output the common non-$\perp$ value if the parties *continue* participating in Π_{BC}. To achieve this goal, we make a provision in Π_{BC}. Namely, each P_i who outputs $\perp$ at time $3\Delta + T_{\mathsf{SBA}}$ "switches" its output to $m^\star$, if P_i *eventually* receives $m^\star$ from S's Acast. We stress that this switching is *only* for the parties who obtained $\perp$ at time $3\Delta + T_{\mathsf{SBA}}$. To differentiate between the two ways of obtaining output, we use the terms *regular-mode* and *fallback-mode*. Regular-mode consists of the process of deciding the output at time $3\Delta + T_{\mathsf{SBA}}$, while fallback-mode is the process of deciding the output beyond time $3\Delta + T_{\mathsf{SBA}}$.

Protocol Π_{BC}

(Regular Mode)

- On having the input $m \in \{0,1\}^\ell$, sender S Acasts m.
- At time 3Δ, each $P_i \in \mathcal{P}$ participates in an instance of Π_{SBA}, where the input of P_i is $m^\star$ if $m^\star \in \{0,1\}^\ell$ is received from the Acast of S, else the input is $\perp$.
- **(Local Computation)**: At time $3\Delta + T_{\mathsf{SBA}}$, each $P_i \in \mathcal{P}$ does the following.
  - If some $m^\star \in \{0,1\}^\ell$ is received from the Acast of S *and* $m^\star$ is computed as the output during the instance of Π_{SBA}, then output $m^\star$. Else output $\perp$.

(Fallback Mode)

- Every $P_i \in \mathcal{P}$ who has computed the output $\perp$ at time $3\Delta + T_{\mathsf{SBA}}$, changes it to $m^\star$, if $m^\star$ is received by P_i from the Acast of S.

Fig. 1: Synchronous broadcast with asynchronous guarantees.

The properties of Π_{BC}, stated in Theorem 2, are proved in Appendix C.

Theorem 2. *Let $\mathcal{A}$ be an adversary, characterized by $\mathcal{Z}$, satisfying $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Let S has input $m \in \{0,1\}^\ell$ for Π_{BC}. Then Π_{BC} achieves the following, with a communication complexity of $\mathcal{O}(n^3\ell)$ bits, where $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{SBA}}$.*

- *Synchronous network:* **(a) $\mathcal{Z}$-Liveness***: At time T_{BC}, each honest party has an output.* **(b) $\mathcal{Z}$-Validity***: If S is honest, then at time T_{BC}, each honest party outputs m.* **(c) $\mathcal{Z}$-Consistency***: If S is corrupt, then the output of every honest party is the same at time T_{BC}.* **(d) $\mathcal{Z}$-Fallback Consistency***: If S is corrupt, and some honest party outputs $m^\star \neq \bot$ at time T through fallback-mode, then every honest party outputs $m^\star$ by time $T + 2\Delta$.*
- *Asynchronous Network:* **(a) $\mathcal{Z}$-Liveness***: At time T_{BC}, each honest party has an output.* **(b) $\mathcal{Z}$-Weak Validity***: If S is honest, then at time T_{BC}, each honest party outputs m or $\bot$.* **(c) $\mathcal{Z}$-Fallback Validity***: If S is honest, then each honest party with output $\bot$ at time T_{BC}, eventually outputs m through fallback-mode.* **(d) $\mathcal{Z}$-Weak Consistency***: If S is corrupt, then at time T_{BC}, each honest party outputs a common $m^\star \neq \bot$ or $\bot$.* **(e) $\mathcal{Z}$-Fallback Consistency***: If S is corrupt, and some honest party outputs $m^\star \neq \bot$ at time T where $T \geq T_{\mathsf{BC}}$, then each honest party eventually outputs $m^\star$.*

In the rest of the paper, we use the following terminologies while using Π_{BC}.

Terminologies for Π_{BC}: We say that P_i *broadcasts* m to mean that P_i invokes an instance of Π_{BC} as S with input m, and the parties participate in this instance. Similarly, we say that P_j *receives* m *from the broadcast of* P_i *through regular-mode (resp. fallback-mode)*, to mean that P_j has the output m at time T_{BC} (resp. after time T_{BC}) during the instance of Π_{BC}.

3.2 Protocols $\Pi_{\mathsf{BC}} + \Pi_{\mathsf{ABA}} \Rightarrow$ Best-of-Both-Worlds BA

We now combine protocols Π_{BC} and Π_{ABA}, by generalizing the idea used in [2] against *threshold* adversaries. In the protocol, every party first broadcasts its input bit (for the BA protocol) through an instance of Π_{BC}. If the network is *synchronous*, then all honest parties should have received the inputs of all the (honest) sender parties from their broadcasts through regular-mode by time T_{BC}. Consequently, at time T_{BC}, the parties decide an output for *all* the n instances of Π_{BC}. Based on these outputs, the parties decide their respective inputs for the Π_{ABA} protocol. Specifically, if "sufficiently many" outputs from the Π_{BC} instances are found to be *same*, then the parties consider this output value as their input for the Π_{ABA} instance. Else, they stick to their original inputs. The overall output for Π_{BA} is then set to be the output from Π_{ABA}.

Protocol Π_{BA}

- On having input $b_i \in \{0,1\}$, broadcast b_i.
- For $j = 1, \ldots, n$, let $b_i^{(j)} \in \{0,1,\bot\}$ be received from the broadcast of P_j through **regular-mode**. Include P_j to a set $\mathcal{R}$ if $b_i^{(j)} \neq \bot$. Compute the input $v_i^\star$ for an instance of Π_{ABA} as follows.
  - If $\mathcal{P} \setminus \mathcal{R} \in \mathcal{Z}$, then compute $v_i^\star$ as follows.

  - If there exists a subset of parties $\mathcal{R}_i \subseteq \mathcal{R}$, such that $\mathcal{R} \setminus \mathcal{R}_i \in \mathcal{Z}$ and $b_i^{(j)} = b$ for all the parties $P_j \in \mathcal{R}_i$, then set $v_i^\star = b$.[a]
  - Else set $v_i^\star = 1$.
- Else set $v_i^\star = b_i$.
- At time T_{BC}, participate in an instance of Π_{ABA} with input $v_i^\star$. Output the result of Π_{ABA}.

[a] If there are multiple such $\mathcal{R}_i$, then break the tie using some pre-determined rule.

Fig. 2: The best-of-both-worlds BA. The above code is executed by every $P_i \in \mathcal{P}$.

The properties of Π_{BA}, stated in Theorem 3, are proved in Appendix C.

Theorem 3. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Moreover, let Π_{ABA} be an ABA protocol, satisfying the conditions as stated in Lemma 2. Then, Π_{BA} achieves the following.*
- **Synchronous Network**: *The protocol is a $\mathcal{Z}$-perfectly-secure SBA protocol, where all honest parties obtain an output within time $T_{\mathsf{BA}} = T_{\mathsf{BC}} + T_{\mathsf{ABA}}$. The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}| \cdot n^5 \log|\mathbb{F}| + n^6 \log n)$ bits.*
- **Asynchronous Network**: *The protocol is a $\mathcal{Z}$-perfectly-secure ABA protocol, with an expected communication of $\mathcal{O}(|\mathcal{Z}| \cdot n^7 \log|\mathbb{F}| + n^8 \log n)$ bits.*

4 Best-of-Both-Worlds VSS Protocol

We present our best-of-both-worlds VSS protocol Π_{VSS} (Fig 3), assuming that the conditions Con (see Condition 1 in Section 2) hold. In the protocol, there exists a *dealer* $\mathsf{D} \in \mathcal{P}$ with a private input $s \in \mathbb{K}$. The goal is to "verifiably" generate a secret-sharing of s with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$, *irrespective* of the network type. If D is *honest*, then in an *asynchronous* network, s is *eventually* secret-shared. In a *synchronous* network, s is secret-shared after a *fixed* time such that the view of the adversary remains independent of s, *irrespective* of the network type. Note that s is *always* secret-shared with respect to $\mathbb{S}$, which is defined with respect to the *synchronous* adversary structure $\mathcal{Z}_s$, *even* if the network is *asynchronous*.

The *verifiability* of Π_{VSS} guarantees that if D is *corrupt*, then either no honest party obtains any output (this happens if D *does not* invoke the protocol at the first place), or there exists some value $s^\star \in \mathbb{K}$ (which may be different from s) to which D is "committed" and which is secret-shared with respect to $\mathbb{S}$. Note that in the latter case, we cannot bound the time within which $s^\star$ will be secret-shared, even if the network is *synchronous*. This is because a *corrupt* D may delay its messages arbitrarily, and the parties will *not know* the network type.

Protocol Π_{VSS} is obtained by carefully "stitching" together the *synchronous* VSS (SVSS) and *asynchronous* VSS (AVSS) protocols of [27] and [15] respectively. We first explain the idea behind these protocols individually, and then proceed to explain how we stitch them together.

SVSS Against $\mathcal{Q}^{(3)}$ Adversary Structures: The SVSS protocol of [27] is executed in a sequence of *synchronized phases*, and can tolerate $\mathcal{Q}^{(3)}$ adversary structures. Consider an arbitrary adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^{(3)}$ condition, and let $\mathbb{S}_{\mathcal{Z}} = (S_1, \ldots, S_{|\mathcal{Z}|})$ be the sharing specification where $S_m = \mathcal{P} \setminus Z_m$, for $m = 1, \ldots, |\mathcal{Z}|$. To share s, during the *first* phase, D picks a random vector of shares $(s_1, \ldots, s_{|\mathcal{Z}|})$, which sum up to s. Then all the parties in S_m are given the share s_m. To verify if all the (honest) parties in S_m have received the *same* share from D, the parties in S_m perform a *pairwise consistency* check of their supposedly common share during the *second* phase, and publicly broadcast the results during the *third* phase. If any party in S_m publicly complaints for an inconsistency, then during the *fourth* phase, D makes the share s_m corresponding to S_m *public* by broadcasting it. Note that this *does not* violate the privacy for an *honest* D, since a complaint for inconsistency from S_m implies that S_m has at least one *corrupt* party and so, the adversary will already know the value of s_m. If D *does not* "resolve" any complaint during the fourth phase (implying D is *corrupt*), then it is *publicly discarded*, and everyone takes a default sharing of some publicly-known value on the behalf of D. The protocol ensures that by the end of the *fourth* phase, *all honest* parties in S_m have the *same* share, and that the sum of these shares across all the S_m sets is the value shared by D.

AVSS Against $\mathcal{Q}^{(4)}$ Adversary Structures: The AVSS protocol of [15] closely follows the SVSS protocol of [27]. However, the phases are *no longer* synchronized. Moreover, during the pairwise consistency phase, the parties *cannot* afford to wait to know the status of the consistency checks between all pairs of parties, since the potentially *corrupt* parties may *never* respond. Instead, corresponding to every S_m, the parties check for the existence of a set of "core" parties $\mathcal{C}_m \subseteq S_m$, with $S_m \setminus \mathcal{C}_m \in \mathcal{Z}$, who publicly confirmed the receipt of the same share from D. To ensure that all the parties agree on the core sets, D is assigned the task of identifying the core sets and broadcasting them. The protocol proceeds *only* upon the receipt of core sets from D and their verification. While an *honest* D will eventually find and broadcast core sets, a *corrupt* D may *not* do so, in which case the parties obtain no shares. Once the core sets are identified and verified, it guarantees that all the (honest) parties in every core set $\mathcal{C}_m$ have received the same share from D. The goal will then be to ensure that even the (honest) parties "outside" $\mathcal{C}_m$ (namely, the parties in $S_m \setminus \mathcal{C}_m$) get this common share. Since $\mathcal{Z}$ now satisfies the $\mathcal{Q}^{(4)}$ condition, the "majority" of the parties in $\mathcal{C}_m$ are guaranteed to be *honest*. Hence, the parties in $S_m \setminus \mathcal{C}_m$ can "filter" out the common share held by the parties in $\mathcal{C}_m$, by applying the "majority rule" on the shares received from the parties in $\mathcal{C}_m$ during pairwise consistency tests.

Best-of-Both-Worlds VSS Protocol with Conditions Con: In protocol Π_{VSS}, the parties first start executing the steps of the above SVSS protocol, *assuming* a *synchronous* network, where all the instances of broadcast happen by executing an instance of Π_{BC} with respect to the adversary structure $\mathcal{Z}_s$. If indeed the network in *synchronous*, then within time $2\Delta + T_{\mathsf{BC}}$, the results of pairwise consistency tests will be publicly available. Moreover, if any inconsistency is reported, then within time $2\Delta + 2T_{\mathsf{BC}}$, the dealer D should have resolved all those inconsistencies by making public the "disputed" shares. However, unlike the SVSS protocol, the parties *cannot* afford to discard D if it fails to resolve any inconsistency within time $2\Delta+2T_{\mathsf{BC}}$, as the network could be *asynchronous*, and D's responses may be arbitrarily *delayed*, even if D is *honest*. Moreover, in an *asynchronous* network, some honest parties may be seeing the inconsistencies being reported within time $2\Delta + T_{\mathsf{BC}}$ as well as D's responses within time $2\Delta + 2T_{\mathsf{BC}}$, while other honest parties *may not* be seeing these inconsistencies and D's responses within these timeouts. This may result in the *former* set of honest parties considering the shares made public by D, while the latter set of honest parties, thinking that the network is *asynchronous*, wait for core sets of parties to be made public by D (as done in the AVSS). However, this may lead to the *violation* of the *commitment* property in case D is *corrupt*, and network is *asynchronous*. In more detail, consider a set S_m for which pairwise *inconsistency* is reported, and for which D also finds a set of core parties $\mathcal{C}_m$. Then, it might be possible that the parties in $\mathcal{C}_m$ have received the common share s_m from D, but in response to the inconsistencies reported for S_m, dealer D responds with s'_m, where $s'_m \neq s_m$. This will lead to a situation where one set of honest parties (who see inconsistencies and s'_m within the timeout of $2\Delta + T_{\mathsf{BC}}$ and $2\Delta + 2T_{\mathsf{BC}}$ respectively) consider s'_m as the share for S_m, while another set of honest parties, who do not see the inconsistencies and s'_m within the timeout, eventually see $\mathcal{C}_m$ and filter out the share s_m.

To deal with the above problem, apart from resolving the inconsistencies reported for *any* set S_m, the dealer D *also* finds and broadcasts a core set of parties $\mathcal{C}_m$, who have confirmed receiving the same share from D corresponding to *all* the sets S_m, such that $S_m \setminus \mathcal{C}_m \in \mathcal{Z}_s$. *Additionally*, if there is any inconsistency reported for S_m, then *apart* from D, *every* party in S_m *also* makes public its version of the share corresponding to S_m, which it has received from D. Now, at time $2\Delta + 2T_{\mathsf{BC}}$, the parties check if D has broadcasted a core set $\mathcal{C}_m$ for each S_m. Moreover, if any inconsistency has been reported corresponding to S_m, the parties check if "sufficiently many" parties from $\mathcal{C}_m$ have made public the same share, as made public by D. This *prevents* a *corrupt* D from making public a share, which is *different* from the share which it distributed to the parties in $\mathcal{C}_m$.

If the network is *asynchronous*, then different parties may have *different* "opinion" regarding whether D has broadcasted "valid" core sets $\mathcal{C}_m$. Hence, at time $2\Delta + 2T_{\mathsf{BC}}$, the parties run an instance of our Π_{BA} protocol to decide what the case is. If the parties find that D has broadcasted valid core sets $\mathcal{C}_m$ corresponding to each S_m, then the parties in S_m proceed to compute their share as follows: if D has made public the share for S_m in response to any inconsistency, then it is taken as the share for S_m. If no share has been made public for S_m, then the parties check if "sufficiently many" parties have reported the same share during the pairwise consistency test within time 2Δ, which we show should have happened if the network is *synchronous*, and if the parties maintain sufficient timeouts. If none of these conditions hold, then the parties proceed to filter out the common share, held by the parties in $\mathcal{C}_m$, through "majority rule".

If Π_{BA} indicates that D has *not* made public core sets within time $2\Delta + 2T_{\mathsf{BC}}$, then either the network is *asynchronous* or D is *corrupt*. So the parties resort to the steps used in AVSS. Namely, D finds and broadcasts a set of core parties $\mathcal{E}_m$ corresponding to each S_m, where $S_m \setminus \mathcal{E}_m \in \mathcal{Z}_a$. Then, the parties filter out the common share, held by the parties in $\mathcal{E}_m$, through majority rule.

Protocol $\Pi_{\mathsf{VSS}}(\mathsf{D}, s, \mathbb{S})$

Let $\mathbb{S} = (S_1, \ldots, S_m, \ldots, S_q)$, where for $m = 1, \ldots, q$, the set $S_m \stackrel{def}{=} \mathcal{P} \setminus Z_m$, and where $\mathcal{Z}_s = \{Z_1, \ldots, Z_q\}$ is the *synchronous* adversary structure.

- **Phase I — Share Distribution**: D, on having the input s, does the following.
  - Randomly select shares $s^{(1)}, \ldots, s^{(q)} \in \mathbb{K}$ such that $s = s^{(1)} + \ldots + s^{(q)}$. For $m = 1, \ldots, q$, send the share $s^{(m)}$ to every party in the set S_m.
- **Phase II — Pairwise Consistency Checks**: For $m = 1, \ldots, q$, each party $P_i \in S_m$ does the following.
  - Upon receiving $s_i^{(m)}$ from D, wait till the local time becomes a multiple of Δ. Then, send $s_i^{(m)}$ to every party $P_j \in S_m$.
- **Phase III — Broadcasting Results of Pairwise Consistency Checks**: For $m = 1, \ldots, q$, each party P_i in S_m does the following.
  - On receiving $s_j^{(m)}$ from any $P_j \in S_m$, wait till the local time becomes a multiple of Δ. Then, do the following.
    - If a share $s_i^{(m)}$ corresponding to S_m has been received from D, then, broadcast $\texttt{OK}(m, i, j)$ if $s_i^{(m)} = s_j^{(m)}$ holds. Else, broadcast $\texttt{NOK}(m, i)$.
    - If $s_j^{(m)}$ and $s_k^{(m)}$ have been received from any P_j and P_k respectively, belonging to S_m such that $s_j^{(m)} \neq s_k^{(m)}$, then broadcast $\texttt{NOK}(m, i)$.
- **Local Computation — Constructing Consistency Graphs**: Each party $P_i \in \mathcal{P}$ does the following.
  - For $m = 1, \ldots, q$, construct an undirected consistency graph $G_i^{(m)}$ over the parties in S_m, where the edge (P_j, P_k) is included in $G_i^{(m)}$ if P_i has received $\texttt{OK}(m, j, k)$ and $\texttt{OK}(m, k, j)$ from the broadcast of P_j and P_k respectively, either through regular or fallback mode.
- **Phase IV — Resolving Complaints and Broadcasting Core Sets Based On $\mathcal{Z}_s$**: Each $P_i \in \mathcal{P}$ (including D) does the following at time $2\Delta + T_{\mathsf{BC}}$.
  - If $\texttt{NOK}(m, j)$ is received from the broadcast of any $P_j \in S_m$ through regular-mode corresponding to any $m \in \{1, \ldots, q\}$, then do the following:
    - **If** $P_i = \mathsf{D}$: Broadcast $\texttt{Resolve}(m, s^{(m)})$.
    - **If** $P_i \neq \mathsf{D}$: Broadcast $\texttt{Resolve}(m, s_i^{(m)})$, provided $P_i \in S_m$ and P_i has received $s_i^{(m)}$ from D.
  - **(If** $P_i = \mathsf{D}$**)**: For $m = 1, \ldots, q$, check if there exists some $\mathcal{C}_m \subseteq S_m$ which constitutes a clique in the graph $G_{\mathsf{D}}^{(m)}$, such that $S_m \setminus \mathcal{C}_m \in \mathcal{Z}_s$. If $\mathcal{C}_1, \ldots, \mathcal{C}_q$ are found, then broadcast $\mathcal{C}_1, \ldots, \mathcal{C}_q$.
- **Local Computation — Verifying and Accepting Core sets**: Each party $P_i \in \mathcal{P}$ (including D) does the following at time $2\Delta + 2T_{\mathsf{BC}}$.
  - If $\mathcal{C}_1, \ldots, \mathcal{C}_q$ is received from the broadcast of D through the regular mode, then *accept* these sets, if all the following hold.
    - For $m = 1, \ldots, q$, the set $\mathcal{C}_m$ constitutes a clique in the consistency graph $G_i^{(m)}$ at time $2\Delta + T_{\mathsf{BC}}$. In addition, $S_m \setminus \mathcal{C}_m \in \mathcal{Z}_s$.

- For $m = 1, \ldots, q$, if $\texttt{NOK}(m, j)$ was received from the broadcast of any $P_j \in S_m$ through regular mode at time $2\Delta + T_{\mathsf{BC}}$, then the following must hold true at time $2\Delta + 2T_{\mathsf{BC}}$.
  - $\texttt{Resolve}(m, s^{(m)})$ is received from the broadcast of D through regular-mode.
  - $\texttt{Resolve}(m, s^{(m)})$ is received from the broadcast of a set of parties $\mathcal{C}'_m$ through regular-mode, where $\mathcal{C}'_m \subseteq \mathcal{C}_m$, and $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$.

- **Phase V — Deciding Whether Core Sets Based on $\mathcal{Z}_s$ have Been Accepted by Any Honest Party**: At time $2\Delta + 2T_{\mathsf{BC}}$, each $P_i \in \mathcal{P}$ participates in an instance of Π_{BA} with input $b_i = 1$ if it has accepted sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$, else, with input $b_i = 0$, and waits for time T_{BA}.
- **Local Computation — Computing Shares Through Core Sets Based on $\mathcal{Z}_s$**: If the output of Π_{BA} is 1, then each party $P_i \in \mathcal{P}$ does the following.
  - If $\mathcal{C}_1, \ldots, \mathcal{C}_q$ are not received yet, then wait to receive them from the broadcast of D through fallback-mode.
  - For $m = 1, \ldots, q$, compute the share $s_i^{(m)}$ corresponding to S_m as follows, provided $P_i \in S_m$.
    - If, at time $2\Delta + 2T_{\mathsf{BC}}$, $\texttt{Resolve}(m, s^{(m)})$ was received from the broadcast of D and from a subset of parties $\mathcal{C}'_m \subseteq \mathcal{C}_m$ **through regular-mode**, where $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$, then output $s_i^{(m)} = s^{(m)}$.
    - Else, if a common value, say $s^{(m)}$, was received from a subset of parties $\mathcal{C}''_m \subseteq \mathcal{C}_m$ **at time** 2Δ where $\mathcal{C}_m \setminus \mathcal{C}''_m \in \mathcal{Z}_s$, then output $s_i^{(m)} = s^{(m)}$.
    - Else wait till there exists a subset of parties $\mathcal{C}'''_m \subseteq \mathcal{C}_m$ where $\mathcal{C}_m \setminus \mathcal{C}'''_m \in \mathcal{Z}_a$, such that a common value, say $s^{(m)}$, is received from all the parties in $\mathcal{C}'''_m$. Upon finding such a $\mathcal{C}'''_m$, output $s_i^{(m)} = s^{(m)}$.
- **Phase VI — Broadcasting Core Sets Based on $\mathcal{Z}_a$**: If the output of Π_{BA} is 0, then for $m = 1, \ldots, q$, dealer D does the following in its graph $G_{\mathsf{D}}^{(m)}$.
  - Check if there exists a subset of parties $\mathcal{E}_m \subseteq S_m$, which constitutes a clique in the graph $G_{\mathsf{D}}^{(m)}$, such that $S_m \setminus \mathcal{E}_m \in \mathcal{Z}_a$.
  - Upon finding $\mathcal{E}_1, \ldots, \mathcal{E}_q$, broadcast $\mathcal{E}_1, \ldots, \mathcal{E}_q$.
- **Local Computation — Computing Shares Through Core Sets Based on $\mathcal{Z}_a$**: If the output of Π_{BA} is 0, then each party $P_i \in \mathcal{P}$ does the following.
  - Participate in any instance of Π_{BC} invoked by D for broadcasting sets $\mathcal{E}_1, \ldots, \mathcal{E}_q$, only after time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.
  - Wait till sets $\mathcal{E}_1, \ldots, \mathcal{E}_q$ are received from the broadcast of D, and then *accept* these sets if they satisfy the following conditions.
    - For $m = 1, \ldots, q$, the set $\mathcal{E}_m$ constitutes a clique in the graph $G_i^{(m)}$.
    - For $m = 1, \ldots, q$, the condition $S_m \setminus \mathcal{E}_m \in \mathcal{Z}_a$ holds.
  - If $\mathcal{E}_1, \ldots, \mathcal{E}_q$ are accepted, then compute the share $s_i^{(m)}$ corresponding to every S_m where $P_i \in S_m$ as follows.
    - If $P_i \in \mathcal{E}_m$, then output $s_i^{(m)}$ received from D.
    - Else, wait till there exists a subset $\mathcal{E}'_m \subseteq \mathcal{E}_m$ where $\mathcal{E}_m \setminus \mathcal{E}'_m \in \mathcal{Z}_s$, such that there exists a common values, say $s^{(m)}$, received from all the parties in $\mathcal{E}'_m$. Upon finding such an $\mathcal{E}'_m$, output $s_i^{(m)} = s^{(m)}$.

Fig. 3: Best-of-both-worlds VSS protocol for the sharing specification $\mathbb{S}$.

The properties of Π_{VSS}, stated in Theorem 4, are proved in Appendix D.

Theorem 4. *Let $\mathcal{Z}_s$ and $\mathcal{Z}_a$ be adversary structures, satisfying the conditions* Con *(see Section 2). Moreover, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then, Π_{VSS} achieves the following, where* D *has input $s \in \mathbb{K}$ for Π_{VSS}.*

- *If* D *is honest, then the following hold.*
  - $\mathcal{Z}_s$**-correctness***: In a synchronous network, s is secret-shared with respect to $\mathbb{S}$ at time $T_{\mathsf{VSS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.*
  - $\mathcal{Z}_a$**-correctness***: In an asynchronous network, almost-surely, s is eventually secret-shared with respect to $\mathbb{S}$.*
  - **Privacy***: Adversary's view remains independent of s in any network.*
- *If* D *is corrupt, then either no honest party obtains any output or there exists some $s^\star \in \mathbb{K}$, such that the following hold.*[6]
  - $\mathcal{Z}_a$**-commitment***: In an asynchronous network, almost-surely, $s^\star$ is eventually secret-shared with respect to $\mathbb{S}$.*
  - $\mathcal{Z}_s$**-commitment***: In a synchronous network, $s^\star$ is secret-shared with respect to $\mathbb{S}$, such that the following hold.*
    - *If any honest party outputs its shares at time T_{VSS}, then all honest parties output their shares at time T_{VSS}.*
    - *If any honest party outputs its shares at time $T > T_{\mathsf{VSS}}$, then every honest party outputs its shares by time $T + 2\Delta$.*
- **Communication Complexity***: The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^4(\log|\mathbb{K}| + \log|\mathcal{Z}_s| + \log n) + n^5 \log n)$ bits, and invokes one instance of Π_{BA}.*

Π_{VSS} **for** L **Secrets:** If D has L secrets to share, then it can invoke L independent instances of Π_{VSS}. However, instead of computing and broadcasting L number of $\mathcal{C}_1, \ldots, \mathcal{C}_q$ sets, D can compute and broadcast sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$ once, for *all* the L instances of Π_{VSS}.[7] The parties will need to execute a *single* instance of Π_{BA} to decide whether D has broadcasted valid $\mathcal{C}_1, \ldots, \mathcal{C}_q$ sets, corresponding to *all* L instances of Π_{VSS}. The resultant protocol will incur a communication of $\mathcal{O}(L \cdot |\mathcal{Z}_s| \cdot n^4(\log|\mathbb{K}| + \log|\mathcal{Z}_s| + \log n) + n^5 \log n)$ bits and invokes one instance of Π_{BA}. To avoid repetition, we do not provide the formal details.

5 The Preprocessing Phase Protocol

Our protocol for the preprocessing phase allows the parties to generate secret-sharing of c_M number of multiplication-triples, random for the adversary, with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Before discussing the protocol, we discuss two sub-protocols used.

[6] In the *best-of-both-worlds* setting, it is *not* necessary that the honest parties obtain an output within a known time-out in a *synchronous* network for a *corrupt* D (unlike the *commitment* property of traditional SVSS). This is because a *corrupt* D may not invoke the protocol and the parties will *not* be knowing the network type.

[7] Such common $\mathcal{C}_1, \ldots, \mathcal{C}_q$ sets for all the L instances are guaranteed for an *honest* D.

5.1 Agreement on a Common Subset (ACS)

In the protocol Π_{ACS}, there exists a set $\mathcal{Q} \subseteq \mathcal{P}$, such that it will be *guaranteed* that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ *either* satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition *or* $\mathcal{Q}^{(3,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Moreover, each party in $\mathcal{Q}$ will have L values, which it would like to secret-share using Π_{VSS}.[8] As *corrupt* dealers may *not* invoke their instances of Π_{VSS}, the parties can compute outputs from *only* a subset of Π_{VSS} instances corresponding to a subset of parties $\mathcal{Q} \backslash Z$, for some $Z \in \mathcal{Z}_s$ (*even* in a *synchronous* network). However, in an *asynchronous* network, *different* parties may compute outputs from Π_{VSS} instances of *different* subsets of $\mathcal{Q} \backslash Z$ parties, corresponding to *different* $Z \in \mathcal{Z}_s$. Protocol Π_{ACS} allows the parties to agree on a *common* subset $\mathcal{CS}$ of parties, where $\mathcal{Q} \setminus \mathcal{CS} \in \mathcal{Z}_s$, such that *all* honest parties will be able to compute their outputs corresponding to the Π_{VSS} instances of the parties in $\mathcal{CS}$. Moreover, in a *synchronous* network, *all honest* parties from $\mathcal{Q}$ are guaranteed to be present in $\mathcal{CS}$.[9] Protocol Π_{ACS} is obtained by generalizing the ACS protocol of [2], which was designed for *threshold* adversaries. The formal description of the protocol Π_{ACS} and its properties are available in Appendix E.

5.2 The Multiplication Protocol

Protocol Π_{Mult} takes as input the secret-shared pairs of values $\{([a^{(\ell)}], [b^{(\ell)}])\}_{\ell=1,\ldots L}$, and securely generates $\{[c^{(\ell)}]\}_{\ell=1,\ldots,L}$, where $c^{(\ell)} = a^{(\ell)} \cdot b^{(\ell)}$, without revealing any additional information to the adversary. The protocol is obtained by "combining" the *synchronous* multiplication protocol of [27], with the *asynchronous* multiplication protocol of [15], and adapting them to the best-of-both-worlds setting. For simplicity, we discuss the idea of Π_{Mult} for the case when $L = 1$. The modifications for a general L are straight forward.

Let $[a]$ and $[b]$ be the inputs to the protocol. The goal is to securely compute $[c]$, where $c = a \cdot b$. For this, the parties securely compute a secret-sharing of each summand $[a]_l \cdot [b]_m$. A secret-sharing of c can then be obtained by summing the secret-sharing of each summand $[a]_l \cdot [b]_m$. To generate a secret-sharing of the summand $[a]_l \cdot [b]_m$, the parties do the following: let $\mathcal{Q}_{l,m}$ be the set of parties who are guaranteed to have both the shares $[a]_l$, as well as $[b]_m$. Notice that $\mathcal{Q}_{l,m}$ is *not* empty and $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}_{l,m}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, since $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Hence, *irrespective* of the network type, the set $\mathcal{Q}_{l,m}$ is *bound* to have *at least* one honest party. Each party in the set $\mathcal{Q}_{l,m}$ is asked to independently secret-share the summand $[a]_l \cdot [b]_m$, and the parties then agree on a common subset of parties $\mathcal{R}_{l,m}$ from $\mathcal{Q}_{l,m}$, where $\mathcal{Q}_{l,m} \backslash \mathcal{R}_{l,m} \in \mathcal{Z}_s$, which have shared some summand. For this, the parties execute an instance of the Π_{ACS} protocol. The properties of Π_{ACS} guarantees that in a *synchronous* network, *all honest* parties from $\mathcal{Q}_{l,m}$ are present in $\mathcal{R}_{l,m}$. On the other hand, even if the network is *asynchronous*, the set $\mathcal{R}_{l,m}$ is *bound* to have *at*

[8] Looking ahead, in our *preprocessing phase* protocol, $\mathcal{Q} = S_l \cap S_m$ corresponding to some $S_l, S_m \in \mathbb{S}$ and hence, the $\mathcal{Q}^{(1,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition will be satisfied. In our MPC protocol, $\mathcal{Q} = \mathcal{P}$ and hence the $\mathcal{Q}^{(3,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition will be satisfied.

[9] This property will be very crucial in a *synchronous* network.

least one honest party from $\mathcal{Q}_{l,m}$. Hence, *irrespective* of the network type, it will be *guaranteed* that *at least* one party in $\mathcal{R}_{l,m}$ has secret-shared the summand $[a]_l \cdot [b]_m$. However, since the exact identity of the honest parties in $\mathcal{R}_{l,m}$ is *not* known, the parties check if *all* the parties in $\mathcal{R}_{l,m}$ have shared the same summand. The idea here is that if *all* the parties in $\mathcal{R}_{l,m}$ have shared the same summand, then any of these secret-sharings can be taken as a secret-sharing of $[a]_l \cdot [b]_m$. Else, the parties publicly reconstruct the shares $[a]_l$ and $[b]_m$ and compute a default secret-sharing of $[a]_l \cdot [b]_m$. Notice that in the latter case, the privacy of a and b is still preserved, as in this case, the set $\mathcal{R}_{l,m}$ consists of *corrupt* parties, who already know the values of both $[a]_l$ as well as $[b]_m$.

Protocol Π_{Mult} and its properties are available in Appendix E.

5.3 The Preprocessing Phase Protocol

Given protocols Π_{ACS} and Π_{Mult}, the preprocessing phase protocol $\Pi_{\mathsf{PreProcessing}}$ is standard and straight forward. The protocol has two stages. During the *first* stage, the parties securely generate secret-sharing of c_M pairs of random values. For this, the parties run an instance of Π_{ACS}, where the input for each party will be c_M pairs of random values. During the *second* stage, a secret-sharing of the product of each pair is computed securely by executing an instance of Π_{Mult}. Protocol $\Pi_{\mathsf{PreProcessing}}$ and its properties are available in Appendix E.

6 Best-of-both-Worlds Circuit-Evaluation Protocol

Given protocols $\Pi_{\mathsf{PreProcessing}}$ and Π_{ACS}, the circuit-evaluation protocol Π_{CirEval} for evaluating cir is standard and straight forward. Here, we outline the protocol steps and state its properties. We defer the full details to Appendix F. Protocol Π_{CirEval} consists of four phases. In the *first* phase, the parties generate secret-sharing of c_M random multiplication-triples through $\Pi_{\mathsf{PreProcessing}}$. Additionally, they invoke Π_{ACS} to generate secret-sharing of their respective inputs for f, and agree on a *common* subset of parties $\mathcal{CS}$, where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$, such that the inputs of the parties in $\mathcal{CS}$ are secret-shared. The inputs of the remaining parties are set to 0. Note that in a *synchronous* network, *all honest* parties will be in $\mathcal{CS}$. In the *second* phase, the parties securely evaluate each gate in the circuit in a secret-shared fashion, after which the parties *publicly* reconstruct the secret-shared output in the *third* phase. The *last* phase is the *termination phase*, where the parties check whether "sufficiently many" parties have obtained the same output, in which case the parties "safely" take that output and terminate the protocol (and all the underlying sub-protocols).

Theorem 5. *Let $\mathcal{A}$ be an adversary, characterized by adversary structures $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in a synchronous and asynchronous network respectively, satisfying the conditions* Con *(see Condition 1 in Section 2). Moreover, let $f : \mathbb{K}^n \rightarrow \mathbb{K}$ be a function represented by an arithmetic circuit* cir *over $\mathbb{K}$, consisting of c_M number of multiplication gates, with a multiplicative depth of D_M, with each* *party having an input* $x_i \in \mathbb{K}$*. Then,* Π_{CirEval} *incurs a communication of* $\mathcal{O}(c_M \cdot |\mathcal{Z}_s|^3 \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ *bits, invokes* $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ *instances of* Π_{BA}*, and achieves the following.*

- *In a synchronous network, all honest parties output* $y = f(x_1, \ldots, x_n)$ *at time* $(30n + D_M + 6k + 38) \cdot \Delta$*, where* $x_j = 0$ *for every* $P_j \notin \mathcal{CS}$*, such that* $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$*, and every honest party is present in* $\mathcal{CS}$*; here* k *is the constant from Lemma 2, as determined by the protocol* Π_{ABA}*.*
- *In an asynchronous network, almost-surely, the honest parties eventually output* $y = f(x_1, \ldots, x_n)$ *where* $x_j = 0$ *for every* $P_j \notin \mathcal{CS}$ *and where* $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$*.*
- *The view of* $\mathcal{A}$ *remains independent of the inputs of the honest parties in* $\mathcal{CS}$*.*

References

1. I. Abraham, D. Dolev, and J. Y. Halpern. An Almost-surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience. In *PODC*, pages 405–414. ACM, 2008.
2. A. Appan, A. Chandramouli, and A. Choudhury. Perfectly-Secure Synchronous MPC with Asynchronous Fallback Guarantees. In *PODC*, pages 92–102. ACM, 2022.
3. A. Appan, A. Chandramouli, and A. Choudhury. Revisiting the Efficiency of Asynchronous Multi Party Computation Against General Adversaries. *IACR Cryptol. ePrint Arch.*, page 651, 2022.
4. L. Bangalore, A. Choudhury, and A. Patra. The Power of Shunning: Efficient Asynchronous Byzantine Agreement Revisited. *J. ACM*, 67(3):14:1–14:59, 2020.
5. D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.
6. M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pages 1–10. ACM, 1988.
8. P. Berman, J. A. Garay, and K. J. Perry. Towards Optimal Distributed Consensus (Extended Abstract). In *FOCS*, pages 410–415. IEEE Computer Society, 1989.
9. E. Blum, J. Katz, and J. Loss. Synchronous Consensus with Optimal Asynchronous Fallback Guarantees. In *TCC*, volume 11891 of *Lecture Notes in Computer Science*, pages 131–150. Springer, 2019.
10. E. Blum, J. Katz, and J. Loss. Tardigrade: An Atomic Broadcast Protocol for Arbitrary Network Conditions. In *ASIACRYPT*, volume 13091 of *Lecture Notes in Computer Science*, pages 547–572. Springer, 2021.
11. E. Blum, C. L. Zhang, and J. Loss. Always Have a Backup Plan: Fully Secure Synchronous MPC with Asynchronous Fallback. In *CRYPTO*, volume 12171 of *Lecture Notes in Computer Science*, pages 707–731. Springer, 2020.
12. D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *STOC*, pages 11–19. ACM, 1988.
13. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395. IEEE Computer Society, 1985.

14. A. Choudhury. Almost-Surely Terminating Asynchronous Byzantine Agreement Against General Adversaries with Optimal Resilience. Cryptology ePrint Archive, Paper 2022/917, 2022. `https://eprint.iacr.org/2022/917`.
15. A. Choudhury and N. Pappu. Perfectly-Secure Asynchronous MPC for General Adversaries (Extended Abstract). In *INDOCRYPT*, volume 12578 of *Lecture Notes in Computer Science*, pages 786–809. Springer, 2020.
16. R. Cramer, I. Damgård, and U. M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer Verlag, 2000.
17. G. Deligios, M. Hirt, and C. Liu-Zhang. Round-Efficient Byzantine Agreement and Multi-party Computation with Asynchronous Fallback. In *TCC*, volume 13042 of *Lecture Notes in Computer Science*, pages 623–653. Springer, 2021.
18. M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382, 1985.
19. M. Fitzi and U. M. Maurer. Efficient Byzantine Agreement Secure Against General Adversaries. In *DISC*, volume 1499 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 1998.
20. D. Ghinea, C. Liu-Zhang, and R. Wattenhofer. Optimal Synchronous Approximate Agreement with Asynchronous Fallback. In *PODC*, pages 70–80. ACM, 2022.
21. O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.
22. Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC*, pages 25–34. ACM, 1997.
23. Martin Hirt and Ueli Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
24. Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In *ASIACRYPT*, volume 8270 of *Lecture Notes in Computer Science*, pages 181–200. Springer, 2013.
25. M. V. N. Ashwin Kumar, K. Srinathan, and C. Pandu Rangan. Asynchronous Perfectly Secure Computation Tolerating Generalized Adversaries. In *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 497–512. Springer, 2002.
26. K. Kursawe and F. C. Freiling. Byzantine Fault Tolerance on General Hybrid Adversary Structures. Technical Report, RWTH Aachen, 2005.
27. U. M. Maurer. Secure Multi-party Computation Made Simple. In *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2002.
28. A. Momose and L. Ren. Multi-Threshold Byzantine Fault Tolerance. In *CCS*, pages 1686–1699. ACM, 2021.
29. M. C. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.
30. T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pages 73–85. ACM, 1989.
31. A. C. Yao. Protocols for Secure Computations (Extended Abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

Supplementary Material

A Existing Primitives

A.1 Asynchronous Reliable Broadcast with Weak Synchronous Guarantees

Let $\mathcal{Z}$ be an adversary structure such that $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. The Acast protocol Π_{ACast} with respect to $\mathcal{Z}$ is presented in Fig 4. The current description of the protocol is taken from [3].

Protocol $\Pi_{\mathsf{ACast}}(\mathsf{S}, m)$

1. If $P_i = \mathsf{S}$, then send the message $(\texttt{init}, m)$ to all the parties in $\mathcal{P}$.
2. If a message $(\texttt{init}, m)$ is received from S, then send the message $(\texttt{echo}, m)$ to all the parties in $\mathcal{P}$. Execute this step at most once.
3. If a message $(\texttt{echo}, m')$ is received from a set of parties $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}$, then send the message $(\texttt{ready}, m')$ to all the parties.
4. If a message $(\texttt{ready}, m')$ is received from a set of parties C where $C \notin \mathcal{Z}$, then send the message $(\texttt{ready}, m')$ to all the parties in $\mathcal{P}$.
5. If $(\texttt{ready}, m')$ is received from a set of parties $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}$, then output m'.

Fig. 4: The perfectly-secure Acast protocol with respect to an adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. The above code is executed by every $P_i \in \mathcal{P}$

We next prove the properties of the protocol Π_{ACast}. The proofs for the case of asynchronous network are borrowed from [3].

Lemma 1. Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}$, satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, such that $\mathcal{A}$ can corrupt any subset of parties from $\mathcal{Z}$ during the execution of Π_{ACast}. Then Π_{ACast} achieves the following.

- *Asynchronous Network*:
  - **$\mathcal{Z}$-Liveness**: If S is *honest*, then all honest parties eventually obtain an output.
  - **$\mathcal{Z}$-Validity**: If S is *honest*, then every honest party with an output, outputs m.
  - **$\mathcal{Z}$-Consistency**: If S is *corrupt* and some honest party outputs $m^\star$, then every honest party *eventually* outputs $m^\star$.
- *Synchronous Network*:
  - **$\mathcal{Z}$-Liveness**: If S is *honest*, then all honest parties obtain an output within time 3Δ.
  - **$\mathcal{Z}$-Validity**: If S is *honest*, then every honest party with an output, outputs m.
  - **$\mathcal{Z}$-Consistency**: If S is *corrupt* and some honest party outputs $m^\star$ at time T, then every honest P_i outputs $m^\star$ by the end of time $T + 2\Delta$.
- *Communication Complexity*: $\mathcal{O}(n^2\ell)$ bits are communicated, where S's message is of size ℓ bits.

Proof. We first prove the properties assuming an *asynchronous* network. Let $Z_c \in \mathcal{Z}$ be the set of parties *corrupted* by the adversary during the protocol execution, and let $\mathcal{H} \stackrel{def}{=} \mathcal{P} \setminus Z_c$ be the set of *honest* parties. We start with the *validity* property, for which we consider an *honest* S. We show that all honest parties eventually output m. This is because all honest parties complete steps $2-5$ in the protocol even if the corrupt parties do not send their messages. This follows from the fact that the messages of the parties in $\mathcal{H}$ are eventually delivered to all the honest parties, and $\mathcal{P} \setminus \mathcal{H} = Z_c \in \mathcal{Z}$. The parties in Z_c may send `echo` messages for m', where $m' \neq m$. Similarly, the parties in Z_c may send `ready` messages for m', where $m' \neq m$. However, since $Z_c \in \mathcal{Z}$, and since $\mathcal{P} \setminus Z_c = \mathcal{H} \notin \mathcal{Z}$, no honest party ever generates a `ready` message for m', neither in step 3, nor in step 4.[10] This also proves the *liveness* property.

We next prove the *consistency* property for which we consider a *corrupt* S. Let P_h be an *honest* party who outputs $m^\star$. We have to show that all honest parties eventually obtain the output $m^\star$. Since P_h obtained the output $m^\star$, it received `ready` messages for $m^\star$ during step 5 of the protocol from a set of parties $\mathcal{P} \setminus Z$, for some $Z \in \mathcal{Z}$. Let $\mathcal{H}^{(m^\star)}$ be the set of *honest* parties whose `ready` messages are received by P_h during step 5. It is easy to see that $\mathcal{H}^{(m^\star)} \notin \mathcal{Z}$, as otherwise, $\mathcal{Z}$ *does not* satisfy the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, which is a contradiction. The `ready` messages of the parties in $\mathcal{H}^{(m^\star)}$ are eventually delivered to every honest party and hence, *each* honest party (including P_h) eventually executes step 4 and sends a `ready` message for $m^\star$. It follows that the `ready` messages of *all* the parties in $\mathcal{P} \setminus Z_c$ are eventually delivered to every honest party (irrespective of whether the parties in Z_c send all the required messages) guaranteeing that all honest parties eventually obtain *some* output. We wish to show that this output is $m^\star$.

On the contrary, let $P_{h'} \neq P_h$ be an *honest* party who outputs $m^{\star\star} \neq m^\star$. This implies that $P_{h'}$ received `ready` message for $m^{\star\star}$ from at least one *honest* party. From the protocol steps, it follows that an *honest* party generates a `ready` message for some potential m, only if it either receives `echo` messages for m during step 3 from a set of parties $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}$, or `ready` messages for m from a set of parties $C \notin \mathcal{Z}$ during step 4. So, in order that a subset of parties $\mathcal{P} \setminus Z$, for some $Z \in \mathcal{Z}$, eventually generates `ready` messages for some potential m during step 5, it must be the case that some *honest* party has received `echo` messages for m during step 1 from a set of parties $\mathcal{P} \setminus Z'$ for some $Z' \in \mathcal{Z}$, and has generated a `ready` message for m.

Since P_h received the `ready` message for $m^\star$ from at least one honest party, it must be the case that some honest party has received `echo` messages for $m^\star$ from a set of parties $\mathcal{P} \setminus Z_\alpha$, for some $Z_\alpha \in \mathcal{Z}$. Similarly, since $P_{h'}$ received a `ready` message for $m^{\star\star}$ from at least one honest party, it must be the case that some honest party has received `echo` messages for $m^{\star\star}$ from a set of parties $\mathcal{P} \setminus Z_\beta$, for some $Z_\beta \in \mathcal{Z}$. Let $\mathcal{T} = (\mathcal{P} \setminus Z_\alpha) \cap (\mathcal{P} \setminus Z_\beta)$. Since $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, it follows that $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(1)}(\mathcal{T}, \mathcal{Z})$ condition, and hence, $\mathcal{T}$ is guaranteed to have at least one *honest* party. This further implies that there exists some honest party who generated an `echo` message for $m^\star$ as

[10] If $\mathcal{H} \in \mathcal{Z}$, then $\mathcal{Z}$ *does not* satisfy the $\mathcal{Q}^{(2)}(\mathcal{P}, \mathcal{Z})$ condition, which is a contradiction.

well as $m^{\star\star}$ during step 1, which is impossible. This is because an honest party executes step 1 at most once, and hence, generates an `echo` message at most once.

The proofs of the properties in a *synchronous* network closely follow the proofs of the properties in the *asynchronous* network. Abusing the notation, let $Z_c \in \mathcal{Z}$ be the set of *corrupt* parties, and let $\mathcal{H} \stackrel{def}{=} \mathcal{P} \setminus Z_c$ be the set of *honest* parties. If S is *honest*, then its `init` message for m is delivered within time Δ. As a result, every *honest* party sends an `echo` message for m to all the parties, which is delivered within time 2Δ. Consequently, every *honest* party sends a `ready` message for m to all the parties, which is delivered within time 3Δ. Since $\mathcal{P} \setminus \mathcal{H} = Z_c \in \mathcal{Z}$, every honest party will receive the `ready` messages for m from all the parties in $\mathcal{H}$ within time 3Δ and output m. This proves the *liveness* and *validity* in the synchronous network.

If S is *corrupt*, and some honest party P_h outputs $m^\star$ at time T, then it implies that P_h has received `ready` messages for $m^\star$ during step 5 of the protocol at time T from a set of *honest* parties $\mathcal{H}^{(m^\star)}$, such that $\mathcal{H}^{(m^\star)} \notin \mathcal{Z}$. These ready messages are guaranteed to be received by every other honest party within time $T + \Delta$. Consequently, every *honest* party who has not yet executed step 4 will do so, and will send a `ready` message for $m^\star$ at time $T + \Delta$. Hence, by the end of time $T + \Delta$, every honest party would have sent a `ready` message for $m^\star$ to every other honest party, which will be delivered within time $T + 2\Delta$. As a result, every honest party will output $m^\star$ latest at time $T + 2\Delta$. This proves the *consistency* in the synchronous network.

The communication complexity (both in a synchronous as well as asynchronous network) simply follows from the fact that every honest party may need to send an `echo` and `ready` message for m to every other party.

A.2 Reconstruction Protocol

Let $s \in \mathbb{K}$ be a value, which is secret-shared with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then protocol Π_{Rec} (Fig 5) allows the parties to publicly reconstruct s.

Protocol $\Pi_{\mathsf{Rec}}(s, \mathbb{S})$

Let $\mathbb{S} = (S_1, \ldots, S_m, \ldots, S_q)$, where for $m = 1, \ldots, q$, the set $S_m \stackrel{def}{=} \mathcal{P} \setminus Z_m$ and where $\mathcal{Z}_s = \{Z_1, \ldots, Z_q\}$ is the *synchronous* adversary structure.

- Corresponding to every $S_m \in \mathbb{S}$, each $P_i \in S_m$ sends the share $[s]_m$ to all the parties in $\mathcal{P}$.
- Each $P_i \in \mathcal{P}$ computes its output as follows.
  - Corresponding to every $S_m \in \mathbb{S}$, wait till a common value, say $s^{(m)}$, is received from a subset of parties $S_m \setminus Z$, for some $Z \in \mathcal{Z}_s$.
  - Output $s = s^{(1)} + \ldots + s^{(m)}$.

Fig. 5: Protocol for publicly reconstructing a secret-shared value

We next prove the properties of the protocol Π_{Rec}.

Lemma 3. *Let $s \in \mathbb{K}$ be a value, which is secret-shared with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then protocol Π_{Rec} achieves the following with a communication complexity of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2 \log |\mathbb{K}|)$ bits.*

- **Synchronous network**: *All honest parties output s within time Δ.*
- **Asynchronous network**: *All honest parties eventually output s.*

Proof. The communication complexity follows easily from the protocol steps, since corresponding to each $S_m \in \mathbb{S}$, *every* party in S_m needs to send its version of the share $[s]_m$ to *all* the parties in $\mathcal{P}$.

Next, consider an *asynchronous* network and let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties. Consider an *arbitrary* $S_m \in \mathbb{S}$ and an *arbitrary honest* party P_i. We show that P_i eventually sets $s^{(m)}$ to $[s]_m$. This will imply that P_i eventually outputs s. Let $\mathcal{H}_m \stackrel{def}{=} S_m \setminus Z^\star$ be the set of *honest* parties in S_m. All the parties in $\mathcal{H}_m$ send $[s]_m$ to P_i, which are eventually delivered to P_i. Moreover, $S_m \setminus \mathcal{H}_m \in \mathcal{Z}_s$. Hence, it is confirmed that P_i eventually sets $s^{(m)}$ to some value. To complete the proof, we need to show that this values will be the same as $[s]_m$. So, let there exist some $Z_\alpha \in \mathcal{Z}_s$, such that P_i received a common value from all the parties in $S_m \setminus Z_\alpha$, and sets $s^{(m)}$ to this common value. Since $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ condition, it follows that $\mathcal{H}_m \cap (S_m \setminus Z_\alpha) \neq \emptyset$. Thus there exists at least one *honest* party in the set $S_m \setminus Z_\alpha$, who would have sent $[s]_m$ to P_i, implying that P_i sets $s^{(m)}$ to $[s]_m$.

The proof of the lemma statement in a *synchronous* network is exactly the same as above. Moreover, all honest parties will output s within time Δ, since the shares of all honest parties will be delivered within time Δ.

A.3 Beaver's Protocol

Let u and v be two values from $\mathbb{K}$, which are secret-shared with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Moreover, let $(a, b, c) \in \mathbb{K}^3$ be a multiplication-triple, where a, b and c are secret-shared with respect to $\mathbb{S}$ and where $c = a \cdot b$. Then, Beaver's protocol for securely computing $[u \cdot v]$, is given in Fig 6.

Protocol $\Pi_{\mathsf{Beaver}}(([u],[v]),([a],[b],[c]))$

- The parties in $\mathcal{P}$ locally compute $[d]$ and $[e]$, where $[d] = [u] - [a] = [u - a]$ and $[e] = [v] - [b] = [v - b]$.
- The parties publicly reconstruct d and e by executing $\Pi_{\mathsf{Rec}}(d, \mathbb{S})$ and $\Pi_{\mathsf{Rec}}(e, \mathbb{S})$.
- The parties in $\mathcal{P}$ locally compute $[u \cdot v] = d \cdot e + d \cdot [b] + e \cdot [a] + [c]$ and output $[u \cdot v]$.

Fig. 6: Beaver's protocol for securely computing a secret-sharing of the product of two secret-shared values

The properties of the protocol Π_{Beaver} are stated in Lemma 4.

Lemma 4. *Let $\mathcal{A}$ be an adversary, characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network, satisfying the conditions* Con *(see Condition 1 in Section 2). Moreover, let u and v be two values from $\mathbb{K}$, which are secret-shared with respect to the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Furthermore, let $(a, b, c) \in \mathbb{K}^3$ be a multiplication-triple, where a, b and c are secret-shared with respect to $\mathbb{S}$ and where $c = a \cdot b$. Then protocol Π_{Beaver} achieves the following.*

1. $\mathcal{Z}_s$**-correctness***: In a synchronous network, all honest parties output $[w]$ within time Δ, where $w = u \cdot v$.*
2. $\mathcal{Z}_a$**-Correctness***: In an asynchronous network, all honest parties eventually output $[w]$, where $w = u \cdot v$.*
3. **Privacy***: If (a, b, c) is random from the point of view of the adversary, then the view of $\mathcal{A}$ is distributed independent of u and v, irrespective of the network type.*
4. **Communication Complexity:** *The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2 \log |\mathbb{K}|)$ bits.*

Proof. Note that $u \cdot v = ((u - a) + a) \cdot ((v - b) + b) = de + db + ea + c$ holds, where $d = u - a$ and $e = v - b$. From the *linearity* property of the secret-sharing, the parties will be able to locally compute $[d]$ and $[e]$. Moreover, from the properties of Π_{Rec}, all honest parties will be able to compute d and e in a *synchronous* network, within time Δ. On the other hand, in an *asynchronous* network, the honest parties will eventually compute d and e. The $\mathcal{Z}_s$-correctness and $\mathcal{Z}_a$-correctness now follow easily.

The privacy is argued as follows: the only step where the parties communicate is during the reconstruction of d and e. As $d = u - a$, if a is random for $\mathcal{A}$, then even after learning d, in the view of $\mathcal{A}$, the value of u remains as secure as it was before. Similarly, if b is random, then even after learning e, in the view of $\mathcal{A}$, the value v remains as secure as before.

The communication complexity follows from the communication complexity of Π_{Rec}.

B Synchronous BA with Asynchronous Guarantees

Let $\mathcal{Z}$ be an adversary structure satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. We present a protocol Π_{SBA} (Fig 7) which is $\mathcal{Z}$-perfectly-secure in a *synchronous* network, and which has $\mathcal{Z}$-guaranteed liveness in an *asynchronous* network. The protocol is obtained by generalizing the SBA protocol of [8] which was designed against *threshold* adversaries to tolerate $t < n/3$ corruptions.

Let $\mathcal{K} \subseteq \mathcal{P}$ denote a predetermined set of publicly-known parties, such that $\mathcal{K} \notin \mathcal{Z}$. Such a set $\mathcal{K}$ always exists, as the set $\mathcal{P}$ trivially constitutes a candidate $\mathcal{K}$ set. Protocol Π_{SBA} proceeds in phases, where in each phase, a *unique* party from $\mathcal{K}$ is publicly designated as the *king*. Hence, the number of phases is $|\mathcal{K}|$. The general idea behind the protocol is very simple. In each phase, the parties attempt to check if *all honest* parties have the *same* bit. Then, depending upon the answer, the parties do one of the following.

(1): If all honest parties are found to hold a common bit, then the parties "stick" to that bit for all subsequent phases, *irrespective* of the king parties.

(2): Else, the parties take the "help" of the king-party of this phase so that if the king-party is *honest*, then, at the end of this phase, all honest parties have the same bit

If all the honest parties start the protocol with a *common* input bit, say b, then throughout the protocol, the parties retain the bit b, and finally output b, thus ensuring *validity*. On the other hand, *consistency* is guaranteed because there will be *at least* one phase where the corresponding king-party is guaranteed to be *honest*, and hence, at the end of that phase, all honest parties will have a common bit, which they retain till the end of the protocol.

Protocol Π_{SBA}

Let $\mathcal{K} \subseteq \mathcal{P}$ be a publicly-known predetermined set of king-parties, such that $\mathcal{K} \notin \mathcal{Z}$. For simplicity and without loss of generality, let $\mathcal{K} = \{P_1, \ldots, P_{|\mathcal{K}|}\}$.

- On having the input b_i, initialize the *preference bit* $\mathsf{pref}_i = b_i$.
- For $k = 1, \ldots, |\mathcal{K}|$, do the following in phase k, where $P_k \in \mathcal{K}$ is the predetermined designated king-party for phase k:
  1. Send the current preference bit pref_i to every party in $\mathcal{P}$.
  2. If some preference bit b is received from a set of parties $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}$, then send $(\mathsf{propose}, b)$ to every party in $\mathcal{P}$.
     – If $(\mathsf{propose}, b)$ is received from some set of parties $C \notin \mathcal{Z}$, then update the preference bit $\mathsf{pref}_i = b$
  3. If $P_i = P_k$, then send $(\mathsf{king}, k, \mathsf{pref}_k)$ to every party in $\mathcal{P}$.
  4. Setting the preference bit for the next phase:
     – If, during step 2, $(\mathsf{propose}, b)$ was received from a set of parties $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}$, then set $\mathsf{pref}_i = b$.
     – Else, set $\mathsf{pref}_i = \mathsf{pref}_k$, where $(\mathsf{king}, k, \mathsf{pref}_k)$ was received from the king-party during step 3.
- Output pref_i.

Fig. 7: A perfectly-secure SBA protocol with asynchronous guaranteed liveness. The above code is executed by every $P_i \in \mathcal{P}$

Before proving the properties of Π_{SBA}, we prove some helping lemmas which will be useful later.

Lemma 5. *In a synchronous network, if all honest parties hold the same preference bit b at the beginning of a phase k in Π_{SBA}, then they retain the same bit b at the end of phase k.*

Proof. Let $Z_c \in \mathcal{Z}$ be the set of corrupt parties and $\mathcal{H} \stackrel{def}{=} \mathcal{P} \setminus Z_c$ be the set of *honest* parties. Since every party in $\mathcal{H}$ holds the same preference bit b at the beginning of phase k, they will all send b to each party in step 1 of phase k.

Thus, each party in $\mathcal{H}$ receives the bit b from the set of parties $\mathcal{H}$, where $\mathcal{H} = \mathcal{P} \setminus Z_c$. The parties in Z_c may send the bit $\bar{b}$ as their preference bit.

However, since $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, $\mathcal{H} \notin \mathcal{Z}$. Consequently, each party in $\mathcal{H}$ sends $(\mathsf{propose}, b)$ to each party in step 2 of phase k.

Thus, each party in $\mathcal{H}$ receives $(\mathsf{propose}, b)$ from the set of parties $\mathcal{H}$ during step 2 of phase k. Now due to the same reasons as above, it follows that every party in $\mathcal{H}$ sets b as their preference bit at the end of phase k.

Lemma 6. *In a synchronous network, if any honest party $P_i \in \mathcal{P}$ proposes a bit b in step 2 of phase k in Π_{SBA}, then no other honest party $P_j \in \mathcal{P}$ proposes a bit $\bar{b}$.*

Proof. We prove the lemma through a contradiction. So, let there exist two honest parties $P_i, P_j \in \mathcal{P}$, such that P_i sends $(\mathsf{propose}, b)$, and P_j sends $(\mathsf{propose}, \bar{b})$ during step 2 of phase k. This implies that P_i must have received the preference bit b during step 1 of phase k from a set of parties $\mathcal{P} \setminus Z_\alpha$, for some $Z_\alpha \in \mathcal{Z}$. Similarly, P_j must have received the preference bit $\bar{b}$ during step 1 of phase k from a set of parties $\mathcal{P} \setminus Z_\beta$, for some $Z_\beta \in \mathcal{Z}$. Let $\mathcal{T} = (\mathcal{P} \setminus Z_\alpha) \cap (\mathcal{P} \setminus Z_\beta)$. Since $\mathcal{Z}$ satisfies the $Q^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, it follows that $\mathcal{Z}$ satisfies the $Q^{(1)}(\mathcal{T}, \mathcal{Z})$ condition, and hence, there exists at least one *honest* party in $\mathcal{T}$, say P_k. Hence, P_k must have sent b as its preference bit to P_i and $\bar{b}$ as its preference bit to P_j during step 1 of phase k, which is impossible.

Lemma 7. *In a synchronous network, for any phase k, if the designated king-party P_k is honest, then all honest parties have the same preference bit at the end of phase k.*

Proof. Since P_k is *honest*, it sends the *same* $(\mathsf{king}, k, \mathsf{pref}_k)$ message to *all* the parties during the step 3 of phase k. To prove the lemma, we consider two cases that are possible for the honest parties during step 4 of phase k:

- **Case I — All honest parties set pref_k as their preference bit**: In this case, the lemma holds trivially.
- **Case II — Some honest P_i does not set pref_k as its preference bit**: In this case, during step 4, party P_i must have set its preference bit to b, where the bit b is proposed to P_i during step 2 by a set of parties $\mathcal{P} \setminus Z$, for some $Z \in \mathcal{Z}$. Then, consider the set $(\mathcal{P} \setminus Z) \setminus Z_c = \mathcal{P} \setminus (Z \cup Z_c)$, where $Z_c \in \mathcal{Z}$ is the set of *corrupt* parties. The set $\mathcal{P} \setminus (Z \cup Z_c)$ is *non-empty*, consisting of *only honest* parties and $\mathcal{P} \setminus (Z \cup Z_c) \notin \mathcal{Z}$. This is because the set $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition.
  Now, during step 2 of phase k, party P_k will receive the $(\mathsf{propose}, b)$ message from all the parties in $\mathcal{P} \setminus (Z \cup Z_c)$. Moreover, from Lemma 6, it follows that any honest party *outside* $\mathcal{P} \setminus (Z \cup Z_c)$ will *never* send a $(\mathsf{propose}, \bar{b})$ message to P_k during step 2 of phase k. Furthermore, even though the parties in Z_c may send a $(\mathsf{propose}, \bar{b})$ message to P_k during step 2 of phase k, these messages *will not* be considered by P_k to determine pref_k, since $Z_c \in \mathcal{Z}$. Based on the above arguments, it follows that P_k would set $\mathsf{pref}_k = b$ at the end of step 2 of phase k. Since P_k is *honest*, it sends b as pref_k during the step 3 of phase k. Thus, even in this case, every honest party sets the same preference at the end of phase k.

We now prove the properties of the protocol Π_{SBA} in a *synchronous* network.

Lemma 8. *Let $\mathcal{A}$ be an adversary characterised by an adversary structure $\mathcal{Z}$, satisfying the $Q^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, such that $\mathcal{A}$ can corrupt any subset of parties from $\mathcal{Z}$ during the execution of Π_{SBA}. Then, Π_{SBA} achieves the following in a synchronous network.*

- **$\mathcal{Z}$-Liveness**: *All honest parties obtain an output within time $3n \cdot \Delta$.*
- **$\mathcal{Z}$-Validity**: *If all honest parties have input b, then they output b.*
- **$\mathcal{Z}$-Consistency**: *All honest parties output the same value.*
- **Communication Complexity**: *The protocol incurs a communication of $\mathcal{O}(n^3)$ bits.*[11]

Proof. Each phase involves 3 rounds of communication and consequently, each phase requires 3Δ time. The $\mathcal{Z}$-liveness then follows from the fact that all the parties will output something after $|\mathcal{K}|$ phases, where $|\mathcal{K}| \leq n$.

If all honest parties hold the same input b, then at the start of phase 1, they will all set their preference bit as b. Then from repeated application of Lemma 5, it follows that the honest parties retain the bit b as their preference bit throughout at the end of each phase. Thus, Π_{SBA} achieves $\mathcal{Z}$-validity.

For $\mathcal{Z}$-consistency, let there exist a phase $k \in \{1, \ldots, |\mathcal{K}|\}$, such that the designated king-party P_k of this phase is *honest*. From Lemma 7, it follows that at the end of phase k, every honest party sets the same preference bit. Then, from repeated application of Lemma 5, it follows that the honest parties retain the same preference bit till the end of the protocol and output that bit. To complete the proof for consistency, we need to show that there exists at least one *honest* $P_k \in \mathcal{K}$. However, this follows from the fact that $\mathcal{K} \notin \mathcal{Z}$.

In each phase, $\mathcal{O}(n^2)$ bits are exchanged and so overall $\mathcal{O}(|\mathcal{K}| \cdot n^2)$ bits are exchanged. The communication complexity then follows, since $|\mathcal{K}| \leq n$.

Remark 1 (**$\mathcal{Z}$-guaranteed liveness for Π_{SBA} in an asynchronous network).** For any given adversary structure $\mathcal{Z}$, the largest possible set $\mathcal{K}$ is the set $\mathcal{P}$ of all parties, as $\mathcal{P} \notin \mathcal{Z}$. So the maximum time taken by Π_{SBA} to generate an output (in a *synchronous* network) is $3n \cdot \Delta$. Thus, to achieve $\mathcal{Z}$-guaranteed liveness in an *asynchronous* network, it suffices to execute Π_{SBA} till time $3n \times \Delta$ and then output $\perp$, if no output can be deduced from Π_{SBA}. Therefore, Π_{SBA} achieves $\mathcal{Z}$-guaranteed liveness even in an asynchronous network.

C Properties of the Best-of-Both-Worlds BA Protocol

We first start with the properties of the protocol Π_{BC} (see Fig 1 for the formal steps of the protocol).

Theorem 2. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure*

[11] If the inputs of the parties are of size ℓ bits, then the parties can run ℓ independent instances of the protocol, one corresponding to each bit of their input. This will incur a communication complexity of $\mathcal{O}(n^3\ell)$ bits.

$\mathcal{Z}$ satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Moreover, let S have input $m \in \{0,1\}^\ell$ for Π_{BC}. Then, Π_{BC} achieves the following with a communication complexity of $\mathcal{O}(n^3\ell)$ bits, where $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{SBA}}$ and $T_{\mathsf{SBA}} \leq 3n \cdot \Delta$.

- *Synchronous network:*
  - **(a) $\mathcal{Z}$-Liveness**: *At time T_{BC}, each honest party has an output.*
  - **(b) $\mathcal{Z}$-Validity**: *If S is honest, then at time T_{BC}, each honest party outputs m.*
  - **(c) $\mathcal{Z}$-Consistency**: *If S is corrupt, then the output of every honest party is the same at time T_{BC}.*
  - **(d) $\mathcal{Z}$-Fallback Consistency**: *If S is corrupt and some honest party outputs $m^\star \neq \bot$ at time T through fallback-mode, then every honest party outputs $m^\star$ by time $T + 2\Delta$.*
- *Asynchronous Network:*
  - **(a) $\mathcal{Z}$-Liveness**: *At time T_{BC}, each honest party has an output.*
  - **(b) $\mathcal{Z}$-Weak Validity**: *If S is honest, then at time T_{BC}, each honest party outputs m or $\bot$.*
  - **(c) $\mathcal{Z}$-Fallback Validity**: *If S is honest, then each honest party with output $\bot$ at time T_{BC}, eventually outputs m through fallback-mode.*
  - **(d) $\mathcal{Z}$-Weak Consistency**: *If S is corrupt, then at time T_{BC}, each honest party outputs a common $m^\star \neq \bot$ or $\bot$.*
  - **(e) $\mathcal{Z}$-Fallback Consistency**: *If S is corrupt and some honest party outputs $m^\star \neq \bot$ at time T where $T \geq T_{\mathsf{BC}}$, then each honest party eventually outputs $m^\star$.*

Proof. The $\mathcal{Z}$-*liveness* property follows from the fact that every honest party outputs something (including $\bot$) at (local) time T_{BC}, irrespective of the type of the network. We next prove the rest of the properties of the protocol in the *synchronous* network.

If S is *honest*, then due to the $\mathcal{Z}$-*liveness* and $\mathcal{Z}$-*validity* properties of Π_{ACast} in the *synchronous* network, all honest parties receive m from the Acast of S at time 3Δ. Consequently, all honest parties participate with input m in the instance of Π_{SBA}. The $\mathcal{Z}$-*guaranteed liveness* and $\mathcal{Z}$-*validity* properties of Π_{SBA} in the *synchronous* network guarantees that at time $3\Delta + T_{\mathsf{SBA}}$, all honest parties will have m as the output from the instance of Π_{SBA}. As a result, all honest parties output m at time T_{BC}, thus proving the $\mathcal{Z}$-*validity* property.

To prove the $\mathcal{Z}$-*consistency* property, we consider a *corrupt* S. From the $\mathcal{Z}$-*consistency* property of Π_{SBA} in the *synchronous* network, all honest parties will have the *same* output from the instance of Π_{SBA} at time T_{BC}. If all honest parties have the output $\bot$ for Π_{BC} at time T_{BC}, then $\mathcal{Z}$-*consistency* holds trivially. So, consider the case when some *honest* party, say P_i, has the output $m^\star \neq \bot$ for Π_{BC} at time T_{BC}. This implies that all honest parties have the output $m^\star$ from the instance of Π_{SBA}. Moreover, at time 3Δ, at least one *honest* party, say P_h, has received $m^\star$ from the Acast of S. If the latter does not hold, then all honest parties would have participated with input $\bot$ in the instance of Π_{SBA}, and from the $\mathcal{Z}$-*validity* of Π_{SBA} in the *synchronous* network, all honest parties would compute $\bot$ as the output during the instance of Π_{SBA}, which is a contradiction. Since P_h

has received $m^\star$ from S's Acast at time 3Δ, it follows from the $\mathcal{Z}$*-consistency* property of Π_{ACast} in the *synchronous* network that *all* honest parties will receive $m^\star$ from S's Acast by time 5Δ. Moreover, $5\Delta < 3\Delta + T_{\mathsf{SBA}}$ holds. Consequently, at time $3\Delta + T_{\mathsf{SBA}}$, *all* honest parties will have $m^\star$ from S's Acast *and* as the output of Π_{SBA}, implying that all honest parties output $m^\star$ for Π_{BC}.

We next prove the $\mathcal{Z}$*-fallback consistency* property for which we again consider a *corrupt* S. Let P_h be an *honest* party who outputs $m^\star \neq \perp$ at time T through fallback-mode. Note that $T > T_{\mathsf{BC}}$, as the output during the fallback-mode is computed only after time T_{BC}. We also note that *each* honest party has output $\perp$ at time T_{BC}. This is because, from the proof of the $\mathcal{Z}$*-consistency* property of Π_{BC} (see above), if any *honest* party has an output $m' \neq \perp$ at time T_{BC}, then *all* honest parties (including P_h) must have computed the output m' at time T_{BC}. Hence, P_h will never change its output to $m^\star$.[12] Now since P_h has obtained the output $m^\star$, it implies that at time T, it has received $m^\star$ from the Acast of S. It then follows from the $\mathcal{Z}$*-consistency* of Π_{ACast} in the *synchronous* network that every honest party will also receive $m^\star$ from the Acast of S, latest by time $T + 2\Delta$ and output $m^\star$. This completes the proof of all the properties in the *synchronous* network.

We next prove the properties of the protocol Π_{BC} in an *asynchronous* network. The $\mathcal{Z}$*-weak validity* property follows from the $\mathcal{Z}$*-validity* property of Π_{ACast} in the *asynchronous* network, which ensures that no honest party ever receives an m' from the Acast of S, where $m' \neq m$. So, if at all any honest party outputs a value different from $\perp$ at time T_{BC}, it has to be m. The $\mathcal{Z}$*-weak consistency* property follows using similar arguments as used to prove $\mathcal{Z}$*-consistency* in the *synchronous* network; however we now rely on the $\mathcal{Z}$*-validity* and $\mathcal{Z}$*-consistency* properties of Π_{ACast} in the asynchronous network. The latter property ensures that for a *corrupt* S, two different honest parties never end up receiving m_1 and m_2 from the Acast of S, where $m_1 \neq m_2$.

For the $\mathcal{Z}$*-fallback validity* property, consider an *honest* S, and let P_i be an arbitrary *honest* party who outputs $\perp$ at (local) time T_{BC}. Since the parties keep on participating in the protocol beyond time T_{BC}, it follows from the $\mathcal{Z}$*-liveness* and $\mathcal{Z}$*-validity* properties of Π_{ACast} in the *asynchronous* network that party P_i will *eventually* receive m from the Acast of S, by executing the steps of the fallback-mode of Π_{BC}. Consequently, party P_i eventually changes its output from $\perp$ to m.

For the $\mathcal{Z}$*-fallback consistency* property, we consider a *corrupt* S. Let P_j be an *honest* party who outputs some $m^\star$ different from $\perp$ at time T, where $T \geq T_{\mathsf{BC}}$. This implies that P_j has obtained $m^\star$ from the Acast of S. Now, consider an arbitrary *honest* P_i. From the $\mathcal{Z}$*-liveness* and $\mathcal{Z}$*-weak consistency* properties of Π_{BC} in *asynchronous* network proved above, it follows that P_i outputs either $m^\star$ or $\perp$ at local time T_{BC}. If P_i has output $\perp$, then from the $\mathcal{Z}$*-consistency* property of Π_{ACast} in the *asynchronous* network, it follows that P_i will also eventually

[12] Recall that in the protocol Π_{BC}, the parties who obtain an output different from $\perp$ at time T_{BC}, never change their output.

obtain $m^\star$ from the Acast of S, by executing the steps of the fallback-mode of Π_{BC}. Consequently, party P_i eventually changes its output from $\bot$ to $m^\star$.

The *communication complexity* (both in the synchronous as well as asynchronous network) follows from the communication complexity of Π_{SBA} and Π_{ACast}.

C.1 Properties of the Protocol Π_{BA}

In this section, we prove the properties of our best-of-both-worlds BA protocol Π_{BA} (see Fig 2 for the formal description of the protocol).

Theorem 3. *Let $\mathcal{A}$ be an adversary, characterized by an adversary structure $\mathcal{Z}$, satisfying the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition. Moreover, let Π_{ABA} be an ABA protocol satisfying the conditions as stated in Lemma 2. Then Π_{BA} achieves the following.*

- **Synchronous Network**: *The protocol is a $\mathcal{Z}$-perfectly-secure SBA protocol, where all honest parties obtain an output within time $T_{\mathsf{BA}} = T_{\mathsf{BC}} + T_{\mathsf{ABA}}$. The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}| \cdot n^5 \log |\mathbb{F}| + n^6 \log n)$ bits.*
- **Asynchronous Network**: *The protocol is a $\mathcal{Z}$-perfectly-secure ABA protocol with an expected communication of $\mathcal{O}(|\mathcal{Z}| \cdot n^7 \log |\mathbb{F}| + n^8 \log n)$ bits.*

Proof. We start with the properties in a *synchronous* network. The $\mathcal{Z}$-liveness property of Π_{BC} in the *synchronous* network guarantees that all honest parties will have some output, from each instance of Π_{BC}, at time T_{BC}. Moreover, the *$\mathcal{Z}$-validity* and *$\mathcal{Z}$-consistency* properties of Π_{BC} in the *synchronous* network guarantee that irrespective of the sender parties, *all* honest parties will have a common output from each individual instance of Π_{BC}, at time T_{BC}. Now since the parties decide their respective inputs for the instance of Π_{ABA} *deterministically* based on the individual outputs from the n instances of Π_{BC} at time T_{BC}, it follows that all honest parties participate with a *common* input in the protocol Π_{ABA}. Hence, all honest parties obtain an output by the end of time $T_{\mathsf{BC}} + T_{\mathsf{ABA}}$, thus ensuring *$\mathcal{Z}$-guaranteed liveness* of Π_{BA}. Moreover, the *$\mathcal{Z}$-consistency* property of Π_{ABA} in the *synchronous* network guarantees that all honest parties have a *common* output from the instance of Π_{ABA}, which is taken as the output of Π_{BA}, thus proving the *$\mathcal{Z}$-consistency* of Π_{BA}.

For proving the validity property in a synchronous network, let all *honest* parties have the same input bit b. Let $Z_c \in \mathcal{Z}$ be the set of *corrupt* parties and $\mathcal{H} \stackrel{def}{=} \mathcal{P} \setminus Z_c$ be the set of *honest* parties. From the *$\mathcal{Z}$-validity* of Π_{BC} in the *synchronous* network, all honest parties will receive b as the output at time T_{BC} in all the Π_{BC} instances corresponding to the sender parties in $\mathcal{H}$. Since $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, it follows that $\mathcal{H} \notin \mathcal{Z}$. Consequently, all honest parties will find a *common* subset $\mathcal{R}$ in the protocol, as the set $\mathcal{H}$ constitutes a candidate $\mathcal{R}$. Moreover, $\mathcal{H} \subseteq \mathcal{R}$. Furthermore, $\mathcal{R} \setminus \mathcal{H} \subseteq Z_c \in \mathcal{Z}$ and $\mathcal{R} \setminus Z_c \subseteq \mathcal{H} \notin \mathcal{Z}$. Hence, all honest parties P_i will find a *common* subset $R_i \subseteq \mathcal{R}$, as per the protocol, such that b is computed as the output during the Π_{BC} instances of all the parties in $\mathcal{R}_i$. As a result, all honest parties will participate with input b in the instance of

Π_{ABA} and hence, output b at the end of Π_{ABA}, which follows from the $\mathcal{Z}$*-validity* of Π_{ABA} in the *synchronous* network. This proves the $\mathcal{Z}$*-validity* of Π_{BA}.

We next prove the properties of Π_{BA} in an *asynchronous* network. The $\mathcal{Z}$*-consistency* of the protocol Π_{BA} follows from the $\mathcal{Z}$*-consistency* of the protocol Π_{ABA} in the *asynchronous* network, since the overall output of the protocol Π_{BA} is same as the output of the protocol Π_{ABA}. The $\mathcal{Z}$*-liveness* of the protocol Π_{BC} in the *asynchronous* network guarantees that all honest parties will have some output from all the n instances of Π_{BC} at local time T_{BC}. Consequently, all honest parties will participate with some input in the instance of Π_{ABA}. The $\mathcal{Z}$*-almost-surely liveness* of Π_{ABA} in the *asynchronous* network then implies the $\mathcal{Z}$*-almost-surely liveness* of Π_{BA}.

For proving the validity in an *asynchronous* network, let all *honest* parties have the same input bit b. Let $Z_c \in \mathcal{Z}$ be the set of *corrupt* parties and $\mathcal{H} \stackrel{def}{=} \mathcal{P} \setminus Z_c$ be the set of *honest* parties. We claim that all *honest* parties participate with the input b during the instance of Π_{ABA}. The $\mathcal{Z}$*-validity* of Π_{ABA} in the *asynchronous* network then automatically implies the $\mathcal{Z}$*-validity* of Π_{BA}.

To prove the above claim, consider an arbitrary *honest* party P_h. There are two possible cases. If P_h fails to find a subset $\mathcal{R}$ satisfying the protocol conditions, then the claim holds trivially, as P_h participates in the instance of Π_{ABA} with its input for Π_{BA}, which is the bit b. So, consider the case when P_h finds a subset $\mathcal{R}$ such that $\mathcal{P} \setminus \mathcal{R} \in \mathcal{Z}$, and where, corresponding to each $P_j \in \mathcal{R}$, party P_h has computed an output $b_h^{(j)} \in \{0, 1\}$ at local time T_{BC} during the instance $\Pi_{\mathsf{BC}}^{(j)}$. Now, consider the subset of *honest* parties $\mathcal{H} \cap \mathcal{R}$ in the set $\mathcal{R}$. It follows that $\mathcal{R} \setminus (\mathcal{H} \cap \mathcal{R}) \subseteq Z_c \in \mathcal{Z}$. Also, since $\mathcal{Z}$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$ condition, it follows that $(\mathcal{H} \cap \mathcal{R}) \notin \mathcal{Z}$. Moreover, P_h will compute the output b at local time T_{BC} in the instance of Π_{BC} corresponding to *every* $P_j \in (\mathcal{H} \cap \mathcal{R})$, which follows from the $\mathcal{Z}$*-weak validity* of Π_{BC} in the *asynchronous* network. From these arguments, it follow that P_h will find a candidate subset $\mathcal{R}_h$, where $\mathcal{R} \setminus \mathcal{R}_h \in \mathcal{Z}$ and where b is computed as the output at local time T_{BC} in the instance of Π_{BC}, corresponding to *every* $P_j \in \mathcal{R}_h$. This is because the subset of parties $\mathcal{H} \cap \mathcal{R}$ constitutes a candidate $\mathcal{R}_h$. Consequently, P_h will set b as its input for the instance of Π_{ABA}, thus proving the claim.

The communication complexity, both in a synchronous as well as in an asynchronous network, follows easily from the protocol steps and from the communication complexity of Π_{SBA} and Π_{ABA}.

D Properties of the Best-of-Both-Worlds VSS Protocol

In this section we prove the properties of the protocol Π_{VSS} (see Fig 3 for the formal description of the protocol). Recall that we want to prove the properties of Π_{VSS} assuming an adversary $\mathcal{A}$ characterized by an adversary structure $\mathcal{Z}_s$ in a *synchronous* network, and an adversary structure $\mathcal{Z}_a$ in an *asynchronous* network, satisfying the following conditions.

- $\mathcal{Z}_s \neq \mathcal{Z}_a$;
- For every subset $Z \in \mathcal{Z}_a$, there exists a subset $Z' \in \mathcal{Z}_s$, such that $Z \subseteq Z'$;

– $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.

Moreover, we are considering the sharing specification $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Note that the above conditions automatically imply that $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ condition and $\mathcal{Z}_a$ satisfies the $\mathcal{Q}^{(4)}(\mathcal{P}, \mathcal{Z}_a)$ condition. Before proving the properties, we prove a related property, which will later be useful while proving the properties of Π_{VSS}.

Lemma 9. *In protocol Π_{VSS} if the network is synchronous and if the output of Π_{BA} is 1, then the following hold.*

- *All honest parties participated in the instance of Π_{BA} with input 1.*
- *Corresponding to every $S_m \in \mathbb{S}$, all honest parties in $\mathcal{C}_m$ received a common share from D by time Δ.*

Proof. Let the network be *synchronous* and let all the honest parties output 1 during the instance of Π_{BA}. This implies at least one *honest* party, say P_h, has participated in Π_{BA} with input 1. If this is not the case and if *all* honest parties participated with input 0 in Π_{BA}, then from the $\mathcal{Z}_s$-validity of Π_{BA} in the *synchronous* network, all honest parties would compute the output 0 during the instance of Π_{BA}, which is a contradiction.

Since P_h has participated with input 1 in the instance of Π_{BA}, it implies that P_h has received the sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$ from the broadcast of D at time $2\Delta + 2T_{\mathsf{BC}}$ through regular-mode and accepted these sets. From the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-consistency of Π_{BC} in the *synchronous* network, every other honest party would also receive these sets from the broadcast of D at time $2\Delta + 2T_{\mathsf{BC}}$. Since P_h has accepted $\mathcal{C}_1, \ldots, \mathcal{C}_q$, it implies that P_h has checked that all the following hold.

- For $m = 1, \ldots, q$, the set $\mathcal{C}_m$ constitutes a clique in the consistency graph $G_h^{(m)}$ of P_h at time $2\Delta + T_{\mathsf{BC}}$. That is, for every $P_i, P_j \in \mathcal{C}_m$, the messages $\texttt{OK}(m, i, j)$ and $\texttt{OK}(m, j, i)$ have been received from the broadcast of P_i and P_j respectively by time $2\Delta + T_{\mathsf{BC}}$. By the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-consistency of Π_{BC} in the *synchronous* network, these messages are also received by every other honest party by time $2\Delta + T_{\mathsf{BC}}$. Consequently, $\mathcal{C}_1, \ldots, \mathcal{C}_q$ will constitute a clique in the consistency graphs $G_k^{(1)}, \ldots, G_k^{(m)}$ respectively of *every* honest party at time $2\Delta + T_{\mathsf{BC}}$.
- For $m = 1, \ldots, q$, the condition $S_m \setminus \mathcal{C}_m \in \mathcal{Z}_s$ holds. It is easy to see that *every* honest party will find that this condition holds.
- For $m = 1, \ldots, q$, if $\texttt{NOK}(m, j)$ was received from the broadcast of any $P_j \in S_m$ through regular mode at time $2\Delta + T_{\mathsf{BC}}$, then the following hold at time $2\Delta + 2T_{\mathsf{BC}}$.
  - $\texttt{Resolve}(m, s^{(m)})$ is received from the broadcast of D through regular-mode.
  - $\texttt{Resolve}(m, s^{(m)})$ is received from the broadcast of a subset of parties $\mathcal{C}'_m$ through regular-mode, where $\mathcal{C}'_m \subseteq \mathcal{C}_m$ and $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$.

  From the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-consistency of Π_{BC} in the *synchronous* network, the above conditions will be also satisfied for *every* honest party. In more detail, if any $\texttt{NOK}(m, j)$ was received by P_h from the broadcast of any $P_j \in S_m$ through regular-mode at time $2\Delta + T_{\mathsf{BC}}$, then the same $\texttt{NOK}(m, j)$ message would be received through regular-mode at time $2\Delta + T_{\mathsf{BC}}$ by *every* honest party. Due to a similar reason, any corresponding $\mathtt{Resolve}(m, s^{(m)})$ message which is received by P_h through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$, either from the broadcast of D or from the broadcast of any party in $\mathcal{C}_m$, will also be received by *every* honest party.

It thus follows that the conditions for accepting the $\mathcal{C}_1, \ldots, \mathcal{C}_q$ will hold for *every* honest party at time $2\Delta + 2T_{\mathsf{BC}}$ and so, *all* honest parties participate with input 1 during the instance of Π_{BA}. This proves the first part of the lemma.

We now prove the second part of the lemma. The statement is obviously true, if D is *honest*. So, we consider a *corrupt* D. Let S_m be an arbitrary set in $\mathbb{S}$. We first note that all the *honest* parties in $\mathcal{C}_m$ received a common share, say $s^{(m)}$, from D. This is because, from the proof of the first part of the lemma, there exists some *honest* party P_h, who has received $\mathcal{C}_m$ from the broadcast of D through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$ and has accepted $\mathcal{C}_m$. And while accepting $\mathcal{C}_m$, party P_h has verified that $\mathcal{C}_m$ constitutes a clique in its consistency graph $G_h^{(m)}$ at time $2\Delta + T_{\mathsf{BC}}$. Hence, the messages $\mathtt{OK}(m, i, j)$ and $\mathtt{OK}(m, j, i)$ have been received by P_h from the broadcast of *every honest* $P_i, P_j \in \mathcal{C}_m$ by time $2\Delta + T_{\mathsf{BC}}$. This automatically implies that $s_i^{(m)} = s_j^{(m)}$ holds, where $s_i^{(m)}$ and $s_j^{(m)}$ denotes the shares received from D by P_i and P_j respectively, corresponding to S_m. We wish to show that both P_i and P_j would have received their respective shares within time Δ.

On the contrary, let P_i receive $s_i^{(m)}$ from D at time $\Delta + \delta$, where $\delta > 0$. From the protocol steps, P_i starts performing pairwise consistency checks only when its local time becomes a multiple of Δ. Hence, P_i must have started sending its share $s_i^{(m)}$ to the other parties at time $c \cdot \Delta$, where $c > 1$. Similarly, from the protocol steps, P_j will start broadcasting the $\mathtt{OK}(m, j, i)$ message, *only* at time $(c + 1) \cdot \Delta$, since it waits till its local time becomes a multiple of Δ, before broadcasting any $\mathtt{OK}$ or $\mathtt{NOK}$ messages. Now, by the $\mathcal{Z}_s$-validity of Π_{BC} in the *synchronous* network, it takes T_{BC} time for the $\mathtt{OK}(m, j, i)$ messages to be received by *any* honest party. Hence, the edge (P_i, P_j) gets added in the consistency graph $G_k^{(m)}$ of *every* honest party only at time $(c + 1) \cdot \Delta + T_{\mathsf{BC}}$. However, this is a contradiction, since $(c + 1) \cdot \Delta + T_{\mathsf{BC}} > 2\Delta + T_{\mathsf{BC}}$.

We now proceed to prove the properties of the protocol Π_{VSS}. We start with the *correctness* property in the *synchronous* network.

Lemma 10. *In protocol Π_{VSS}, if D is honest and participates with input s, then in a synchronous network, s is secret-shared, with respect to the sharing specification $\mathbb{S}$, at time $T_{\mathsf{VSS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.*

Proof. Let $Z_s^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties, and let $\mathcal{H}_s \stackrel{def}{=} \mathcal{P} \setminus Z_s^\star$ be the set of *honest* parties. We show that corresponding to *every* $S_m \in \mathbb{S}$, all *honest* parties in S_m output the share $s^{(m)}$ at time T_{VSS}, where $s^{(m)}$ is the share picked by D, corresponding to S_m. The lemma then follows from the fact that since D is *honest*, it selects the shares $s^{(1)}, \ldots, s^{(q)}$, satisfying the condition

$s^{(1)} + \ldots + s^{(q)} = s$. So, consider an *arbitrary* $S_m \in \mathbb{S}$, and let $\mathcal{H}_m = S_m \setminus Z_s^\star$ be the set of *honest* parties in S_m.

During Phase I, *every* $P_i \in \mathcal{H}_m$ receives the share $s_i^{(m)}$ from D within time Δ, where $s_i^{(m)} = s^{(m)}$. During Phase II, *every* $P_i \in \mathcal{H}_m$ sends $s_i^{(m)}$ to *every* $P_j \in S_m$, which takes at most Δ time to be delivered. Hence, by time 2Δ, *every* $P_i \in \mathcal{H}_m$ receives $s_j^{(m)}$ from *every* $P_j \in \mathcal{H}_m$, such that $s_i^{(m)} = s_j^{(m)}$ holds. Consequently, during Phase III, *every* $P_i \in \mathcal{H}_m$ broadcasts $\texttt{OK}(m, i, j)$ corresponding to *every* $P_j \in \mathcal{H}_m$, and vice versa. From the $\mathcal{Z}_s$-validity of Π_{BC} in the *synchronous* network, it follows that *all* the parties in $\mathcal{H}_m$ receive the $\texttt{OK}(m, i, j)$ and $\texttt{OK}(m, j, i)$ messages through regular-mode at time $2\Delta + T_{\mathsf{BC}}$, from the broadcast of *every* $P_i \in \mathcal{H}_m$ and *every* $P_j \in \mathcal{H}_m$. Hence, corresponding to *every* $P_i, P_j \in \mathcal{H}_m$, the edge (P_i, P_j) will be added to the graph $G_k^{(m)}$ of *every* $P_k \in \mathcal{H}_m$. Furthermore, from the $\mathcal{Z}_s$-consistency property of Π_{BC} in the *synchronous* network, the graph $G_k^{(m)}$ will be the *same* for *every* $P_k \in \mathcal{H}_m$ (including D) at time $2\Delta + T_{\mathsf{BC}}$.

From the above arguments, the set of parties in $\mathcal{H}_m$ will constitute a clique in the consistency graph of *all* the parties in $\mathcal{H}_m$. Moreover, $S_m \setminus \mathcal{H}_m \subseteq Z_s^\star \in \mathcal{Z}_s$. Hence, during Phase IV, D will be able to find a candidate $\mathcal{C}_m$ set and broadcast it, which will be received by *all* the parties in $\mathcal{H}_s$ through regular-mode within time $2\Delta + 2T_{\mathsf{BC}}$ (follows from the $\mathcal{Z}_s$-validity of Π_{BC} in the *synchronous* network). Now, consider an *arbitrary honest* party P_k, who receives the message $\texttt{NOK}(m, j)$ from a party $P_j \in S_m$ through regular-mode at time $2\Delta + T_{\mathsf{BC}}$. From the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-consistency properties of Π_{BC} in the *synchronous* network, this $\texttt{NOK}(m, j)$ message will be received by *all* the parties in $\mathcal{H}_m$ through regular-mode at time $2\Delta + T_{\mathsf{BC}}$. Consequently, *all* the parties in $\mathcal{H}_m$ (including D) will respond by broadcasting the $\texttt{Resolve}(m, s^{(m)})$ message, which will be received by *all* the *honest* parties through regular mode at time $2\Delta + 2T_{\mathsf{BC}}$. Since $\mathcal{C}_m \setminus \mathcal{H}_m \subseteq Z_s^\star \in \mathcal{Z}_s$, it follows that the conditions for accepting $\mathcal{C}_m$ will hold for *all* the honest parties at time $2\Delta + 2T_{\mathsf{BC}}$. Consequently, *all honest* parties will participate with input 1 in the instance of Π_{BA} and from the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-guaranteed liveness properties of Π_{BA} in the *synchronous* network, all honest parties will compute the output 1 in the instance of Π_{BA} at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.

Finally, we show that *all* the parties in $\mathcal{H}_m$ output $s^{(m)}$ at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. For this, we consider the following possible cases with respect to $\mathcal{C}_m$.

1. **An $\texttt{NOK}(m, \star)$ message was received at time $2\Delta + T_{\mathsf{BC}}$ through regular-mode from the broadcast of some party in S_m**: Since *all honest* parties accept $\mathcal{C}_m$, it implies that corresponding to this $\texttt{NOK}$ message, all honest parties have received a $(\texttt{Resolve}, s^{(m)})$ message from the broadcast of D through regular-mode, as well as from the broadcast of a subset of parties $\mathcal{C}'_m \subseteq \mathcal{C}_m$ through regular-mode, where $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$, at time $2\Delta + 2T_{\mathsf{BC}}$. Hence, according to the protocol steps, *every honest* party in S_m outputs $s^{(m)}$ as the share, corresponding to S_m. Also, it is easy to see that the honest parties output $s^{(m)}$ as the share at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.

2. **No $\texttt{NOK}(m, \star)$ message was received through regular-mode from the broadcast of any party in S_m within time** $2\Delta + T_{\mathsf{BC}}$: From Lemma 9, *all* parties in $\mathcal{H}_m \cap \mathcal{C}_m$ would have received the common share $s^{(m)}$ from D, corresponding to the set S_m, within time Δ. As part of the pairwise consistency test, the share $s^{(m)}$ from all the parties in $\mathcal{H}_m \cap \mathcal{C}_m$ would have been delivered to all the *honest* parties in S_m within time 2Δ. This implies that within time 2Δ, *all* honest parties would have received $s^{(m)}$ from a subset of parties $\mathcal{C}''_m \subseteq \mathcal{C}_m$, where $\mathcal{C}_m \setminus \mathcal{C}''_m \in \mathcal{Z}_s$. This is because the set $(\mathcal{C}_m \cap \mathcal{H}_m)$ definitely constitutes a candidate $\mathcal{C}''_m$. Moreover, *no* honest party would have ever received a value *different* from $s^{(m)}$ within time 2Δ, from any party in S_m. On the contrary, if any *honest* P_i receives $s^{(m)}_j$ from P_j and $s^{(m)}_k$ from P_k within time 2Δ, where $P_j, P_k \in S_m$ and where $s^{(m)}_j \neq s^{(m)}_k$, then P_i would have broadcasted an $\texttt{NOK}(m, i)$ message at time 2Δ, which would have been received by all honest parties through regular-mode at time $2\Delta + T_{\mathsf{BC}}$, which is a contradiction.
Hence, in this case also, *every honest* party in S_m outputs $s^{(m)}$ as the share, corresponding to S_m. Also, it is easy to see that the honest parties output $s^{(m)}$ as the share at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.

We next prove the correctness property in an *asynchronous* network.

Lemma 11. *In protocol Π_{VSS}, if D is honest and participates with input s, then in an asynchronous network, almost-surely s is eventually secret-shared with respect to the sharing specification $\mathbb{S}$.*

Proof. Let $Z^\star_a \in \mathcal{Z}_s$ be the set of *corrupt* parties, and let $\mathcal{H}_a \stackrel{def}{=} \mathcal{P} \setminus Z^\star_a$ be the set of *honest* parties. We show that corresponding to *every* $S_m \in \mathbb{S}$, almost-surely, all *honest* parties in S_m eventually output the share $s^{(m)}$, where $s^{(m)}$ is the share picked by D corresponding to S_m. The $\mathcal{Z}_a$-correctness then follows from the fact that since D is *honest*, it selects the shares $s^{(1)}, \ldots, s^{(q)}$, satisfying the condition $s^{(1)} + \ldots + s^{(q)} = s$. So consider an *arbitrary* $S_m \in \mathbb{S}$ and let $\mathcal{H}_m = S_m \setminus Z^\star_a$ be the set of *honest* parties in S_m.

We first note that *every* honest party participates with *some* input in the instance of Π_{BA} at local time $2\Delta + 2T_{\mathsf{BC}}$. Hence from the $\mathcal{Z}_a$-almost-surely liveness and $\mathcal{Z}_a$-consistency properties of Π_{BA} in the *asynchronous* network, all honest parties eventually compute a common output, during the instance of Π_{BA}. Now there are two possible cases with respect to the output of Π_{BA}, according to which the parties proceed to compute their shares.

1. **The output of Π_{BA} is 1**: Since the output of Π_{BA} is 1, due to the $\mathcal{Z}_a$-validity of Π_{BA} in the *asynchronous* network, at least one honest party, say P_h, has received the sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$ from the broadcast of D through regular-mode, within time $2\Delta + T_{\mathsf{BC}}$ and accepted these sets. It then follows that all honest parties also receive the sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$ from the broadcast of D, either through regular-mode or through fallback-mode. This follows from the $\mathcal{Z}_a$-weak validity and $\mathcal{Z}_a$-fallback validity properties of Π_{BC} in the *asynchronous* network. Since P_h has accepted $\mathcal{C}_m$, it has verified that $\mathcal{C}_m$ constitutes a

clique in the consistency graph $G_h^{(m)}$. This implies that corresponding to S_m, *all honest* parties in $\mathcal{C}_m$ have received the *same* share from D, which is $s^{(m)}$, since we are considering an *honest* D. We will show that all *honest* parties in S_m eventually output $s^{(m)}$ as the share, corresponding to S_m.

So consider an *arbitrary honest* $P_i \in S_m$. From the protocol steps, if the output of Π_{BA} is 1, then P_i computes its share corresponding to S_m, based on one of the following three conditions. Assuming that at least one of these conditions eventually hold for P_i, we first show that the share computed by P_i corresponding to S_m, is bound to be $s^{(m)}$. This is followed by showing that indeed at least one of these conditions eventually hold for P_i.

- **Condition A**: At time $2\Delta + 2T_{\mathsf{BC}}$, party P_i received the $\texttt{Resolve}(m, s^{(m)})$ message from the broadcast of D, as well as from the broadcast of a subset of parties $\mathcal{C}'_m \subseteq \mathcal{C}_m$ through regular-mode, where $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$. Clearly, in this case, P_i outputs $s^{(m)}$ as the share corresponding to S_m.
- **Condition B**: At time 2Δ, there exists a subset of parties $\mathcal{C}''_m \subseteq \mathcal{C}_m$ where $\mathcal{C}_m \setminus \mathcal{C}''_m \in \mathcal{Z}_s$, such that P_i received a common value from *all* the parties in $\mathcal{C}''_m$. We claim that the subset $\mathcal{C}''_m$ is bound to contain at least one *honest* party from $\mathcal{C}_m$, which would have sent $s^{(m)}$ to P_i, due to which P_i will output $s^{(m)}$ as the share corresponding to S_m. In more detail, let $S_m \setminus \mathcal{C}_m = Z_\alpha \in \mathcal{Z}_s$, and $\mathcal{C}_m \setminus \mathcal{C}''_m = Z_\beta \in \mathcal{Z}_s$. Also, note that $\mathcal{P} \setminus S_m = Z_m \in \mathcal{Z}_s$. Now, if $\mathcal{C}''_m$ *does not* contain any honest party from $\mathcal{C}_m$, it implies that $\mathcal{C}''_m \subseteq Z_a^\star \in \mathcal{Z}_a$. This further implies that $\mathcal{P} \subseteq Z_m \cup Z_\alpha \cup Z_\beta \cup Z_a^\star$, which is a contradiction to the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.
- **Condition C**: There exists a subset of parties $\mathcal{C}'''_m \subseteq \mathcal{C}_m$, where $\mathcal{C}_m \setminus \mathcal{C}'''_m \in \mathcal{Z}_a$, such that P_i received a common value from *all* the parties in $\mathcal{C}'''_m$. In this case also, one can show that the subset $\mathcal{C}'''_m$ is bound to contain at least one *honest* party from $\mathcal{C}_m$, who would have sent $s^{(m)}$ to P_i. This is because $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition and every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$. Clearly, P_i outputs $s^{(m)}$ as the share corresponding to S_m.

Thus, we have shown that *irrespective* of the way P_i would have computed its output share corresponding to S_m, it is bound to be the *same* as $s^{(m)}$. To complete the proof, we just need to show that at least one of the conditions from A, B and C above eventually holds for P_i. For this, we note that in the *worst* case, the condition C is bound to *eventually* hold, irrespective of conditions A and B. This is because the set of *honest* parties in $\mathcal{C}_m$, namely the parties in $\mathcal{C}_m \setminus Z_a^\star$, *always* constitute a candidate $\mathcal{C}'''_m$ set for P_i. This follows from the fact that the share $s^{(m)}$ from *all* the parties in $\mathcal{C}_m \setminus Z_a^\star$ will be eventually delivered to P_i.

2. **The output of Π_{BA} is 0**: Since D is *honest*, every pair of parties $P_j, P_k \in \mathcal{H}_m$ eventually broadcast $\texttt{OK}(m, j, k)$ and $\texttt{OK}(m, k, j)$ messages, as they eventually receive the same share $s^{(m)}$ from D and exchange among themselves. From the $\mathcal{Z}_a$-validity of Π_{BC} in the *asynchronous* network, these messages are eventually delivered to every honest party. Also from the $\mathcal{Z}_a$-consistency

of Π_{BC} in the *asynchronous* network, any OK message which is received by D from the broadcast of any *corrupt* party, will eventually be received by every other honest party as well. Since $S_m \setminus \mathcal{H}_m \in \mathcal{Z}_a$, it follows that *all honest* parties will eventually find a subset of parties $\mathcal{E}_m \subseteq S_m$, where $S_m \setminus \mathcal{E}_m \in \mathcal{Z}_a$, which constitutes a clique in the consistency graph $G^{(m)}$ of *all* honest parties. This is because the set $\mathcal{H}_m$ constitutes such a candidate $\mathcal{E}_m$. Consequently, D eventually finds and broadcasts $\mathcal{E}_m$. From the $\mathcal{Z}_a$-weal validity and $\mathcal{Z}_a$-fallback validity properties of Π_{BC} in the *asynchronous* network, $\mathcal{E}_m$ will be eventually received and accepted by *all* honest parties.

Next, consider an arbitrary $P_i \in \mathcal{H}_m$. We wish to show that P_i eventually outputs $s^{(m)}$ as the share, corresponding to S_m. Now, there are two possible cases. If $P_i \in \mathcal{E}_m$, then from the protocol steps, P_i indeed outputs $s^{(m)}$ as its share, corresponding to S_m. So, consider the other case when $P_i \notin \mathcal{E}_m$. Note that all the parties in $\mathcal{H}_m$ eventually receive the common share $s^{(m)}$ from D, since D is *honest*. Also note that $\mathcal{E}_m \setminus \mathcal{H}_m \in \mathcal{Z}_s$; this is because every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$. Hence, it follows that party P_i will eventually find a candidate subset $\mathcal{E}'_m \subseteq \mathcal{E}_m$, where $\mathcal{E}_m \setminus \mathcal{E}'_m \in \mathcal{Z}_s$, such that P_i receives a *common* value from *all* the parties in $\mathcal{E}'_m$ and set that value as its share, corresponding to S_m. This is because the subset $(\mathcal{H}_m \cap \mathcal{E}_m)$ always constitute such a candidate $\mathcal{E}'_m$ set. Hence, it is confirmed that P_i is guaranteed to output some share corresponding to S_m. To complete the proof, we need to show that this share is the *same* as $s^{(m)}$.

So, let P_i find a candidate $\mathcal{E}'_m$ set, satisfying the above conditions, based on which it computes its output share corresponding to S_m. We claim that this set $\mathcal{E}'_m$ contains at least one *honest* party from $\mathcal{H}_m$; i.e. $\mathcal{H}_m \cap \mathcal{E}'_m \neq \emptyset$. On the contrary, let the candidate $\mathcal{E}'_m$ for P_i consists of only *corrupt* parties. That is, $\mathcal{E}'_m \subseteq Z^\star_a$. We consider the worst case scenario where $Z^\star_a \in \mathcal{Z}_s$ as well, since every subset in $\mathcal{Z}_a$ is assumed to be a subset of some subset in $\mathcal{Z}_s$. Also note that $S_m = \mathcal{P} \setminus Z_m$, where $Z_m \in \mathcal{Z}_s$. Let $S_m \setminus \mathcal{E}_m \subseteq Z_\beta \in \mathcal{Z}_a$. And let $\mathcal{E}_m \setminus \mathcal{E}'_m \subseteq Z_\alpha \in \mathcal{Z}_s$. Hence, we get that $\mathcal{P} \subseteq Z_m \cup Z^\star_a \cup Z_\alpha \cup Z_\beta$, which is a contradiction, since $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.

We next prove the *privacy* property.

Lemma 12. *In protocol Π_{VSS}, if* D *is honest and participates with input s, then irrespective of the type of network, the view of the adversary remains independent of s.*

Proof. We prove privacy in a *synchronous* network. The privacy in an *asynchronous* network automatically follows, since every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$. So, consider a *synchronous* network, and let D be *honest*. Let $Z_c \in \mathcal{Z}_s$ be the set of *corrupt* parties. Then consider the set $S_c \in \mathbb{S}$, where $S_c \stackrel{def}{=} \mathcal{P} \setminus Z_c$. We claim that, throughout the protocol, the adversary does not learn anything about the share $s^{(c)}$, and its view remains independent of $s^{(c)}$. The $\mathcal{Z}_s$-privacy then follows from the fact that D selects the share $s^{(c)}$ randomly, and the probability distribution of $s^{(c)}$ is *independent* of s.

Since D is *honest*, it sends the share $s^{(c)}$, *only* to the parties in S_c, which consists of *only* honest parties. Similarly, as part of the pairwise consistency tests, the share $s^{(c)}$ is exchanged *only* among the parties in S_c. Moreover, since S_c consists of *no* corrupt parties, it follows that *no* party from S_c ever broadcasts an $\texttt{NOK}(c, \star)$ message, corresponding to S_c. Consequently, *no* party from S_c, as well as D, ever broadcasts a $\texttt{Resolve}(c, s^{(c)})$ message. Thus, throughout the protocol, the view of the adversary remains independent of the share $s^{(c)}$.

We next proceed to prove the commitment properties. We start with the *synchronous* network.

Lemma 13. *In protocol Π_{VSS}, if D is corrupt, then either no honest party obtains any output, or there exists a value $s^\star \in \mathbb{K}$ held by D, which is secret-shared with respect to the sharing specification $\mathbb{S}$, such that the following hold.*
- *If any honest party outputs its shares at time $T_{\mathsf{VSS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$, then all honest parties output their shares at time T_{VSS}.*
- *If any honest party outputs its shares at time $T > T_{\mathsf{VSS}}$, then every honest party outputs its shares by time $T + 2\Delta$.*

Proof. If *no* honest party obtains any output, then the lemma holds trivially. So, consider the case when some *honest* party, say P_h, obtains an output. We note that *every* honest party participates with *some* input in the instance of Π_{BA} at time $2\Delta + 2T_{\mathsf{BC}}$. Hence, by the $\mathcal{Z}_s$-consistency and $\mathcal{Z}_s$-guaranteed liveness properties of Π_{BA} in the *synchronous* network, the instance of Π_{BA} generates an output at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$ for every honest party. Now there are two possible cases.

1. **The output of Π_{BA} is** 1: In this case, from Lemma 9, it follows that *all honest* parties participated with input 1 at time $2\Delta + 2T_{\mathsf{BC}}$ during the instance of Π_{BA}. This implies that *all honest* parties received $\mathcal{C}_1, \ldots, \mathcal{C}_q$ from the broadcast of D through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$ and accepted these sets. We next claim that corresponding to *every* $S_m \in \mathbb{S}$, *all honest* parties in S_m output some *common* share say $s^{(m)}$, corresponding to S_m, at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Let $s^\star \stackrel{def}{=} s^{(1)} + \ldots + s^{(m)}$. It will then follow that the value $s^\star$ is secret-shared at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.
   The proof of the above claim closely follows the $\mathcal{Z}_s$-correctness proof in the *synchronous* network (see the proof of Lemma 10). Consider an *arbitrary* $S_m \in \mathbb{S}$. Then there are two possible cases.
   - **An $\texttt{NOK}(m, \star)$ message was received at time $2\Delta + T_{\mathsf{BC}}$ through regular-mode from the broadcast of some party in S_m**: In this case, *all honest* parties in S_m will output some *common* share, say $s^{(m)}$, corresponding to S_m at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. The proof for this is *exactly* the same as the proof of Lemma 10 for the same case.
   - **No $\texttt{NOK}(m, \star)$ message was received within time $2\Delta + T_{\mathsf{BC}}$ through regular-mode from the broadcast of any party in S_m**: Let $\mathcal{H}_m$ be the set of *honest* parties in S_m. Since $\mathcal{C}_m$ is accepted by all the honest parties, it follows that the parties in $\mathcal{H}_m \cap \mathcal{C}_m$ constitute a clique in the

consistency graph $G^{(m)}$ of *all* honest parties. This further implies that all the parties in $\mathcal{H}_m \cap \mathcal{C}_m$ received a common share from D corresponding to S_m, say $s^{(m)}$. Moreover, from Lemma 9, *all* the parties in $\mathcal{H}_m \cap \mathcal{C}_m$ would have received $s^{(m)}$ from D, within time Δ. Now similar to the proof for the same case in Lemma 10, it can be concluded that all honest parties output $s^{(m)}$ as the share corresponding to S_m, at time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.

2. **The output of Π_{BA} is** 0: Since P_h has obtained an output, it implies that it has received sets $\mathcal{E}_1, \ldots, \mathcal{E}_q$ from the broadcast of D and accepted them. Let T be the time at which P_h accepted $\mathcal{E}_1, \ldots, \mathcal{E}_q$. Note that $T > 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. This is because from the protocol steps, the *honest* parties start participating in the instance of Π_{BC} of D for broadcasting $\mathcal{E}_1, \ldots, \mathcal{E}_q$, *only after* time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. By the $\mathcal{Z}_s$-consistency and $\mathcal{Z}_s$-fallback consistency of Π_{BC} in the *synchronous* network, *all honest* parties will receive and accept the sets $\mathcal{E}_1, \ldots, \mathcal{E}_q$, latest by time $T + 2\Delta$. Since the parries have accepted $\mathcal{E}_1, \ldots, \mathcal{E}_q$, it implies that corresponding to every $S_m \in \mathbb{S}$, *all* the honest parties in $\mathcal{E}_m$ have received a common value from D, say $s^{(m)}$. We claim that all the honest parties in S_m output $s^{(m)}$ as the share, corresponding to S_m, latest by time $T + 2\Delta$. This will automatically imply that the value $s^\star \stackrel{def}{=} s^{(1)} + \ldots + s^{(m)}$ is secret-shared, latest by time $T + 2\Delta$. The proof for the above claim is exactly the *same* as the proof of Lemma 11, for the case when the output of Π_{BA} is 0 and is omitted.

We finally prove the commitment property in an *asynchronous* network.

Lemma 14. *In protocol Π_{VSS}, if D is corrupt, then either no honest party obtains any output or there exists some value $s^\star \in \mathbb{K}$ held by D, such that almost-surely $s^\star$ is secret-shared, with respect to the sharing specification $\mathbb{S}$.*

Proof. If no honest party obtains an output, then the lemma holds trivially. So, consider the case when some honest party, say P_h, has obtained an output in Π_{VSS}. Note that *every honest* party participates with some input in the instance of Π_{BA} at local time $2\Delta + 2T_{\mathsf{BC}}$. Hence, from the $\mathcal{Z}_a$-almost-surely liveness and $\mathcal{Z}_a$-consistency properties of Π_{BA} in the *asynchronous* network, it follows that the instance of Π_{BA} eventually generates a *common* output for every honest party. Now, there are two possible cases.

1. **The output of Π_{BA} instance is** 1: This implies that all honest parties eventually receive the sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$ from the broadcast of D (either through regular-mode or fallback-mode), and accept these sets. The proof for this is identical to the case "when the output of Π_{BA} instance is 1", in the proof of Lemma 11. This further implies that corresponding to each $S_m \in \mathbb{S}$, all the *honest* parties in $\mathcal{C}_m$ have received a common value from D, say $s^{(m)}$, since the *honest* parties in $\mathcal{C}_m$ constitute a clique. We claim that all the *honest* parties in S_m eventually output $s^{(m)}$ as the share, corresponding to the set S_m. This will automatically imply that the value $s^\star \stackrel{def}{=} s^{(1)} + \ldots + s^{(q)}$ is eventually secret-shared among the parties. The proof for the claim again

closely follows the case "when the output of Π_{BA} instance is 1", in the proof of Lemma 11.

So, consider an *arbitrary honest* $P_i \in S_m$, and let $Z_a^\star$ be the set of *corrupt* parties. From the protocol steps, P_i computes its share corresponding to S_m based on one of the following three conditions. Assuming that at least one of these conditions eventually hold for P_i, we first show that the share computed by P_i corresponding to S_m is bound to be $s^{(m)}$. This is followed by showing that at least one of these conditions eventually hold for P_i.

- **Condition A**: At time $2\Delta + 2T_{\mathsf{BC}}$, there exists some value $s'^{(m)}$ such that party P_i received the $\texttt{Resolve}(m, s'^{(m)})$ message from the broadcast of D, as well as from the broadcast of a subset of parties $\mathcal{C}'_m \subseteq \mathcal{C}_m$ through regular-mode, where $\mathcal{C}_m \setminus \mathcal{C}'_m \in \mathcal{Z}_s$. And consequently P_i sets $s'^{(m)}$ as its share corresponding to S_m.
  We argue that $\mathcal{C}'_m$ is bound to contain *at least* one honest party from $\mathcal{C}_m$, which broadcasts $\texttt{Resolve}(m, s'^{(m)})$ message, where $s'^{(m)} = s^{(m)}$. In more detail, let $S_m \setminus \mathcal{C}_m = Z_\alpha \in \mathcal{Z}_s$ and $\mathcal{C}_m \setminus \mathcal{C}'_m = Z_\beta \in \mathcal{Z}_s$. Also, note that $\mathcal{P} \setminus S_m = Z_m \in \mathcal{Z}_s$. Now if $\mathcal{C}'_m$ *does not* contain any honest party from $\mathcal{C}_m$, it implies that $\mathcal{C}'_m \subseteq Z_a^\star \in \mathcal{Z}_a$. This further implies that $\mathcal{P} \subseteq Z_m \cup Z_\alpha \cup Z_\beta \cup Z_a^\star$, which is a contradiction to the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.
- **Condition B**: At time 2Δ, there exists a subset of parties $\mathcal{C}''_m \subseteq \mathcal{C}_m$, where $\mathcal{C}_m \setminus \mathcal{C}''_m \in \mathcal{Z}_s$, such that P_i received a common value from *all* the parties in $\mathcal{C}''_m$, say $s'^{(m)}$. And consequently, P_i sets $s'^{(m)}$ as its share corresponding to S_m.
  We claim that the subset $\mathcal{C}''_m$ is bound to contain at least one *honest* party from $\mathcal{C}_m$, who would have sent $s'^{(m)} = s^{(m)}$ to P_i within time 2Δ. The proof for the claim is similar to the case above where we have shown that $\mathcal{C}'_m$ is bound to contain at least one *honest* party from $\mathcal{C}_m$.
- **Condition C**: There exists a subset of parties $\mathcal{C}'''_m \subseteq \mathcal{C}_m$ where $\mathcal{C}_m \setminus \mathcal{C}'''_m \in \mathcal{Z}_a$, such that P_i received a common value from *all* the parties in $\mathcal{C}'''_m$, say $s'^{(m)}$. And consequently, P_i sets $s'^{(m)}$ as its share, corresponding to S_m. In this case also, one can show that the subset $\mathcal{C}'''_m$ is bound to contain at least one *honest* party from $\mathcal{C}_m$, who would have sent $s'^{(m)} = s^{(m)}$ to P_i. This is because $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition and every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$.

Thus, we have shown that *irrespective* of the way P_i would have computed its output share corresponding to S_m, it is bound to be the *same* as $s^{(m)}$. Now it is easy to see that at least one of the conditions A, B and C above, eventually holds for P_i. Specially, the condition C is bound to *eventually* hold, irrespective of conditions A and B. This is because the set of *honest* parties in $\mathcal{C}_m$, namely the parties in $\mathcal{C}_m \setminus Z_a^\star$, *always* constitute a candidate $\mathcal{C}'''_m$ set for P_i. This follows from the fact that the share $s^{(m)}$ from *all* the parties in $\mathcal{C}_m \setminus Z_a^\star$ will be eventually delivered to P_i.

2. **The output of Π_{BA} instance is** 0: Since P_h has computed its output, it follows that it has received the sets $\mathcal{E}_1, \ldots, \mathcal{E}_q$ from the broadcast of D,

and accepted these sets. From the $\mathcal{Z}_a$-weak consistency and $\mathcal{Z}_a$-fallback consistency properties of Π_{BC} in the *asynchronous* network, it follows that *all honest* parties eventually receive these sets and accept them. We also note that corresponding to every $S_m \in \mathbb{S}$, *all honest* parties in $\mathcal{E}_m$ received the same share from D, say $s^{(m)}$. Now similar to the proof for the case "the output of Π_{BA} instance is 0" in the proof of Lemma 14, it can be shown that *all honest* parties in S_m eventually set $s^{(m)}$ as the share, corresponding to S_m. This automatically implies that the value $s^\star \stackrel{def}{=} s^{(1)} + \ldots + s^{(q)}$ is eventually secret-shared.

Lemma 15. *Protocol Π_{VSS} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot (n^4 \log |\mathbb{K}| + n^5(\log n + \log |\mathcal{Z}_s|)))$ bits and invokes one instance of Π_{BA}.*

Proof. During phase I, corresponding to every $S_m \in \mathbb{S}$, dealer D needs to send a share, consisting of one element of $\mathbb{K}$, to *all* the parties in S_m. This incurs a total communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n)$ elements from $\mathbb{K}$. During phase II, every party in S_m sends an element from $\mathbb{K}$ to every other party in S_m. This incurs a total communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2)$ elements from $\mathbb{K}$. During phase III, every party in S_m may broadcast an `OK` message for every other party in S_m. This requires broadcasting $\mathcal{O}(|\mathcal{Z}_s| \cdot n^2 \cdot (\log n + \log |\mathcal{Z}_s|))$ bits, as each `OK` message encodes the identity of two parties and the identity of a set from $\mathcal{Z}_s$, requiring $2\log n + \log |\mathcal{Z}_s|$ bits. Additionally, during phase III, corresponding to any $S_m \in \mathbb{S}$, each party in S_m may broadcast an `NOK` message. This requires broadcasting $\mathcal{O}(|\mathcal{Z}_s| \cdot n \cdot (\log |\mathcal{Z}_s| + \log n))$ bits, as each `NOK` message encodes the identity of a party and the identity of a set from $\mathcal{Z}_s$. During phase IV, corresponding to each $S_m \in \mathbb{S}$, up to $\mathcal{O}(n)$ elements from $\mathbb{K}$ may be broadcasted to resolve the conflicts. This incurs a total broadcast of $\mathcal{O}(|\mathcal{Z}_s| \cdot n)$ elements from $\mathbb{K}$. Additionally, D may broadcast sets $\mathcal{C}_1, \ldots, \mathcal{C}_q$, where each set can be encoded by an n-bit vector.

The communication complexity now follows by summing up all the above costs and from the communication complexity of the protocol Π_{BC}, along with the fact that each element from $\mathbb{K}$ can be represented by $\log |\mathbb{K}|$ bits, and the fact that $q = |\mathcal{Z}_s|$.

Remark 2 (**Further improvement in the communication complexity of Π_{VSS}**). The complexity of the phase III can be significantly reduced by making the following modification: party P_i now broadcasts a *single* $\texttt{OK}(i, j)$ message corresponding to P_j, if the pairwise consistency test is positive between P_i's and P_j's share across *all* the sets in $\mathbb{S}$, to which both P_i and P_j belongs. That is, if corresponding to *every* $S_m \in \mathbb{S}$ such that $P_i, P_j \in S_m$, the condition $s_i^{(m)} = s_j^{(m)}$ holds. Consequently, during phase III, only $\mathcal{O}(n^2)$ `OK` messages need to be broadcasted, where the size of each message will now be *only* $\mathcal{O}(\log n)$ bits. This will reduce the communication complexity of Π_{VSS} to $\mathcal{O}(|\mathcal{Z}_s| \cdot n^4 \cdot (\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + n^5 \log n)$ bits, along with one instance of Π_{BA}.

The proof of the following theorem now follows from Lemma 10-15 and Remark 2.

Theorem 4. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network satisfying the following conditions.*
- $\mathcal{Z}_s \neq \mathcal{Z}_a$;
- *For every subset $Z \in \mathcal{Z}_a$, there exists a subset $Z' \in \mathcal{Z}_s$, such that $Z \subseteq Z'$;*
- *$\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.*

Moreover, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then protocol Π_{VSS} achieves the following properties, where D has a private input $s \in \mathbb{K}$ for Π_{VSS}.
- *If D is honest, then the following hold.*
  - $\mathcal{Z}_s$**-correctness***: In a synchronous network, s is secret-shared with respect to $\mathbb{S}$, at time $T_{\mathsf{VSS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.*
  - $\mathcal{Z}_a$**-correctness***: In an asynchronous network, almost-surely, s is eventually secret-shared, with respect to $\mathbb{S}$.*
  - **Privacy***: The view of $\mathcal{A}$ remains independent of s, irrespective of the network type.*
- *If D is corrupt, then either no honest party obtains any output or there exists some $s^\star \in \mathbb{K}$, such that the following hold.*
  - $\mathcal{Z}_a$**-commitment***: In an asynchronous network, almost-surely, $s^\star$ is eventually secret-shared, with respect to $\mathbb{S}$.*
  - $\mathcal{Z}_s$**-commitment***: In a synchronous network, $s^\star$ is secret-shared, with respect to $\mathbb{S}$, such that the following hold.*
    - *If any honest party outputs its shares at time T_{VSS}, then all honest parties output their shares at time T_{VSS}.*
    - *If any honest party outputs its shares at time $T > T_{\mathsf{VSS}}$, then every honest party outputs its shares by time $T + 2\Delta$.*
- **Communication Complexity***: The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^4 \cdot (\log|\mathbb{K}| + \log|\mathcal{Z}_s| + \log n) + n^5 \log n)$ bits and invokes one instance of Π_{BA}.*

E Properties of the Preprocessing Phase Protocol

In this section, we formally present our preprocessing phase protocol and prove its properties. We first start with the description of our ACS protocol and proof of its properties.

E.1 Protocol Π_{ACS} and Its Properties

For simplicity, we present Π_{ACS} when $L = 1$; the modifications for a general L are straightforward.

Protocol $\Pi_{\mathsf{ACS}}(\mathcal{Q})$

- **Phase I — Secret Sharing the Input**: If $P_i \in \mathcal{Q}$, the do the following.
  - On having the input x_i, act as a dealer D and invoke an instance $\Pi_{\mathsf{VSS}}^{(i)}$ of Π_{VSS} with input x_i.

- Participate in the instance $\Pi_{\mathsf{VSS}}^{(j)}$ invoked by every $P_j \in \mathcal{Q}$, and wait for time T_{VSS}. Initialize a set $\mathcal{C}_i = \emptyset$ after time T_{VSS} and include $P_j \in \mathcal{Q}$ in $\mathcal{C}_i$ if an output is computed in the instance $\Pi_{\mathsf{VSS}}^{(j)}$.

- **Phase II — Identifying the Common Set of Selected Parties**:
  - Corresponding to every $P_j \in \mathcal{Q}$, participate in an instance of $\Pi_{\mathsf{BA}}^{(j)}$ of Π_{BA} with input 1, if $P_j \in \mathcal{C}_i$.
  - Once 1 has been obtained as the output from instances of Π_{BA} corresponding to a set of parties in $\mathcal{Q} \setminus Z$ for some $Z \in \mathcal{Z}_s$, participate with input 0 in all the Π_{BA} instances $\Pi_{\mathsf{BA}}^{(j)}$, such that $P_j \in \mathcal{Q}$ and $P_j \notin \mathcal{C}_i$.
  - Once all the instances of Π_{BA} corresponding to the parties in $\mathcal{Q}$ have produced a binary output, then output $\mathcal{CS}$, which is the set of parties $P_j \in \mathcal{Q}$ such that 1 is obtained as the output in the instance $\Pi_{\mathsf{BA}}^{(j)}$.

Fig. 8: Agreement on a common subset of parties. The above code is for each $P_i \in \mathcal{P}$.

We next prove the properties of the protocol Π_{ACS}, assuming $L = 1$.

Lemma 16. *. Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network satisfying the following conditions.*

- $\mathcal{Z}_s \neq \mathcal{Z}_a$;
- *For every subset $Z \in \mathcal{Z}_a$, there exists a subset $Z' \in \mathcal{Z}_s$, such that $Z \subseteq Z'$;*
- *$\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.*

Moreover, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Furthermore, let $\mathcal{Q} \subseteq \mathcal{P}$, such that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ either satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition or the $\mathcal{Q}^{(3,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Then, Π_{ACS} achieves the following, where every (honest) $P_i \in \mathcal{Q}$ has input $x_i \in \mathbb{K}$ for Π_{ACS}.

- $\mathcal{Z}_s$**-correctness***: If the network is synchronous, then at time $T_{\mathsf{ACS}} \stackrel{def}{=} T_{\mathsf{VSS}} + 2T_{\mathsf{BA}}$, all honest parties output a common subset of parties $\mathcal{CS} \subseteq \mathcal{Q}$, where $\mathcal{Q} \setminus \mathcal{CS} \in \mathcal{Z}_s$, such that the following hold.*
  - *All honest parties from $\mathcal{Q}$ will be present in $\mathcal{CS}$.*
  - *Corresponding to every $P_j \in \mathcal{CS}$, there exists some $x_j^\star \in \mathbb{K}$, where $x_j^\star = x_j$ for an honest P_j, such that $x_j^\star$ is secret-shared with respect to $\mathbb{S}$.*
- $\mathcal{Z}_a$**-correctness***: If the network is asynchronous, then almost-surely, the honest parties eventually output a subset $\mathcal{CS} \subseteq \mathcal{Q}$ where $\mathcal{Q} \setminus \mathcal{CS} \in \mathcal{Z}_s$. Moreover, corresponding to every $P_j \in \mathcal{CS}$, there exists an $x_j^\star \in \mathbb{K}$, where $x_j^\star = x_j$ for an honest P_j, such that $x_j^\star$ is eventually secret-shared with respect to $\mathbb{S}$.*
- **Privacy***: Irrespective of the network type, the view of the adversary remains independent of the inputs x_i corresponding to the honest parties $P_i \in \mathcal{Q}$.*
- **Communication Complexity***: The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot (\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + n^6 \log n)$ bits and invokes $\mathcal{O}(n)$ instances of Π_{BA}.*

Proof. The *privacy* property simply follows from the *privacy* property of Π_{VSS}, while *communication complexity* follows from the *communication complexity* of Π_{VSS} and the fact that up to $\mathcal{O}(n)$ instances of Π_{VSS} may be involved, since $|\mathcal{Q}| = \mathcal{O}(n)$. We next prove the *correctness* property.

We first consider a *synchronous* network. Let $Z_s^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties, and let $\mathcal{H} \stackrel{def}{=} \mathcal{Q} \setminus Z_s^\star$ be the set of *honest* parties. We note that $\mathcal{H} \neq \emptyset$, since it is given that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ *either* satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition *or* $\mathcal{Q}^{(3,1)}(\mathcal{Q}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Corresponding to each $P_j \in \mathcal{H}$, every *honest* P_i obtains the output $\{[x_j]_m\}_{P_i \in S_m}$ at time T_{VSS} during $\Pi_{\mathsf{VSS}}^{(j)}$, which follows from the $\mathcal{Z}_s$-correctness of Π_{VSS} in the *synchronous* network. Consequently, at time T_{VSS}, the set $\mathcal{C}_i$ of *every honest* P_i will satisfy the condition $\mathcal{Q} \setminus \mathcal{C}_i \in \mathcal{Z}_s$. This is because $\mathcal{H} \subseteq \mathcal{C}_i$ will hold at time T_{VSS}. Now, corresponding to each $P_j \in \mathcal{H}$, each $P_i \in \mathcal{H}$ starts participating with input 1 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ at time T_{VSS}. Hence, from the $\mathcal{Z}_s$-validity and $\mathcal{Z}_s$-guaranteed liveness properties of Π_{BA} in the *synchronous* network, it follows that at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, every $P_i \in \mathcal{H}$ obtains the output 1 during the instance $\Pi_{\mathsf{BA}}^{(j)}$ corresponding to every $P_j \in \mathcal{H}$. Consequently, at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, every *honest* party will start participating in the remaining Π_{BA} instances for which no input has been provided yet (if there are any), and from the $\mathcal{Z}_s$-guaranteed liveness and $\mathcal{Z}_s$-consistency properties of Π_{BA} in the *synchronous* network, these Π_{BA} instances will produce common outputs for every honest party at time $T_{\mathsf{ACS}} = T_{\mathsf{VSS}} + 2T_{\mathsf{BA}}$. Hence, at time T_{ACS}, every honest party outputs a common $\mathcal{CS}$, where $\mathcal{Q} \setminus \mathcal{CS} \in \mathcal{Z}_s$, and where each $P_j \in \mathcal{H}$ will be present in $\mathcal{CS}$. We next wish to show that corresponding to *every* $P_j \in \mathcal{CS}$, there exists some value which is secret-shared among the parties with respect to $\mathbb{S}$.

Consider an *arbitrary* party $P_j \in \mathcal{CS}$. If $P_j \in \mathcal{H}$, then as argued above, every $P_i \in \mathcal{H}$ computes the shares $\{[x_j]_m\}_{P_i \in S_m}$ at time T_{VSS} itself. Next, consider a *corrupt* $P_j \in \mathcal{CS}$. Since $P_j \in \mathcal{CS}$, it follows that the instance $\Pi_{\mathsf{BA}}^{(j)}$ produces the output 1 for all honest parties. This further implies that at least one *honest* P_i must have computed some output during the instance $\Pi_{\mathsf{VSS}}^{(j)}$ by time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$ (implying that $P_j \in \mathcal{C}_i$) and participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(j)}$. This is because if, at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, party P_j *does not* belong to the $\mathcal{C}_i$ set of *any* honest P_i, then it implies that *all honest* parties start participating with input 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$. Then, from the $\mathcal{Z}_s$-validity of Π_{BA} in the *synchronous* network, every honest party would have obtained the output 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ and hence, P_j will not be present in $\mathcal{CS}$, which is a contradiction. Now, if P_i has computed some output during $\Pi_{\mathsf{VSS}}^{(j)}$ at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, then from the $\mathcal{Z}_s$-commitment of Π_{VSS} in the *synchronous* network, it follows that there exists some value $x_j^\star$, such that $x_j^\star$ will be secret-shared with respect to $\mathbb{S}$ by time $T_{\mathsf{VSS}} + T_{\mathsf{BA}} + 2\Delta$. Since $2\Delta < T_{\mathsf{BA}}$, it follows that at time T_{ACS}, every $P_i \in \mathcal{H}$ has $\{[x_j^\star]_m\}_{P_i \in S_m}$, thus proving the *correctness* property in a *synchronous* network.

We next consider an *asynchronous* network. Let $Z_a^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties, and let $\mathcal{H} \stackrel{def}{=} \mathcal{Q} \setminus \mathcal{Z}_a^\star$ be the set of *honest* parties. Notice that $\mathcal{Q} \setminus \mathcal{H} \in \mathcal{Z}_s$, since every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$. Now, irrespective of the way messages are scheduled, there will eventually be subset of parties $\mathcal{Q} \setminus Z$ for some $Z \in \mathcal{Z}_s$, such that all the parties in $\mathcal{H}$ participate with input 1 in the instances of Π_{BA} corresponding to the parties in $\mathcal{Q} \setminus Z$. This is because corresponding to every $P_j \in \mathcal{H}$, every $P_i \in \mathcal{H}$ *eventually* computes an output during the instance $\Pi_{\mathsf{VSS}}^{(j)}$, which follows from the $\mathcal{Z}_a$-correctness of Π_{VSS} in the *asynchronous* network. So, even if the *corrupt* parties P_j do not invoke their respective $\Pi_{\mathsf{VSS}}^{(j)}$ instances, there will be a set of Π_{BA} instances corresponding to the parties in $\mathcal{Q} \setminus Z$ for some $Z \in \mathcal{Z}_s$ in which all the parties in $\mathcal{H}$ eventually participate with input 1. Consequently, from the $\mathcal{Z}_a$-almost-surely liveness and $\mathcal{Z}_a$-consistency properties of Π_{BA} in the *asynchronous* network, these Π_{BA} instances eventually produce the output 1 for all the parties in $\mathcal{H}$. Hence, all the parties in $\mathcal{H}$ eventually participate with some input in the remaining Π_{BA} instances, which almost-surely produce some output for every honest party eventually. From the properties of Π_{BA} in the *asynchronous* network, it then follows that all the honest parties output the same $\mathcal{CS}$.

Now, consider an *arbitrary* $P_j \in \mathcal{CS}$. It implies that the honest parties computed the output 1 during the instance $\Pi_{\mathsf{BA}}^{(j)}$, which further implies that at least one *honest* P_i participated with input 1 in $\Pi_{\mathsf{BA}}^{(j)}$ after computing its output in the instance $\Pi_{\mathsf{VSS}}^{(j)}$. If P_j is *honest*, then the $\mathcal{Z}_a$-correctness of Π_{VSS} in the *asynchronous* network guarantees that x_j will be *eventually* secret-shared with respect to $\mathbb{S}$, during $\Pi_{\mathsf{VSS}}^{(j)}$. On the other hand, if P_j is *corrupt*, then the $\mathcal{Z}_a$-commitment of Π_{VSS} in the *asynchronous* network guarantees that there exists some $x_j^\star \in \mathbb{K}$, such that $x_j^\star$ will *eventually* be secret-shared with respect to $\mathbb{S}$, during $\Pi_{\mathsf{VSS}}^{(j)}$.

We next discuss the modifications needed in the protocol Π_{ACS} when each party in $\mathcal{Q}$ has L inputs.

Π_{ACS} for L Inputs: Protocol Π_{ACS} can be easily extended if each party has L inputs. Now, each P_j shares L values through instances of Π_{VSS}. Moreover, the parties participate with input 1 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ if they have computed some output in *all* the L instances of Π_{VSS} invoked by P_j as a dealer. The rest of the protocol steps remain the same. The protocol will now incur a communication of $\mathcal{O}(L \cdot |\mathcal{Z}_s| \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s|) + n^6 \log n)$ bits and invokes $\mathcal{O}(n)$ instances of Π_{BA}.

E.2 Multiplication Protocol and Its Properties

In this section, we formally present our multiplication protocol Π_{Mult} and prove its properties. For simplicity, we first consider the case when $L = 1$, and present the formal details of the protocol in Fig 9.

Protocol $\Pi_{\mathsf{Mult}}([a], [b])$

Let $[a]_1, \ldots, [a]_q$ and $[b]_1, \ldots, [b]_q$ be the shares, corresponding to $[a]$ and $[b]$ respectively, where every $P_i \in \mathcal{P}$ holds the shares $\{[a]_m, [b]_m\}_{P_i \in S_m}$.

- For every ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$, the parties do the following to compute $[a_l \cdot b_m]$, where $a_l \stackrel{def}{=} [a]_l$ and $b_m \stackrel{def}{=} [b]_m$.

– Let $\mathcal{Q}_{l,m} \stackrel{def}{=} S_l \cap S_m$. The parties execute an instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$ of Π_{ACS}, where every $P_i \in \mathcal{Q}_{l,m}$ participates with the input $a_l \cdot b_m$.
– Let $\mathcal{R}_{l,m} \subseteq \mathcal{Q}_{l,m}$ be the common subset of parties obtained as the output from the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, where $\mathcal{Q}_{l,m} \setminus \mathcal{R}_{l,m} \in \mathcal{Z}_s$. Let $r \stackrel{def}{=} |\mathcal{R}_{l,m}|$ and let $\mathcal{R}_{l,m} = \{P_{\alpha_1}, \ldots, P_{\alpha_r}\}$. Moreover, corresponding to $P_{\alpha_i} \in \mathcal{R}_{l,m}$, let v_i be the value which is secret-shared on the behalf of P_{α_i} during $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$.[a].
– The parties publicly check whether $v_1, \ldots, v_r$ are all equal. For this, the parties locally compute $r-1$ differences $[d_1] \stackrel{def}{=} [v_1] - [v_2], \ldots, [d_{r-1}] \stackrel{def}{=} [v_1] - [v_r]$. This is followed by publicly reconstructing the differences $d_1, \ldots, d_{r-1}$ by invoking instances of Π_{Rec}, and checking if all of them are 0.
  – If $d_1 = \ldots = d_{r-1} = 0$, then the parties set $[a_l \cdot b_m] = [v_1]$.
  – Else, the parties publicly reconstruct $[a]_l$ and $[b]_m$ by invoking instances $\Pi_{\mathsf{Rec}}([a]_l, \mathbb{S})$ and $\Pi_{\mathsf{Rec}}([b]_m, \mathbb{S})$ of Π_{Rec}. The parties then set $[a_l \cdot b_m]$ to the default sharing of $a_l \cdot b_m$, where $[a_l \cdot b_m]_1 = a_l \cdot b_m$ and $[a_l \cdot b_m]_2 = \ldots = [a_l \cdot b_m]_q = 0$.[b]

– The parties output $[a \cdot b] = \sum_{(l,m) \in \{1,\ldots,q\} \times \{1,\ldots,q\}} [a_l \cdot b_m]$.

[a] If P_{α_i} is *honest*, then $v_i = a_l \cdot b_m$ holds.
[b] The vector of shares $(s, 0, \ldots, 0)$ can be considered as a default sharing of any given $s \in \mathbb{K}$.

Fig. 9: The perfectly-secure multiplication protocol

We next prove the properties of Π_{Mult}. We first start with a helping lemma.

Lemma 17. *In protocol Π_{Mult}, the following hold for every ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$.*
– *$\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}_{l,m}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, where $\mathcal{Q}_{l,m} = S_l \cap S_m$.*
– *In a synchronous network, at time T_{ACS}, all honest parties will compute a set $\mathcal{R}_{l,m} \subseteq \mathcal{Q}_{l,m}$, where $\mathcal{Q}_{l,m} \setminus \mathcal{R}_{l,m} \in \mathcal{Z}_s$, such that $\mathcal{R}_{l,m}$ contains all honest parties from $\mathcal{Q}_{l,m}$.*
– *In an asynchronous network, almost-surely, all honest parties will eventually compute a set $\mathcal{R}_{l,m} \subseteq \mathcal{Q}_{l,m}$, where $\mathcal{Q}_{l,m} \setminus \mathcal{R}_{l,m} \in \mathcal{Z}_s$, such that $\mathcal{R}_{l,m}$ contains at least one honest party from $\mathcal{Q}_{l,m}$.*

Proof. The first property follows from the fact that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, and $\mathcal{Q}_{l,m} \stackrel{def}{=} S_l \cap S_m = \mathcal{P} \setminus (Z_l \cup Z_m)$. Hence, if $\mathcal{Z}_s$ and $\mathcal{Z}_a$ *do not* satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, then it implies that there exist sets $Z_\alpha \in \mathcal{Z}_s$ and $Z_\beta \in \mathcal{Z}_a$ such that $\mathcal{Q}_{l,m} \subseteq Z_\alpha \cup Z_\beta$. This further implies that $Z_\alpha \cup Z_\beta \cup Z_l \cup Z_m \subseteq \mathcal{P}$, which contradicts the fact that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.

For proving the second property, we consider a *synchronous* network. Let $Z_s^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties. Since $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}_{l,m}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, from the $\mathcal{Z}_s$-correctness of Π_{ACS} in the *synchronous* network, it follows that all honest parties will compute $\mathcal{R}_{l,m}$ as the output of the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, such that $\mathcal{Q}_{l,m} \setminus \mathcal{R}_{l,m} \in \mathcal{Z}_s$. Moreover, *all* the parties from $\mathcal{Q}_{l,m} \setminus Z_s^\star$ will be present in $\mathcal{R}_{l,m}$. This proves the second property.

We next consider an *asynchronous* network. Let $Z_a^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties. From the $\mathcal{Z}_a$-correctness of Π_{ACS} in the *asynchronous* network, it follows that, almost-surely, all honest parties will eventually compute $\mathcal{R}_{l,m}$ as the output of the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$ such that $\mathcal{Q}_{l,m} \setminus \mathcal{R}_{l,m} \in \mathcal{Z}_s$. Moreover, $\mathcal{R}_{l,m} \not\subset Z_a^\star$, as otherwise, $\mathcal{Z}_s$ and $\mathcal{Z}_a$ *do not* satisfy the $\mathcal{Q}^{(1,1)}(\mathcal{Q}_{l,m}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, which is a contradiction. Consequently, $\mathcal{R}_{l,m}$ will consists of *at least* one honest party from $\mathcal{Q}_{l,m}$.

We now proceed to prove the properties of Π_{Mult}.

Lemma 18. *In a synchronous network, all honest parties output* $[c]$ *within time* $T_{\mathsf{Mult}} = T_{\mathsf{ACS}} + 2\Delta$*, where* $c = a \cdot b$*. Moreover, the view of the adversary remains independent of* a *and* b*.*

Proof. To prove the lemma, we claim that for each ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$, all honest parties securely compute a secret-sharing of the summand $[a]_l \cdot [b]_m$, within time $T_{\mathsf{ACS}} + 2\Delta$, without revealing any additional information to the adversary. The proof then follows from the fact that the following holds:

$$[c] = [a \cdot b] = \sum_{(l,m) \in \{1,\ldots,q\} \times \{1,\ldots,q\}} [[a]_l \cdot [b]_m].$$

We now proceed to prove our claim, for which we consider an *arbitrary* ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$.

From Lemma 17, at time T_{ACS}, all honest parties will compute the set $\mathcal{R}_{l,m}$ as the output of the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, where $\mathcal{R}_{l,m} \subseteq \mathcal{Q}_{l,m}$, and where *all honest* parties from $\mathcal{Q}_{l,m}$ will be present in $\mathcal{R}_{l,m}$. Let $|\mathcal{R}_{l,m}| = r$ and $\mathcal{R}_{l,m} = \{P_{\alpha_1}, \ldots, P_{\alpha_r}\}$. Moreover, without loss of generality, let P_{α_1} be *honest*. From the $\mathcal{Z}_s$-correctness of Π_{ACS} in the *synchronous* network, corresponding to *every* $P_{\alpha_j} \in \mathcal{R}_{l,m}$, there exists some value, v_j, which will be secret-shared among the parties on the behalf of P_{α_j} during the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$. Moreover, since P_{α_1} is assumed to be *honest*, from the protocol steps, $v_1 = [a]_l \cdot [b]_m$.

Now there are two possible cases:

- *Every party in* $\mathcal{R}_{l,m}$ *participates with input* $[a]_l \cdot [b]_m$ *during* $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$: In this case, $v_1 = v_2 = \ldots = v_r = [a]_l \cdot [b]_m$ and hence $d_1 = \ldots = d_{r-1} = 0$. From the properties of Π_{Rec}, within time $T_{\mathsf{ACS}} + \Delta$, the honest parties reconstruct the $r-1$ differences $d_1, \ldots, d_{r-1}$ and find all of them to be 0. Hence, they set $[[a]_l \cdot [b]_m]$ to $[v_1]$, where v_1 is the same as $[a]_l \cdot [b]_m$. The privacy in this case follows from privacy of Π_{ACS} and the fact that the adversary only learns the $r-1$ differences, which are all 0.
- *Some party in* $\mathcal{R}_{l,m}$ *participates with an input, which is different from* $[a]_l \cdot [b]_m$*, during* $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$: Let $P_i \in \mathcal{R}_{l,m}$ be a corrupt party, corresponding to which v_i is shared during $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, where $v_i \neq [a]_l \cdot [b]_m$. Since $v_i \neq v_1$, it follows that at least one of the $r-1$ differences $d_1, \ldots, d_{r-1}$ will be non-zero, and the parties detect the same when these differences are publicly reconstructed at time $T_{\mathsf{ACS}} + \Delta$. In this case, the parties reconstruct the shares $[a]_l$ and $[b]_m$ at time $T_{\mathsf{ACS}} + 2\Delta$, and take a default secret-sharing

of $[a]_l \cdot [b]_m$. The privacy in this case follows from the fact that since there exists a *corrupt* party in $\mathcal{R}_{l,m}$, the adversary is already aware of the shares $[a]_l$ and $[b]_m$ and hence, publicly reconstructing these values does not add any new information to the view of the adversary.

We next consider prove the properties in an *asynchronous* network.

Lemma 19. *In an asynchronous network, almost-surely, all honest parties eventually output* $[c]$*, where* $c = a \cdot b$*. Moreover, the view of the adversary remains independent of* a *and* b*.*

Proof. The proof is very similar to the proof of Lemma 18. Namely, we show that almost-surely, corresponding to each ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$, all honest parties eventually and securely compute a secret-sharing of the summand $[a]_l \cdot [b]_m$. For this, we consider an *arbitrary* ordered pair $(l, m) \in \{1, \ldots, q\} \times \{1, \ldots, q\}$. From Lemma 17, almost-surely, all honest parties will eventually compute a set $\mathcal{R}_{l,m}$ as the output of the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, where $\mathcal{R}_{l,m} \subseteq \mathcal{Q}_{l,m}$, and where *at least* one *honest* party from $\mathcal{Q}_{l,m}$ will be present in $\mathcal{R}_{l,m}$. Let $|\mathcal{R}_{l,m}| = r$ and $\mathcal{R}_{l,m} = \{P_{\alpha_1}, \ldots, P_{\alpha_r}\}$. Moreover, without loss of generality, let P_{α_1} be *honest*. From the $\mathcal{Z}_a$-correctness of Π_{ACS} in the *asynchronous* network, corresponding to *every* $P_{\alpha_j} \in \mathcal{R}_{l,m}$, there exists some value, v_j, which will be eventually secret-shared among the parties on the behalf of P_{α_j}, during the instance $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$. Moreover, since P_{α_1} is assumed to be *honest*, from the protocol steps, $v_1 = [a]_l \cdot [b]_m$.

Now there are two possible cases:

- *Every party in* $\mathcal{R}_{l,m}$ *participates with input* $[a]_l \cdot [b]_m$ *during* $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$: In this case, $v_1 = v_2 = \ldots = v_r = [a]_l \cdot [b]_m$ and hence $d_1 = \ldots = d_{r-1} = 0$. From the properties of Π_{Rec}, the honest parties eventually reconstruct the $r - 1$ differences $d_1, \ldots, d_{r-1}$ and find all of them to be 0. Hence they set $[[a]_l \cdot [b]_m]$ to $[v_1]$, where v_1 is the same as $[a]_l \cdot [b]_m$. The privacy in this case follows from privacy of Π_{ACS}, and the fact that the adversary only learns $r - 1$ differences which are all 0.
- *Some party in* $\mathcal{R}_{l,m}$ *participates with an input, which is different from* $[a]_l \cdot [b]_m$*, during* $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$: Let $P_i \in \mathcal{R}_{l,m}$ be a corrupt party, corresponding to which v_i is shared during $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, where $v_i \neq [a]_l \cdot [b]_m$. Since $v_i \neq v_1$, it follows that at least one of the $r - 1$ differences $d_1, \ldots, d_{r-1}$ will be non-zero, and the parties detect the same when these differences are eventually reconstructed. In this case, the parties eventually reconstruct the shares $[a]_l$ and $[b]_m$, and take a default secret-sharing of $[a]_l \cdot [b]_m$. The privacy in this case follows from the fact that since there exists a *corrupt* party in $\mathcal{R}_{l,m}$, the adversary is already aware of the shares $[a]_l$ and $[b]_m$ and hence, publicly reconstructing these values does not add any new information to the view of the adversary.

The proof of Lemma 20 now easily follows from Lemma 18 and Lemma 19. The communication complexity follows from the fact that $q^2 = |\mathcal{Z}_s|^2$ instances of Π_{ACS} are executed.

Lemma 20. *Let $\mathcal{A}$ be an adversary, characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network, satisfying the conditions* Con *(see Condition 1 in Section 2). Moreover, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then protocol Π_{Mult} achieves the following properties, where the inputs of the parties are $[a]$ and $[b]$.*

- $\mathcal{Z}_s$**-correctness**: *In a synchronous network, all honest parties output $[c]$ within time $T_{\mathsf{Mult}} = T_{\mathsf{ACS}} + 2\Delta$, where $c = a \cdot b$.*
- $\mathcal{Z}_a$**-correctness**: *In an asynchronous network, almost-surely, the honest parties eventually output $[c]$, where $c = a \cdot b$.*
- **Privacy**: *Irrespective of the network type, the view of the adversary remains independent of a and b.*
- **Communication Complexity**: *Π_{Mult} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^3 \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ bits and invokes $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ instances of Π_{BA}.*

We next discuss the modifications needed in the protocol Π_{Mult} to handle the case when $L > 1$.

Protocol Π_{Mult} for L Pairs of Inputs: If the input for Π_{Mult} is $\{([a^{(\ell)}], [b^{(\ell)}])\}_{\ell=1,\ldots L}$, then during the instance of $\Pi_{\mathsf{ACS}}(\mathcal{Q}_{l,m})$, each party in $\mathcal{Q}_{l,m}$ will have to share L summands. Similarly, corresponding to $\mathcal{R}_{l,m}$, the parties reconstruct $(|\mathcal{R}_{l,m}| - 1) \cdot L$ number of difference values. The rest of the protocol steps remain the same. With these modification, Π_{Mult} will now incur a communication of $\mathcal{O}(L \cdot |\mathcal{Z}_s|^3 \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ bits and invokes $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ instances of Π_{BA}.

E.3 Protocol $\Pi_{\mathsf{PreProcessing}}$ and Its Properties

In this section, we formally present our preprocessing phase protocol $\Pi_{\mathsf{PreProcessing}}$ and prove its properties. For the sake of simplicity, we first explain the protocol to generate one random secret-shared multiplication-triple. The protocol is presented in Fig 10.

Protocol $\Pi_{\mathsf{PreProcessing}}$

- **Generating random pairs of values**: The parties do the the following.
  - Participate in an instance $\Pi_{\mathsf{ACS}}(\mathcal{P})$ of Π_{ACS}, where the input of each P_i is a random pair of values $(a_i, b_i) \in \mathbb{K}$.
  - Let $\mathcal{R}$ be the output of the instance $\Pi_{\mathsf{ACS}}(\mathcal{P})$, where $\mathcal{P} \setminus \mathcal{R} \in \mathcal{Z}_s$. Moreover, corresponding to $P_i \in \mathcal{R}$, let $(a_i^\star, b_i^\star) \in \mathbb{K}$ be the pair of values such that the parties hold $[a_i^\star]$ and $[b_i^\star]$ during the instance $\Pi_{\mathsf{ACS}}(\mathcal{P})$.
  - The parties locally compute $[a] = \sum_{P_i \in \mathcal{R}} [a_i^\star]$ and $[b] = \sum_{P_i \in \mathcal{R}} [b_i^\star]$.
- **Computing the product**: The parties compute $[c]$ by executing $\Pi_{\mathsf{Mult}}([a], [b])$.
  - The parties output $([a], [b], [c])$.

Fig. 10: The preprocessing phase protocol for generating a secret-sharing of one random multiplication-triple

We next prove the properties of the protocol $\Pi_{\mathsf{PreProcessing}}$.

Lemma 21. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network, satisfying the conditions* Con *(see Condition 1 in Section 2). Moreover, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \text{ and } Z_m \in \mathcal{Z}_s\}$. Then $\Pi_{\mathsf{PreProcessing}}$ achieves the following.*

- $\mathcal{Z}_s$**-correctness***: In a synchronous network, all honest parties output $([a], [b], [c])$ within time $T_{\mathsf{PreProcessing}} = T_{\mathsf{ACS}} + T_{\mathsf{Mult}}$, where $c = a \cdot b$.*
- $\mathcal{Z}_a$**-correctness***: In an asynchronous network, almost-surely, the honest parties eventually output $([a], [b], [c])$, where $c = a \cdot b$.*
- **Privacy***: Irrespective of the network type, the view of the adversary remains independent of a, b and c.*
- **Communication Complexity***: $\Pi_{\mathsf{PreProcessing}}$ incurs a communication of $\mathcal{O}(L \cdot |\mathcal{Z}_s|^3 \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ bits and invokes $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ instances of Π_{BA}.*

Proof. We first consider a *synchronous* network. From the $\mathcal{Z}_s$-correctness of Π_{ACS} in the *synchronous* network, it follows that at time T_{ACS}, all honest parties will output a common subset of parties $\mathcal{R} \subseteq \mathcal{P}$, where $\mathcal{P} \setminus \mathcal{R} \in \mathcal{Z}_s$. Moreover, corresponding to every $P_i \in \mathcal{R}$, there will be a pair of values $(a_i^\star, b_i^\star) \in \mathbb{K}$, which will be secret-shared with respect to $\mathbb{S}$ on the behalf of P_i, during the instance of Π_{ACS}. Furthermore, $(a_i^\star, b_i^\star) = (a_i, b_i)$ for every *honest* $P_i \in \mathcal{R}$. Also, if $P_i \in \mathcal{R}$ is *honest*, then the privacy property of Π_{ACS} guarantees that the view of the adversary remains independent of (a_i, b_i). The $\mathcal{Z}_s$-correctness property of Π_{ACS} in the *synchronous* network also guarantees that *all* honest parties from $\mathcal{P}$ will be present in $\mathcal{R}$. Now, since the *honest* parties in $\mathcal{P}$ secret-share random pairs of values during Π_{ACS}, it follows that (a, b) will be random from the point of view of the adversary. Finally, the $\mathcal{Z}_s$-correctness property of Π_{Mult} in the *synchronous* network guarantees that the parties output $([a], [b], [c])$ at time $T_{\mathsf{ACS}} + T_{\mathsf{Mult}}$, where $c = a \cdot b$. Moreover, the privacy property of Π_{Mult} in the *synchronous* network guarantees that adversary does not learn any additional information about a, b and c (except that $c = a \cdot b$), during the instance of Π_{Mult} as well.

The proof of the properties in an *asynchronous* network is almost the same as above, except that we now use the $\mathcal{Z}_a$-correctness property of Π_{ACS} in the *asynchronous* network (which guarantees that the parties eventually compute a common $\mathcal{R}$, containing *at least* one honest party) and the $\mathcal{Z}_a$-correctness property of Π_{Mult} in the *asynchronous* network. To avoid repetition, we do not give the formal details.

The communication complexity follows from the communication complexity of Π_{ACS} and Π_{Mult}.

We next discuss the modifications needed in the protocol $\Pi_{\mathsf{PreProcessing}}$ to generate c_M number of secret-shared multiplication-triples.

$\Pi_{\mathsf{PreProcessing}}$ for Generating c_M Random Multiplication-triples: To generate secret-sharing of c_M random multiplication-triples, the instance of Π_{ACS}

during the first stage is executed by setting $L = c_M$. Consequently, there will be c_M pairs of secret-shared values generated on the behalf of *each* party in $\mathcal{R}$. Moreover, the pairs of values shared by the *honest* parties in $\mathcal{R}$ will be random from the point of view of the adversary. Consequently, summing the secret-sharing of the pairs of values shared by the parties in $\mathcal{R}$, leads to c_M pairs of random values being secret-shared during the *first* stage. Then, during the *second* stage, the parties execute an instance of Π_{Mult}, with the input being the c_M pairs of random secret-shared pairs of values from the first stage. This securely leads to a secret-sharing of the product of each pair. The resultant protocol will now incur a communication of $\mathcal{O}(c_M \cdot |\mathcal{Z}_s|^3 \cdot n^5(\log |\mathbb{K}| + \log |\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ bits, and invokes $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ instances of Π_{BA}.

F The Circuit-Evaluation Protocol and Its Properties

Protocol Π_{CirEval} for securely evaluating the circuit cir is presented in Fig 11.

Protocol $\Pi_{\mathsf{CirEval}}(\mathsf{cir}, \mathcal{Z}_s, \mathcal{Z}_a)$

- **Preprocessing and Input-Sharing** — The parties do the following:
  - Each $P_i \in \mathcal{P}$, on having the input x_i for f, participates in the instance $\Pi_{\mathsf{ACS}}(\mathcal{P})$ of Π_{ACS} with input x_i. Let $\mathcal{CS}$ be the common subset of parties obtained as the output during $\Pi_{\mathsf{ACS}}(\mathcal{P})$, where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$. Corresponding to every $P_j \notin \mathcal{CS}$, set $x_j = 0$, and set $[x_j]$ to the default secret-sharing, where $[x_j]_1 = [x_j]_2 = \ldots = [x_j]_q = 0$.
  - Participate in an instance of $\Pi_{\mathsf{PreProcessing}}$ to generate c_M number of secret-shared, random multiplication-triples $\{[\mathbf{a}^{(j)}], [\mathbf{b}^{(j)}], [\mathbf{c}^{(j)}]\}_{j=1,\ldots,c_M}$.
- **Circuit Evaluation** — Let $G_1, \ldots, G_m$ be a publicly-known topological ordering of the gates of cir. For $k = 1, \ldots, m$, the parties do the following for gate G_k:
  - *If G_k is an addition gate*: the parties locally compute $[w] = [u] + [v]$, where u and v are gate-inputs, and w is the gate-output.
  - *If G_k is a multiplication-with-a-constant gate with constant c*: the parties locally compute $[v] = c \cdot [u]$, where u is the gate-input, and v is the gate-output.
  - *If G_k is an addition-with-a-constant gate with constant c*: the parties locally compute $[v] = c + [u]$, where u is the gate-input, and v is the gate-output.
  - *If G_k is a multiplication gate*: Let G_k be the ℓ^{th} multiplication gate in cir, where $\ell \in \{1, \ldots, c_M\}$, and let $([a^{(\ell)}], [b^{(\ell)}], [c^{(\ell)}])$ be the ℓ^{th} shared multiplication-triple generated during $\Pi_{\mathsf{PreProcessing}}$. Moreover, let $[u]$ and $[v]$ be the shared gate-inputs of G_k. Then, the parties participate in an instance $\Pi_{\mathsf{Beaver}}(([u], [v]), ([a^{(\ell)}], [b^{(\ell)}], [c^{(\ell)}]))$ of Π_{Beaver}, and obtain $[w]$, where $w = u \cdot v$.
- **Output Computation** — Let $[y]$ be the secret-shared circuit-output. The parties participate in an instance $\Pi_{\mathsf{Rec}}(y, \mathbb{S})$ of Π_{Rec} and reconstruct y.
- **Termination**: Each P_i does the following.
  - If y has been obtained during output computation, then send the message $(\texttt{ready}, y)$ to all the parties.

– If the message $(\texttt{ready}, y)$ is received from a set of parties $\mathcal{C}$, where $\mathcal{C} \notin \mathcal{Z}_s$, then send $(\texttt{ready}, y)$ message to all the parties if no `ready` message is sent earlier.
– If the message $(\texttt{ready}, y)$ is received from a set of parties $\mathcal{P} \setminus Z$, for some $Z \in \mathcal{Z}_s$, then terminate all sub-protocols, output y, and terminate.

Fig. 11: A best-of-both-worlds perfectly-secure protocol for securely evaluating the arithmetic circuit cir

Theorem 5. *Let $\mathcal{A}$ be an adversary characterized by an adversary structure $\mathcal{Z}_s$ in a synchronous network and adversary structure $\mathcal{Z}_a$ in an asynchronous network satisfying the conditions* Con *(see Condition 1 in Section 2). Moreover, let $f : \mathbb{K}^n \rightarrow \mathbb{K}$ be a function represented by an arithmetic circuit* cir *over $\mathbb{K}$, consisting of c_M number of multiplication gates, with a multiplicative depth of D_M and where each party P_i has an input $x_i \in \mathbb{K}$ for f. Furthermore, let $\mathbb{S} = \{S_m : S_m = \mathcal{P} \setminus Z_m \;\; and \;\; Z_m \in \mathcal{Z}_s\}$. Then, Π_{CirEval} incurs a communication of $\mathcal{O}(c_M \cdot |\mathcal{Z}_s|^3 \cdot n^5(\log|\mathbb{K}| + \log|\mathcal{Z}_s| + \log n) + |\mathcal{Z}_s|^2 \cdot n^6 \log n)$ bits, invokes $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n)$ instances of Π_{BA}, and achieves the following.*

- *In a synchronous network, all honest parties output $y = f(x_1, \ldots, x_n)$ at time $(30n + D_M + 6k + 38) \cdot \Delta$, where $x_j = 0$ for every $P_j \notin \mathcal{CS}$, such that $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$ and every honest party from $\mathcal{P}$ will be present in $\mathcal{CS}$; here k is the constant from Lemma 2, as determined by the underlying (existing) perfectly-secure ABA protocol Π_{ABA}.*
- *In an asynchronous network, almost-surely, the honest parties eventually output $y = f(x_1, \ldots, x_n)$, where $x_j = 0$ for every $P_j \notin \mathcal{CS}$, and where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$.*
- *The view of the adversary remains independent of the inputs of the honest parties in $\mathcal{CS}$.*

Proof. Consider a *synchronous* network. Let $Z_s^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties, and let $\mathcal{H}_s \stackrel{def}{=} \mathcal{P} \setminus Z_s^\star$ be the set of *honest* parties. From the $\mathcal{Z}_s$-correctness property of $\Pi_{\mathsf{PreProcessing}}$ in the *synchronous* network, at time $T_{\mathsf{PreProcessing}}$, the (honest) parties will have c_M number of secret-shared multiplication-triples, shared with respect to $\mathbb{S}$, from the instance of $\Pi_{\mathsf{PreProcessing}}$. From the $\mathcal{Z}_s$-correctness property of Π_{ACS} in the *synchronous* network, at time T_{ACS}, the (honest) parties will have a common subset $\mathcal{CS}$ from the instance of Π_{ACS}, where *all honest* parties will be present in $\mathcal{CS}$, and where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$. Moreover, corresponding to *every* $P_j \in \mathcal{CS}$, there will be some $x_j \in \mathbb{K}$ held by P_j (which will be the same as P_j's input for f for an *honest* P_j), such that x_j will be secret-shared with respect to $\mathbb{S}$. As $\mathcal{CS}$ will be known *publicly*, the parties take a default secret-sharing of 0 on the behalf of the parties P_j outside $\mathcal{CS}$, by considering $x^{(j)} = 0$. Since $T_{\mathsf{ACS}} < T_{\mathsf{PreProcessing}}$, it follows that at time $T_{\mathsf{PreProcessing}}$, the parties will hold a secret-sharing of c_M multiplication-triples and secret-sharing of $x_1, \ldots, x_n$.

The circuit-evaluation will take $D_M \cdot \Delta$ time. This follows from the fact that linear gates are evaluated locally (non-interactively), while all the *independent* multiplication gates can be evaluated in parallel by running the corresponding instances of Π_{Beaver} in *parallel*, where each such instance requires Δ time.

From the $\mathcal{Z}_s$-correctness property of Π_{Beaver} in the *synchronous* network, the multiplication-gates will be evaluated correctly and hence, during the output-computation phase, the parties will hold a secret-sharing of y (with respect to $\mathbb{S}$), where $y = f(x_1, \ldots, x_n)$. From the properties of Π_{Rec}, it will take Δ time for every party to reconstruct y. Hence, during the termination phase, *all* honest parties will send a `ready` message for y. Since $\mathcal{P} \setminus \mathcal{H}_s \in \mathcal{Z}_s$, every honest party will then terminate with output y at time $T_{\mathsf{PreProcessing}} + (D_M + 2) \cdot \Delta$. By substituting the values of $T_{\mathsf{PreProcessing}}, T_{\mathsf{VSS}}, T_{\mathsf{ACS}}, T_{\mathsf{BC}}, T_{\mathsf{BA}}, T_{\mathsf{SBA}}$ and T_{ABA} and by noting that all instances of Π_{BC} in Π_{CirEval} are invoked with $\mathcal{Z} = \mathcal{Z}_s$, we get that the parties terminate the protocol at time $(D_M + 30n + 6k + 38) \cdot \Delta$, where k is the constant from Lemma 2, as determined by the underlying (existing) perfectly-secure ABA protocol Π_{ABA}.

If we consider an *asynchronous* network, then the proof is similar as above, except that we now use the security properties of the building blocks $\Pi_{\mathsf{PreProcessing}}$, $\Pi_{\mathsf{ACS}}, \Pi_{\mathsf{Beaver}}$ and Π_{Rec} in the *asynchronous* network. Let $Z_a^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties, and let $\mathcal{H}_a \stackrel{def}{=} \mathcal{P} \setminus Z_a^\star$ be the set of *honest* parties. During the termination phase, the parties in $Z_a^\star$ may send `ready` messages for y', where $y' \neq y$. Since every subset in $\mathcal{Z}_a$ is a subset of some subset in $\mathcal{Z}_s$, it follows that no honest party will terminate with output y', where $y' \neq y$. On the other hand, all the parties in $\mathcal{H}_a$ will eventually compute the output y, and will send a `ready` message for y, which is eventually delivered to every honest party. Now, consider an *honest* party P_h, who terminates with output y. We wish to show that every honest party eventually terminates the protocol with output y. This is because P_h must have received `ready` messages for y from a subset of parties $\mathcal{P} \setminus Z_\alpha$, for some Z_α. Since $\mathcal{Z}_s$ satisfies the $\mathcal{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ condition, it follows that that $\mathcal{H}_a \cap (\mathcal{P} \setminus Z_\alpha) \notin \mathcal{Z}_s$. Now the `ready` messages for y from the set of parties $\mathcal{H}_a \cap (\mathcal{P} \setminus Z_\alpha)$ are eventually delivered to *every* honest party. Consequently, irrespective of which stage of the protocol an honest party is in, every party in $\mathcal{H}_a$ (including P_h) eventually sends a `ready` message for y, which is eventually delivered. Since $\mathcal{P} \setminus \mathcal{H}_a \in \mathcal{Z}_s$ (as every subset in $\mathcal{Z}_a$ is a subset of some subset from $\mathcal{Z}_s$), this implies that every honest party eventually terminates with output y.

From the privacy property of Π_{ACS}, corresponding to every *honest* $P_j \in \mathcal{CS}$, the input x_j will be random from the point of view of the adversary. Moreover, from the privacy property of $\Pi_{\mathsf{PreProcessing}}$, the multiplication-triples generated through $\Pi_{\mathsf{PreProcessing}}$ will be random from the point of view of the adversary. During the evaluation of linear gates, no interaction happens among the parties and hence, no additional information about the inputs of the honest parties is revealed. The same is true during the evaluation of multiplication-gates as well, which follows from the privacy property of Π_{Beaver}.

The communication complexity of the protocol follows from the communication complexity of $\Pi_{\mathsf{PreProcessing}}, \Pi_{\mathsf{ACS}}$ and Π_{Beaver}.