Network Agnostic MPC with Statistical Security

Ananya Appan⋆[1] and Ashish Choudhury[2]

[1] SAP Labs, Bangalore India
ananya.appan@iiitb.ac.in
[2] IIIT Bangalore, India
ashish.choudhury@iiitb.ac.in

Abstract. In this work, we initiate the study of network agnostic MPC protocols with *statistical* security. Network agnostic MPC protocols give the best possible security guarantees, *irrespective* of the underlying network type. While network agnostic MPC protocols have been designed earlier with *perfect* and *computational* security, *nothing* is known in the literature regarding the possibility of network agnostic MPC protocols with statistical security. We consider the *general-adversary* model, where the adversary is characterized by an *adversary structure*, which enumerates all possible candidate subsets of corrupt parties. Given an unconditionally-secure PKI setup (a.k.a pseudo-signature setup), known statistically-secure *synchronous* MPC (SMPC) protocols are secure against adversary structures satisfying the $\mathbb{Q}^{(2)}$ condition, meaning that the union of *any two* subsets from the adversary structure *does not* cover the entire set of parties. On the other hand, known statistically-secure *asynchronous* MPC (AMPC) protocols can tolerate $\mathbb{Q}^{(3)}$ adversary structures where the union of *any three* subsets from the adversary structure *does not* cover the entire set of parties.
Fix a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and adversary structures $\mathcal{Z}_s$ and $\mathcal{Z}_a$, satisfying the $\mathbb{Q}^{(2)}$ and $\mathbb{Q}^{(3)}$ conditions respectively, where $\mathcal{Z}_a \subset \mathcal{Z}_s$. Then given an unconditionally-secure PKI, we ask whether it is possible to design a statistically-secure MPC protocol, which is resilient against $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in a *synchronous* and an *asynchronous* network respectively, even if the parties in $\mathcal{P}$ are *unaware* of the network type. We show that it is possible iff $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}$ condition, meaning that the union of any two subsets from $\mathcal{Z}_s$ and any one subset from $\mathcal{Z}_a$ is a proper subset of $\mathcal{P}$. Enroute our MPC protocol, we design several important network agnostic building blocks with the $\mathbb{Q}^{(2,1)}$ condition, such as Byzantine *broadcast*, *Byzantine agreement* (BA), *information checking protocol* (ICP), *verifiable secret-sharing* (VSS) and secure multiplication protocol, whose complexity is polynomial in n and $|\mathcal{Z}_s|$.

1 Introduction

A secure *multiparty computation* (MPC) protocol [53,36,13,52] allows a set of n mutually distrusting parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ with private inputs to securely compute any known function f of their inputs. This is achieved even if a subset

⋆ Work done as a student at IIIT Bangalore.

of the parties are under the control of a centralized *adversary* and behave *maliciously* in a Byzantine fashion during the protocol execution. In any MPC protocol, the parties need to interact over the underlying communication network. Two types of networks have been predominantly considered. The more popular *synchronous* MPC (SMPC) protocols operate over a synchronous network, where every message sent is delivered within a *known* Δ time. Hence, if a receiving party does not receive an expected message within Δ time, then it knows that the corresponding sender party is *corrupt*. The synchronous model *does not* capture real world networks like the Internet appropriately, where messages can be *arbitrarily* delayed. Such networks are better modelled by the *asynchronous* communication model [19]. In any *asynchronous* MPC (AMPC) protocol [12,14], there are *no* timing assumptions on message delays and messages can be arbitrarily, yet finitely delayed. The only guarantee is that every message sent will be *eventually* delivered. The major challenge here is that no participant will know how long it has to wait for an expected message and *cannot* distinguish a "slow" sender party from a corrupt sender party. Consequently, in any AMPC protocol, a party *cannot* afford to receive messages from all the parties, to avoid an endless wait. Hence, to make "progress", as soon a party receives messages from a "subset" of the parties, it has to process them as per the protocol, thus ignoring messages from a subset of potentially non-faulty but slow parties.

SMPC protocols are relatively simpler and enjoy better *fault-tolerance* (which is the maximum number of faults tolerable) compared to AMPC protocols. However, SMPC protocols become completely insecure even if a *single* message (from a non-faulty party) gets delayed. AMPC protocols do not suffer from this shortcoming. On the negative side, AMPC protocols are far more complex than SMPC protocols and enjoy poor fault-tolerance. Moreover, every AMPC protocol suffers from *input deprivation* [10] where, to avoid an endless wait, inputs of *all* non-faulty parties *may not* be considered for the computation of f.

Network Agnostic MPC Protocols. There is a third category of protocols called *network agnostic* MPC protocols, where the parties *will not* be knowing the network type and the protocol should provide the best possible security guarantees depending upon the network type. Such protocols are practically motivated, since the parties *need not* have to worry about the network type.

1.1 Our Motivation and Results

One of the earliest demarcations made in the literature is to categorize MPC protocols based on the computing power of the underlying adversary. The two main categories are *unconditionally-secure* protocols, which remain secure even against *computationally-unbounded* adversaries, and *conditionally-secure* MPC protocols (also called *cryptographically-secure*), which remain secure *only* against *computationally-bounded* adversaries [53,36]. Unconditionally-secure protocols can be further categorized as *perfectly-secure* [13,12] or *statistically-secure* [52,14], depending upon whether the security guarantees are *error-free* or achieved except with a *negligible* probability. The fault-tolerance of statistically-secure MPC protocols are significantly *better* compared to perfectly-secure protocols. The above demarcation carries over even for network agnostic MPC protocols. While perfectly-secure and cryptographically-secure network agnostic MPC protocols have been investigated earlier, *nothing* is known regarding network agnostic statistically-secure MPC protocols. In this work we derive necessary and sufficient condition for such protocols for the *first* time.

Existing Results for Statistically-Secure MPC. Consider the *threshold* setting, where the maximum number of corrupt parties under the adversary's control is upper bounded by a given threshold. In this model, it is known that statistically-secure SMPC tolerating up to t_s faulty parties is possible iff $t_s < n/2$ [52], provided the parties are given some *unconditionally-secure* PKI (a.k.a *pseudo-signature* setup) [50,34].[3] On the other hand, statistically-secure AMPC tolerating up to t_a faulty parties is possible iff $t_a < n/3$ [14,2].

A more generalized form of corruption is the general adversary model (also called *non-threshold* model) [38]. Here, the adversary is specified through a publicly known *adversary structure* $\mathcal{Z} \subset 2^{\mathcal{P}}$, which is the set of all subsets of potentially corruptible parties during the protocol execution. The adversary is allowed to choose any one subset from $\mathcal{Z}$ for corruption. There are several "merits" of studying the general adversary model. For example, it provides more flexibility to model corruption in a fine-grained fashion. A threshold adversary is always a "special" type of non-threshold adversary. Consequently, a protocol in the non-threshold setting *always* implies a protocol in the threshold setting. Also, the protocols in this model are relatively simpler and based on simpler primitives, compared to protocols against threshold adversaries based on complex properties of bivariate polynomials. The downside is that the complexity of the protocols in the non-threshold model is polynomial in n and $|\mathcal{Z}|$, where the latter could be $\mathcal{O}(2^n)$ in the *worst* case. In fact, as noted in [38,39], this is *unavoidable*.

Following [38], given a subset of parties $\mathcal{P}' \subseteq \mathcal{P}$ and $\mathcal{Z}$, we say that $\mathcal{Z}$ satisfies the $\mathbb{Q}^{(k)}(\mathcal{P}', \mathcal{Z})$ condition, if the union of any k subsets from $\mathcal{Z}$ *does not* "cover" $\mathcal{P}'$. That is, for any subsets $Z_{i_1}, \ldots, Z_{i_k} \in \mathcal{Z}$, the condition $(Z_{i_1} \cup \ldots \cup Z_{i_k}) \subset \mathcal{P}'$ holds. In the non-threshold model, statistically-secure SMPC is possible if the underlying adversary structure $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition, provided the parties are given an unconditionally-secure PKI setup [40], while statistically-secure AMPC requires the underlying adversary structure $\mathcal{Z}_a$ to satisfy the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition [40,5].

Our Results for Network Agnostic Statistically-Secure MPC. We consider the most generic form of corruption and ask the following question:

Given an unconditionally-secure PKI, a *synchronous adversary structure* $\mathcal{Z}_s$ and an *asynchronous adversary structure* $\mathcal{Z}_a$ satisfying the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ conditions respectively, where $\mathcal{Z}_a \subset \mathcal{Z}_s$, does there exist a statistically-secure MPC protocol, which remains secure against $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in a synchronous and an asynchronous network respectively?

[3] The setup realizes unconditionally-secure Byzantine agreement [49] with $t_s < n/2$.

We answer the above question affirmatively, iff $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, where by $\mathbb{Q}^{(k,k')}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, we mean that for any $Z_{i_1}, \ldots, Z_{i_k} \in \mathcal{Z}_s$ and $\mathsf{Z}_{j_1}, \ldots, \mathsf{Z}_{j_{k'}} \in \mathcal{Z}_a$, the following holds:

$$(Z_{i_1} \cup \ldots \cup Z_{i_k} \cup \mathsf{Z}_{j_1} \cup \ldots \cup \mathsf{Z}_{j'_k}) \subset \mathcal{P}.$$

Our results when applied against *threshold* adversaries imply that given an unconditionally-secure PKI, and thresholds $0 < t_a < \frac{n}{3} < t_s < \frac{n}{2}$, network agnostic statistically-secure MPC tolerating t_s and t_a corruptions in the *synchronous* and *asynchronous* network is possible, iff $2t_s + t_a < n$ holds. Our results in the context of relevant literature are summarized in Table 1.

Network Type	Corruption Scenario	Security	Condition	Reference
Synchronous	Threshold (t)	Perfect	$t < n/3$	[13]
Synchronous	Non-threshold ($\mathcal{Z}$)	Perfect	$\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$	[38]
Synchronous	Threshold (t)	Statistical	$t < n/2$	[52]
Synchronous	Non-threshold ($\mathcal{Z}$)	Statistical	$\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z})$	[40]
Asynchronous	Threshold (t)	Perfect	$t < n/4$	[12]
Asynchronous	Non-threshold ($\mathcal{Z}$)	Perfect	$\mathbb{Q}^{(4)}(\mathcal{P}, \mathcal{Z})$	[42]
Asynchronous	Threshold (t)	Statistical	$t < n/3$	[14,2]
Asynchronous	Non-threshold ($\mathcal{Z}$)	Statistical	$\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z})$	[5]
Network Agnostic	Threshold (t_s, t_a)	Perfect	$0 < t_a < n/4 < t_s < n/3$ and $3t_s + t_a < n$	[3]
Network Agnostic	Non-threshold ($\mathcal{Z}_s, \mathcal{Z}_a$)	Perfect	$\mathcal{Z}_a \subset \mathcal{Z}_s, \mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s), \mathbb{Q}^{(4)}(\mathcal{P}, \mathcal{Z}_a)$ and $\mathbb{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$	[4]
Network Agnostic	Threshold (t_s, t_a)	Computational	$0 < t_a < n/3 < t_s < n/2$ and $2t_s + t_a < n$	[17]
Network Agnostic	Non-threshold ($\mathcal{Z}_s, \mathcal{Z}_a$)	Statistical	$\mathcal{Z}_a \subset \mathcal{Z}_s, \mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s), \mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ and $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$	This work
Network Agnostic	Threshold (t_s, t_a)	Statistical	$0 < t_a < n/3 < t_s < n/2$ and $2t_s + t_a < n$	This work

Table 1: Various conditions for MPC in different settings

1.2 Detailed Technical Overview

We perform shared *circuit-evaluation* [13,52], where f is abstracted as an arithmetic circuit ckt over a finite field $\mathbb{F}$ and the goal is to securely evaluate each gate in ckt in a *secret-shared* fashion. For every value during the circuit-evaluation, each party holds a share, such that the shares of the corrupt parties *do not* reveal any additional information. Once the function output is secret-shared, it is publicly reconstructed. We deploy a *linear* secret-sharing scheme, which enables the parties to evaluate linear gates in ckt in a *non-interactive* fashion. *Non-linear* gates are evaluated using Beaver's method [8] by deploying secret-shared random *multiplication-triples* which are generated *beforehand*.

To instantiate the above approach with *statistical* security, we need the following ingredients: a *Byzantine agreement* (BA) protocol [49], an *information checking protocol* (ICP) [52], a *verifiable secret sharing* (VSS) protocol [22], a *reconstruction* protocol and finally, a secure *multiplication* protocol. All existing statistically-secure SMPC [52,26,9,37,40] and AMPC [14,48,24,25,5] protocols have instantiations of the above building blocks, *either* in the synchronous *or* asynchronous setting. However, in a network agnostic setting, we face several challenges to instantiate the above building blocks. We now take the reader through a detailed tour of the technical challenges and how we deal with them.

1.2.1 Network Agnostic BA with $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ Condition

A BA protocol [49] allows the parties in $\mathcal{P}$ with private input bits to agree on a common output bit (*consistency*), which is the input of the non-faulty parties, if they have the *same* input bit (*validity*). Given an unconditionally-secure PKI, *synchronous* BA (SBA) is possible iff the underlying adversary structure $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition [50,33,34], while *asynchronous* BA (ABA) requires the underlying adversary structure $\mathcal{Z}_a$ to satisfy the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition [23]. Existing SBA protocols become completely *insecure* in an asynchronous network. On the other hand, any ABA protocol becomes *insecure* when executed in a synchronous network, since $\mathcal{Z}_s$ *need not* satisfy the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_s)$ condition. Hence, we design a network agnostic BA protocol with $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. The protocol is obtained by generalizing the blueprint for network agnostic BA against *threshold* adversaries, first proposed in [15] and later used in [3]. While [15] proposed it for *computational security* with conditions $0 < t_a < \frac{n}{3} < t_s < \frac{n}{2}$ and $2t_s + t_a < n$ in the presence of a *computationally-secure* PKI, later, [3] modified it for *perfect security* and used it with conditions $t_a, t_s < n/3$.[4] We replace the computationally-secure PKI with an *unconditionally-secure* PKI and generalize the building blocks of [15] against non-threshold adversaries and upgrade their security to unconditional-security. Additionally, we also generalize certain primitives from [3] and adapt them to work with the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Since this part mostly follows the existing works, we refer to Section 3 for full details.

1.2.2 Network Agnostic ICP with $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ Condition

An ICP [52,26] is used for authenticating data in the presence of a *computationally-unbounded* adversary. In an ICP, there are *four* entities, a *signer* $\mathsf{S} \in \mathcal{P}$, an *intermediary* $\mathsf{I} \in \mathcal{P}$, a *receiver* $\mathsf{R} \in \mathcal{P}$ and all the parties in $\mathcal{P}$ acting as *verifiers* (note that S, I and R also act as verifiers). An ICP has two sub-protocols, one for the *authentication phase* and one for the *revelation phase*.

In the authentication phase, S has a private input $s \in \mathbb{F}$, which it distributes to I along with some *authentication information*. Each verifier is provided with some *verification information*, followed by the parties verifying whether S has distributed "consistent" information. If the verification is "successful", then the data held by I at the end of this phase is called S*'s IC-Signature on* s *for intermediary* I *and receiver* R, denoted by $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$. Later, during the revelation

[4] *Unlike* computationally-secure BA, the necessary condition for perfectly-secure BA is $t < n/3$ for *both* SBA as well as ABA, where t is the maximum number of faults.

phase, I reveals $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to R, who "verifies" it with respect to the verification information provided by the verifiers and either accepts or rejects s. We require the same security guarantees from ICP as expected from cryptographic signatures, namely *correctness* (if S, I and R are *all* honest, then R should accept s), *unforgeability* (a *corrupt* I should fail to reveal an *honest* S's signature on $s' \neq s$) and *non-repudiation* (if an *honest* I holds some $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$, then later an *honest* R should accept s, even if S is *corrupt*). Additionally, we need *privacy*, guaranteeing that if S, I and R are *all* honest, then Adv does not learn s.[5]

The *only known* instantiation of ICP in the *synchronous* network [40] is secure against $\mathbb{Q}^{(2)}$ adversary structures and becomes *insecure* in the *asynchronous* setting. On the other hand, the *only known* instantiation of ICP in the *asynchronous* setting [5] can tolerate *only* $\mathbb{Q}^{(3)}$ adversary structures. Our network agnostic ICP is a careful adaptation of the *asynchronous* ICP of [5]. We first try to *naively* adapt the ICP to deal with the *network agnostic* setting, followed by the technical problems in the naive adaptation and the modifications needed.

During authentication phase, S embeds s in a random t-degree polynomial $F(x)$ at $x = 0$, where t is the cardinality of the maximum-sized subset in $\mathcal{Z}_s$, and gives $F(x)$ to I. In addition, each verifier P_i is given a random *verification-point* (α_i, v_i) on $F(x)$. To let the parties securely verify that it has distributed consistent information, S additionally distributes a random t-degree polynomial $M(x)$ to I, while each verifier P_i is given a point on $M(x)$ at α_i. Each verifier, upon receiving its verification-points, *publicly* confirms the same. Upon receiving these confirmations, I identifies a a subset of *supporting verifiers* $\mathcal{SV}$ which have confirmed the receipt of their verification-points. To avoid an endless wait, I waits until $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$. After this, the parties *publicly* check the consistency of the $F(x), M(x)$ polynomials and the points distributed to $\mathcal{SV}$, with respect to a *random* linear combination of these polynomials and points, where the linear combiner is selected by I. This ensures that S has *no* knowledge beforehand about the random combiner and hence, any "inconsistency" will be detected with a high probability. If no inconsistency is detected, the parties proceed to the revelation phase, where I reveals $F(x)$ to R, while each verifier in $\mathcal{SV}$ reveals its verification-point to R, who accepts $F(x)$ (and hence $F(0)$) if it sure that the verification point of at least one *non-faulty* verifier in $\mathcal{SV}$ is "consistent" with the revealed $F(x)$. This would ensure that the revealed $F(x)$ is indeed correct with a high probability, since a *corrupt* I will have no information about the verification point of any *non-faulty* verifier in $\mathcal{SV}$, provided S is *non-faulty*. To avoid an endless wait, once R finds a subset of verifiers $\mathcal{SV}' \subseteq \mathcal{SV}$, where $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$, whose verification-points are found to be "consistent" with $F(x)$, it outputs $F(0)$.

[5] IC-signatures are *different* from pseudo-signatures. Pseudo-signatures are "transferable", where a party can transfer a signed message to other parties for verification (depending upon the allowed level of transferability), while IC-signatures *can* be verified only by the designated R and *cannot* be further transferred. Due to the same reason, IC-signatures satisfy the privacy property unlike pseudo-signatures. Most importantly, IC-signatures are generated from the scratch, assuming a pseudo-signature setup which is used to instantiate the instances of broadcast in the ICP.

A Technical Problem and Way-out. The protocol outlined above will achieve all the properties in an *asynchronous* network, due to the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition. However, it *fails* to satisfy the *unforgeability* property in a *synchronous* network. Namely, a *corrupt* I may *not* include *all* the non-faulty verifiers in $\mathcal{SV}$ and may purposely *exclude* a subset of *non-faulty* verifiers belonging to $\mathcal{Z}_s$. Let $\mathcal{H}_{\mathcal{SV}}$ be the set of *non-faulty* verifiers in $\mathcal{SV}$ and let $\mathcal{C}_{\mathcal{SV}}$ be the set of *corrupt* verifiers in $\mathcal{SV}$. Due to the above strategy, the condition $\mathbb{Q}^{(1)}(\mathcal{H}_{\mathcal{SV}}, \mathcal{Z}_s)$ *may not* be satisfied and $\mathcal{SV} \setminus \mathcal{C}_{\mathcal{SV}} = \mathcal{H}_{\mathcal{SV}} \in \mathcal{Z}_s$ *may hold*. As a result, during the revelation phase, I may produce an *incorrect* $F'(x) \neq F(x)$ and the verifiers in $\mathcal{C}_{\mathcal{SV}}$ may change their verification points to "match" $F'(x)$, while only the verification points of the verifiers in $\mathcal{H}_{\mathcal{SV}}$ may turn out to be inconsistent with $F'(x)$. Consequently, $\mathcal{SV}' = \mathcal{C}_{\mathcal{SV}}$ and if $\mathcal{H}_{\mathcal{SV}} \in \mathcal{Z}_s$, then clearly $s' = F'(0)$ will be the output of R, thus breaking the unforgeability property.

To deal with the above issue, we let S identify and announce $\mathcal{SV}$. This ensures that *all* honest verifiers are present in $\mathcal{SV}$, if S is *honest* and the network is *synchronous*, provided S waits for "sufficient" time to let the verifiers announce the receipt of their verification points. Consequently, *now* the condition $\mathbb{Q}^{(1)}(\mathcal{H}_{\mathcal{SV}}, \mathcal{Z}_s)$ *will be* satisfied. Hence, if a *corrupt* I reveals an incorrect $F(x)$, then it will not be accepted, as the condition $\mathcal{SV} \setminus \mathcal{C}_{\mathcal{SV}} \in \mathcal{Z}_s$ *no* longer holds.

Linearity of ICP. Our ICP satisfies the *linearity* property (which will be useful later in our VSS), provided "special care" is taken while generating the IC-signatures. Consider a *fixed* S, I and R and let s_a and s_b be two values, such that I holds $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a)$ and $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$, where *all* the following conditions are satisfied during the underlying instances of the authentication phase.

- The set of supporting verifiers $\mathcal{SV}$ are the *same* during both the instances.
- For $i = 1, \ldots, n$, corresponding to the verifier P_i, signer S uses the *same* α_i, to compute the verification points, during both the instances.
- I uses the *same* linear combiner to verify the consistency of the distributed data in both the instances.

Let $s \stackrel{def}{=} c_1 \cdot s_a + c_2 \cdot s_b$, where c_1, c_2 are *publicly known* constants from $\mathbb{F}$. It then follows that if all the above conditions are satisfied, then I can *locally* compute $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ from $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a)$ and $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$, while each verifier in $\mathcal{SV}$ can *locally* compute their corresponding verification information.

1.2.3 Network Agnostic VSS and Reconstruction

In the network agnostic setting, to ensure privacy, all the values during the circuit evaluation need to be secret-shared "with respect" to $\mathcal{Z}_s$ *irrespective* of the network type. We follow the notion of *additive secret-sharing* [41], also used in the earlier MPC protocols [44,40,5]. Given $\mathcal{Z}_s = \{Z_1, \ldots, Z_{|\mathcal{Z}_s|}\}$, we consider the *sharing specification* $\mathbb{S}_{\mathcal{Z}_s} = \{S_1, \ldots, S_{|\mathcal{Z}_s|}\}$, where each $S_q = \mathcal{P} \setminus Z_q$. Hence there exists at least one subset $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$ which *does not* contain any faulty party, *irrespective* of the network type (since $\mathcal{Z}_a \subset \mathcal{Z}_s$). A value $s \in \mathbb{F}$ is said to be secret-shared, if there exist shares $s_1, \ldots, s_{|\mathcal{Z}_s|}$ which sum up to s, such that all (non-faulty) parties in S_q have the share s_q. We denote a secret-sharing of s by $[s]$, with $[s]_q$ denoting the share corresponding to S_q. If $[s]_1, \ldots, [s]_{|\mathcal{Z}_s|}$ are randomly chosen, then the probability distribution of the shares learnt by the adversary will be independent of s, since at least one share will be *missing* for the adversary. We also note that the above secret-sharing is *linear* since, given secret-sharings $[a]$ and $[b]$ and publicly known constants $c_1, c_2 \in \mathbb{F}$, the condition $c_1 \cdot [a] + c_2 \cdot [b] = [c_1 \cdot a + c_2 \cdot b]$ holds. Consequently, the parties can *non-interactively* compute any publicly known linear function of secret-shared values. Unfortunately, the above secret-sharing *does not* allow for the *robust* reconstruction of a secret-shared value. This is because the corrupt parties may produce *incorrect* shares at the time of reconstruction. To deal with this, we "augment" the above secret-sharing. As part of secret-sharing s, we *also* have publicly known *core-sets* $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$, where each $W_q \subseteq S_q$ such that $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition (ensuring $\mathcal{W}_q$ has at least one *non-faulty* party). Moreover, each (non-faulty) $P_i \in \mathcal{W}_q$ will have the IC-signature $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ of every $P_j \in \mathcal{W}_q$, for every $P_k \notin S_q$, such that the underlying IC-signatures satisfy the *linearity* property.

We call this augmented secret sharing as *linear secret-sharing with IC-signatures*, which is still denoted as $[s]$. Now to *robustly* reconstruct a secret-shared s, we ask the parties in $\mathcal{W}_q$ to make public the share $[s]_q$, along with the IC-signatures of *all* the parties in $\mathcal{W}_q$ on $[s]_q$. Any party P_k can then verify whether $[s]_q$ revealed by P_i is *correct* by verifying the IC-signatures. If P_i is *corrupt* then, due to the *unforgeability* if ICP, it will fail to forge IC-signature of a *non-faulty* P_j on an *incorrect* $[s]_q$. On the other hand, a *non-faulty* P_i will be able to reveal the correct $[s]_q$ and the IC-signature of *every* $P_j \in \mathcal{W}_q$ on $[s]_q$, which are accepted even if P_j is *corrupt* (follows from *non-repudiation* of ICP).

We design a network agnostic VSS protocol Π_{VSS}, which allows a designated *dealer* $\mathsf{D} \in \mathcal{P}$ with input $s \in \mathbb{F}$ to *verifiably* generate $[s]$, where s remains private for a *non-faulty* s. If D is *faulty* then either no non-faulty party obtains any output (if D *does not* invoke the protocol) or there exists *some* $s^\star \in \mathbb{F}$ such that the parties output $[s^\star]$. To design Π_{VSS}, we use certain ideas from the statistically-secure *synchronous* VSS (SVSS) and *asynchronous* VSS (AVSS) of [40] and [5] respectively, along with some new counter-intuitive ideas. In the sequel, we first give a brief outline of the SVSS and AVSS of [40,5], followed by the technical challenges arising in the network agnostic setting and how we deal with them.

Statistically-Secure SVSS of [40] with $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ Condition. The SVSS of [40] proceeds as a sequence of *synchronized* phases. During the *first* phase, D picks random shares $s_1, \ldots, s_{|\mathcal{Z}_s|}$ which sum up to s and sends s_q to the parties in S_q. To verify whether D has distributed consistent shares to the parties in S_q, during the *second* phase, every pair of parties $P_i, P_j \in S_q$ exchange the supposedly common shares received from D, along with their respective IC-signatures. That is P_i, upon receiving s_{qi} from D, gives $\{\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})\}_{P_k \in \mathcal{P}}$ to P_j while P_j, upon receiving s_{qj} from D, gives $\{\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})\}_{P_k \in \mathcal{P}}$ to P_i. Then during the *third* phase, the parties in S_q *publicly* complain about any "inconsistency", in response to which D makes *public* the share s_q corresponding to S_q during the *fourth* phase. Hence, by the end of *fourth* phase it is ensured that, for every S_q, either the share s_q is *publicly* known (if any complaint was reported for S_q) or all (non-faulty) parties in S_q have the same share (along with the respective IC-signatures of each other on it). The privacy of s is maintained for a *non-faulty* D, since the share s_q corresponding to the set S_q consisting of *only non-faulty* parties is *never* made public.

Statistically-Secure AVSS of [5] with $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ Condition. Let $\mathcal{Z}_a = \{\mathsf{Z}_1, \ldots, \mathsf{Z}_{|\mathcal{Z}_a|}\}$ and $\mathbb{S}_{\mathcal{Z}_a} = \{\mathsf{S}_1, \ldots, \mathsf{S}_{|\mathcal{Z}_a|}\}$, where each $\mathsf{S}_q = \mathcal{P} \setminus \mathsf{Z}_q$. The AVSS protocol of [5] also follows an idea similar to the SVSS of [40]. However, now the parties *cannot* afford to wait for *all* the parties in S_q to report the statuses of pairwise consistency tests, as the corrupt parties in S_q may *never* turn up. Hence *instead* of looking for *inconsistencies* in S_q, the parties *rather* check how many parties in S_q are reporting the *pairwise consistency* of their supposedly common share. The idea is that if D has *not* cheated, then a subset of parties $\mathcal{W}_q$ where $\mathsf{S}_q \setminus \mathcal{W}_q \in \mathcal{Z}_a$ should eventually confirm the receipt of a common share from D. Hence, the parties check for *core-sets* $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_a|}$, where each $\mathsf{S}_q \setminus \mathcal{W}_q \in \mathcal{Z}_a$, such that the parties in $\mathcal{W}_q$ have confirmed the receipt of a common share from D. Note that *irrespective* of D, each $\mathcal{W}_q$ is bound to have *at least* one non-faulty party, since $\mathcal{Z}_a$ will satisfy the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_a)$ condition.

The existence of $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_a|}$ *does not* imply that *all* non-faulty parties in S_q have received a common share, even if D is *non-faulty*, since there *might* be non-faulty parties outside $\mathcal{W}_q$. Hence, after the confirmation of the sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_a|}$, the goal is to ensure that every (non-faulty) party in $\mathsf{S}_q \setminus \mathcal{W}_q$ also gets the common share held by the (non-faulty) parties in $\mathcal{W}_q$. For this, the parties in $\mathcal{W}_q$ reveal their shares to these "outsider" parties, along with the required IC-signatures. The outsider parties then "filter" out the correctly revealed shares. The existence of at least one non-faulty party in each $\mathcal{W}_q$ guarantees that the shares filtered by the outsider parties are indeed correct.

Technical Challenges for Network Agnostic VSS and Way Out. Since, in our context, the parties will *not* be knowing the network type, our approach will be to follow the AVSS of [5], where we look for *pairwise consistency* of supposedly the common share in each group. Namely, D on having the input s, picks random shares $s_1, \ldots, s_{|\mathcal{Z}_s|}$ which sum up to s and distributes s_q to each $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$[6]. The parties in S_q then exchange IC-signed versions of their supposedly common share. To avoid an endless wait, the parties can only afford to wait till a subset of parties $\mathcal{W}_q \subseteq S_q$ have confirmed the receipt of a common share from D, where $S_q \setminus \mathcal{W}_q \in \mathcal{Z}_s$ holds. Unfortunately, $S_q \setminus \mathcal{W}_q \in \mathcal{Z}_s$ *need not* guarantee that $\mathcal{W}_q$ has at least one *non-faulty* party, since $\mathcal{Z}_s$ *need not* satisfy the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition, which is *desired* as per our semantics of linear secret-sharing with IC-signatures.

To deal with the above problem, we note that if D has distributed the shares consistently, then the subset of parties $\mathbf{S} \in \mathbb{S}_{\mathcal{Z}_s}$ which consists of *only* non-faulty parties will publicly report the pairwise consistency of their supposedly common share. Hence, we now let D search for a candidate set S_p of parties from

[6] Recall that we need $[s]$ with respect to $\mathcal{Z}_s$, *irrespective* of the network type.

$\mathbb{S}_{\mathcal{Z}_s}$ which have publicly confirmed the pairwise consistency of their supposedly common share. Once D finds such a candidate S_p, it computes and make public the core-sets $\mathcal{W}_q$ as per the following rules, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If all the parties in S_q have confirmed the pairwise consistency of their supposedly common share, then set $\mathcal{W}_q = S_q$. **(A)**
- Else if $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition *and* the parties in $(S_p \cap S_q)$ have confirmed the consistency of their supposedly common share, then set $\mathcal{W}_q = (S_p \cap S_q)$. **(B)**
- Else set $\mathcal{W}_q = S_q$ and make *public* the share s_q. **(C)**

The parties wait till they see D making public some set $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, along with sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$. Upon receiving, the parties verify and "approve" these sets as valid, provided *all* parties in S_p have confirmed the pairwise consistency of their supposedly common share and if each $\mathcal{W}_q$ is computed as per the rule **(A), (B)** or **(C)**. If $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$ are approved, then they indeed *satisfy* the requirements of core-sets as per our semantics of linear secret-sharing with IC-signatures. While this is trivially true if any $\mathcal{W}_q$ is computed either using rule **(A)** or rule **(B)**, the same holds even if $\mathcal{W}_q$ is computed using rule **(C)**. This is because, in this case, the parties publicly set $[s]_q = s_q$. Moreover, the parties take a "default" (linear) IC-signature of s_q on the behalf of S_q, where the IC-signature as well as verification points are *all* set to s_q.

If D is *non-faulty*, then *irrespective* of the network type, it will *always* find a candidate S_p and hence, compute and make public $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$ as per the above rules. This is because the set $\mathbf{S}$ *always* constitutes a candidate S_p. Surprisingly we can show that even if the core-sets are computed with respect to some *different* candidate $S_p \neq \mathbf{S}$, a *non-faulty* D will *never* make public the share corresponding to $\mathbf{S}$, since the rule **(C)** will *not* be applicable over $\mathbf{S}$, implying the privacy of s. If the network is *synchronous*, then the parties in S_p *as well as* $\mathbf{S}$ would report the pairwise consistency of their respective supposedly common share at the *same* time. This is ensured by maintaining sufficient "timeouts" in the protocol to report pairwise consistency of supposedly common shares. Consequently, rule **(A)** will be applied on $\mathbf{S}$. For an *asynchronous* network, rule **(B)** will be *applicable* for $\mathbf{S}$, as $\mathcal{Z}_s$ will satisfy the $\mathbb{Q}^{(1)}(S_p \cap \mathbf{S}, \mathcal{Z}_s)$ condition, due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition and the fact that $\mathbf{S} = \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_a$ in the *asynchronous* network.

1.2.4 Network Agnostic VSS for Multiple Dealers with Linearity

Technical Challenge in Π_{VSS} for Multiple Dealers. If *different* dealers invoke instances of Π_{VSS} to secret-share their inputs, then the linearity property of $[\cdot]$-sharing *need not* hold, since the underlying core-sets might be *different*. In more detail, let D_a and D_b be two *different* dealers which invoke instances $\Pi_{\mathsf{VSS}}^{(a)}$ and $\Pi_{\mathsf{VSS}}^{(b)}$ to generate $[a]$ and $[b]$ respectively. Let $\mathcal{W}_1^{(a)}, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}^{(a)}$ and $\mathcal{W}_1^{(b)}, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}^{(b)}$ be the underlying core-sets for $\Pi_{\mathsf{VSS}}^{(a)}$ and $\Pi_{\mathsf{VSS}}^{(b)}$ respectively. Now consider a scenario where, for some $q \in \{1, \ldots, |\mathcal{Z}_s|\}$, the core-sets $\mathcal{W}_q^{(a)}$ and $\mathcal{W}_q^{(b)}$ are *different*, even though $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q^{(a)}, \mathcal{Z}_s)$ and

$\mathbb{Q}^{(1)}(\mathcal{W}_q^{(b)}, \mathcal{Z}_s)$ conditions. Let $c \stackrel{def}{=} a + b$. Then the parties in S_q can compute $[c]_q = [a]_q + [b]_q$. As part of $[a]$, every (non-faulty) $P_i \in \mathcal{W}_q^{(a)}$ has the IC-signature $\{\mathsf{ICSig}(P_j, P_i, P_k, [a]_q)\}_{P_j \in \mathcal{W}_q^{(a)}, P_k \notin S_q}$, while as part of $[b]$, every (non-faulty) $P_e \in \mathcal{W}_q^{(b)}$ has the IC-signature $\{\mathsf{ICSig}(P_d, P_e, P_f, [b]_q)\}_{P_d \in \mathcal{W}_q^{(b)}, P_f \notin S_q}$, where the underlying IC-signatures satisfy the *linearity* property. However, since $\mathcal{W}_q^{(a)} \neq \mathcal{W}_q^{(b)}$, it is *not* guaranteed that we have a core-set $\mathcal{W}_q^{(c)}$ as part of $[c]$, where $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q^{(c)}, \mathcal{Z}_s)$ condition, such that every (non-faulty) $P_i \in \mathcal{W}_q^{(c)}$ has the IC-signature $\{\mathsf{ICSig}(P_j, P_i, P_k, [c]_q)\}_{P_j \in \mathcal{W}_q^{(c)}, P_k \notin S_q}$. If $\mathcal{W}_q^{(a)} = \mathcal{W}_q^{(b)}$, then the parties *could* set $\mathcal{W}_q^{(c)}$ to $\mathcal{W}_q^{(a)}$ and the *linearity* of IC-signatures would have ensured that every (non-faulty) $P_i \in \mathcal{W}_q^{(c)}$ *non-interactively* computes $\{\mathsf{ICSig}(P_j, P_i, P_k, [c]_q)\}_{P_j \in \mathcal{W}_q^{(c)}, P_k \notin S_q}$ from the IC-signatures, held by P_i as part of $[a]$ and $[b]$. In the *absence* of any core-set $\mathcal{W}_q^{(c)}$, *robust* reconstruction of $[c]_q$ will *fail*, which further implies *failure* of shared circuit-evaluation of ckt, where the inputs for ckt are shared by *different* parties.

Way Out. To deal with the above problem, we ensure that the core-sets are *common* for *all* the secret-shared values during the circuit-evaluation. Namely, there exist *global* core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, which constitute the core-sets for *all* the secret-shared values during the circuit-evaluation, where for each $\mathcal{GW}_q$, $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{GW}_q, \mathcal{Z}_s)$ condition. Maintaining common core-sets is challenging, especially in an *asynchronous* network and Π_{VSS} alone is *not sufficient* to achieve this goal. Rather we use a *different* approach. We generate a "bunch" of linearly secret-shared random values with IC-signatures and common core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ in *advance* through another protocol called Π_{Rand} (discussed in the next section). Later, if any party P_i needs to secret-share some x, then one of these random values is reconstructed *only* towards P_i, which uses it as a one-time pad (OTP) and makes public an OTP-encryption of x. The parties can then take the "default" secret-sharing of the OTP-encryption with IC-signatures and $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ as the core-sets and then non-interactively "remove" the pad from the OTP-encryption. This results in $[x]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ as core-sets. To ensure privacy, we need to generate L random values through Π_{Rand}, if L is the maximum number of values which need to be secret-shared by different parties during the circuit-evaluation. We show that $L \leq n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$ where c_M is the number of multiplication gates in ckt.

1.2.5 Secret-Shared Random Values with Global Core Sets

Protocol Π_{Rand} generates linearly secret-shared random values with IC-signatures and *common* core-sets. We explain the idea behind the protocol for generating one random value. The "standard" way will be to let each P_i pick a random value $r^{(i)}$ and generate $[r^{(i)}]$ by invoking an instance of Π_{VSS}. To avoid an endless wait, the parties only wait for the completion of Π_{VSS} instances invoked by a set of dealers $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$. To identify the common subset of dealers for which the corresponding Π_{VSS} instances have completed, the parties run an instance of *agreement on a common subset* (ACS) primitive [14,19]. This involves invoking n instances of our network agnostic BA, one on behalf of each dealer, to decide the Π_{VSS} instances of which dealers have completed. Let $\mathcal{C}$ be the set of common dealers identified through ACS, where $\mathcal{P} \setminus \mathcal{C} \in \mathcal{Z}_s$. The set $\mathcal{C}$ has at least one *non-faulty* party who has shared a random value. Hence, the sum of the values shared by the dealers in $\mathcal{C}$ will be random for the adversary.

Technical Challenges. The above approach *fails* in our context due to the following two "problems" in the protocol Π_{VSS}, when executed by *different* dealers.

Problem I: The first challenge is to maintain the linearity of underlying IC-signatures. To understand the issue, consider a triplet of parties P_i, P_j, P_k, acting as S, I and R respectively in various instances of Π_{VSS} invoked by *different* dealers. Recall that, to maintain the linearity of IC-signatures, it is *necessary* that P_i selects the *same* set of supporting-verifiers $\mathcal{SV}$ in *all* the instances of authentication phase involving P_j and P_k. This is possible *only if* P_i knows *all* the values on which it wants to generate the IC-signature for P_j and P_k and starts invoking *all* the instances of authentication phase. Instead, if P_i invokes instances of authentication phase *as and when* it has some data to be authenticated for P_j and P_k, then it *may not* be possible to have the *same* $\mathcal{SV}$ in *all* the instances of authentication phase, involving P_i, P_j and P_k in the above roles, especially in an *asynchronous* network. Since, in Π_{VSS}, IC-signatures are generated on the supposedly common shares (after receiving them from the underlying dealer) and multiple instances of Π_{VSS} are invoked (by *different* dealers), this means that P_i should *first* have the data from all the dealers for the various instances of Π_{VSS} and *before* invoking instances of authentication phase to generate IC-signatures on these values for P_j. This *may not* be possible, since P_i *need not* know *beforehand* which dealers it will be receiving shares from as part of Π_{VSS}.

Way Out. To deal with the above issue, we now let the dealers *publicly* commit their shares for the Π_{VSS} instances through *secure verifiable multicast* (SVM). The primitive allows a designated *sender* $\mathsf{Sen} \in \mathcal{P}$ with input v to "verifiably" send v to a designated set of *receivers* $\mathcal{R} \subseteq \mathcal{P}$, without leaking any additional information. The verifiability guarantees that even if Sen is *corrupt*, if the non-faulty parties in $\mathcal{R}$ get any value from Sen, then it will be *common* and *all* the (non-faulty) parties in $\mathcal{P}$ will "know" that Sen has sent some value to $\mathcal{R}$. Our instantiation of SVM is very simple: Sen acts as a dealer and generates $[v]$ through Π_{VSS}. Once $[v]$ is generated, the parties know that Sen has "committed" to some *unknown* value. The next step is to let *only* the parties in $\mathcal{R}$ reconstruct v.

Using SVM, we now let the various dealers distribute the shares during the underlying instances of Π_{VSS} (for Π_{Rand}) as follows. Consider the dealer P_ℓ who has invoked an instance of Π_{VSS} with input $r^{(\ell)}$. For this, it picks random shares $r_1^{(\ell)}, \ldots, r_{|\mathcal{Z}_s|}^{(\ell)}$ which sum up to $r^{(\ell)}$. Now *instead* of directly sending send $r_q^{(\ell)}$ to the parties in S_q, it invokes $|\mathcal{Z}_s|$ instances of SVM with input $r_1^{(\ell)}, \ldots, r_{|\mathcal{Z}_s|}^{(\ell)}$ and $S_1, \ldots, S_{|\mathcal{Z}_s|}$ as the designated set of receivers respectively. This serves *two* purposes. First, it guarantees that all the parties in S_q receive a *common* share from P_ℓ. Second and more importantly, *all* the parties in $\mathcal{P}$ will now *know* that P_ℓ has distributed shares to each set from $\mathbb{S}_{\mathcal{Z}_s}$. The parties then run an instance of ACS to identify a common subset of *committed dealers* $\mathcal{CD} \subseteq \mathcal{P}$, where $\mathcal{P} \setminus \mathcal{CD} \in \mathcal{Z}_s$, which have invoked the desired instances of SVM and delivered the required shares to each group $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$. The way timeouts are maintained as part of the ACS, it will be ensured that in a *synchronous* network, *all non-faulty* dealers are present in $\mathcal{CD}$. Once the set $\mathcal{CD}$ is identified, it is guaranteed that every *non-faulty* party P_i will have the shares from *all* the dealers in $\mathcal{CD}$. And once it has the shares from all the dealers in $\mathcal{CD}$, it starts generating the IC-signatures on these shares for the designated parties as part of the Π_{VSS} instances corresponding to the dealers in $\mathcal{CD}$ and ensures that all the pre-requisites are satisfied to guarantee the linearity of the underlying IC-signatures. Now *instead* of selecting the set of dealers $\mathcal{C}$ (for Π_{Rand}) from $\mathcal{P}$, the parties run an instance of ACS over the set of committed dealers $\mathcal{CD}$ to select $\mathcal{C}$ where $\mathcal{CD} \setminus \mathcal{C} \in \mathcal{Z}_s$ holds. We stress that *irrespective* of the network type, the set $\mathcal{C}$ is *still* guaranteed to have at least one non-faulty party. While this is trivially true in an *asynchronous* network where $\mathcal{Z}_a$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{C}, \mathcal{Z}_a)$ condition, the same is true in the *synchronous* network because $\mathcal{CD}$ will have *all* non-faulty dealers.

Problem II: The second problem (in the proposed Π_{Rand}) is that the underlying core-sets might be *different* for the values shared by the dealers in $\mathcal{CD}$ (and hence $\mathcal{C}$). Instead, we require every dealer in $\mathcal{CD}$ to secret-share random values with *common* underlying core-sets. Only then will it be ensured that the random values generated through Π_{Rand} are secret-shared with common core-sets.

Way Out. Getting rid of the above problem is *not* possible if we let every dealer in $\mathcal{CD}$ compute *individual* core-sets during their respective instances of Π_{VSS}, as per the steps of Π_{VSS}. Recall that in Π_{VSS}, the dealer D computes the underlying core-sets with respect to the "first" set of parties S_p from $\mathbb{S}_{|\mathcal{Z}_s|}$ which confirm the pairwise consistency of their supposedly common share after exchanging IC-signatures on these values. As a result, different dealers (in Π_{Rand}) may end up computing *different* core-sets in their instances of Π_{VSS} with respect to *different* candidate S_p sets. To deal with this issue, we *instead* let each dealer in $\mathcal{CD}$ *continue* computing and publishing *different* "legitimate" core-sets with respect to various "eligible" candidate S_p sets from $\mathbb{S}_{\mathcal{Z}_s}$. The parties run an instance of ACS to identify a *common* subset of dealers $\mathcal{C} \in \mathcal{CD}$ where $\mathcal{CD} \setminus \mathcal{C} \in \mathcal{Z}_s$, such that *all* the dealers have computed and published "valid" core-sets, computed with the respect to the *same* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$. The idea here is that there always exists a set $\mathbf{S} \in \mathbb{S}_{\mathcal{Z}_s}$ consisting of *only* non-faulty parties. So if the set of *non-faulty* dealers $\mathcal{H}$ in $\mathcal{CD}$ keep computing and publishing *all possible* candidate core-sets in their Π_{VSS} instances, then they will publish core-sets with respect to $\mathbf{S}$. Hence, $\mathcal{H}$ and $\mathbf{S}$ *always* constitute the candidate $\mathcal{CD}$ and the common S_p set.

Note that identifying $\mathcal{C}$ out of $\mathcal{CD}$ through ACS satisfying the above requirements is *non-trivial* and requires carefully executing the underlying instances of BA in "two-dimensions". We first run $|\mathcal{Z}_s|$ instances of Π_{BA}, one on the behalf of each set in $\mathbb{S}_{\mathcal{Z}_s}$, where the q^{th} instance is executed to decide whether a subset of dealers in $\mathcal{CD} \setminus Z$ for some $Z \in \mathcal{Z}_s$ have published valid core-sets with respect to the set $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. This enables the parties to identify a *common* set $S_{q_{\mathsf{core}}} \in \mathbb{S}_{\mathcal{Z}_s}$, such that it is guaranteed that a subset of dealers in $\mathcal{CD} \setminus Z$ for some $Z \in \mathcal{Z}_s$ have indeed published valid core-sets with respect to the set $S_{q_{\mathsf{core}}}$. Once the set $S_{q_{\mathsf{core}}}$ is identified, the parties then run $|\mathcal{CD}|$ instances of BA to decide which dealers in $\mathcal{CD}$ have published core-sets with respect to $S_{q_{\mathsf{core}}}$.

1.2.6 Network Agnostic Secure Multiplication

To generate secret-shared random multiplication-triples for evaluating the multiplication gates in ckt (using Beaver's trick), we need a network agnostic secure multiplication protocol which securely generates a secret-sharing of the product of two secret-shared values. The key subprotocol behind our multiplication protocol is a *non-robust* multiplication protocol $\Pi_{\mathsf{BasicMult}}$ (standing for basic multiplication), which takes inputs $[a]$ and $[b]$ and an *existing* set of *globally discarded* parties $\mathcal{GD}$, which contains only *corrupt* parties. The protocol securely generates $[c]$ *without* revealing any additional information about a, b (and c). If *no* party in $\mathcal{P} \setminus \mathcal{GD}$ cheats, then $c = a \cdot b$ holds. The idea behind the protocol is to let each *summand* $[a]_p \cdot [b]_q$ be secret-shared by a *summand-sharing party*. Then $[a \cdot b]$ can be computed from the secret-sharing of each summand, owing to the linearity property. Existing multiplication protocols in the *synchronous* and *asynchronous* setting [40,5] also use an instantiation of $\Pi_{\mathsf{BasicMult}}$, based on the above idea. In the sequel, we recall them, followed by the technical challenges faced in the network agnostic setting and how we deal with them.

$\Pi_{\mathsf{BasicMult}}$ in the Synchronous Setting with $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ Condition [40]. In [40], each summand $[a]_p \cdot [b]_q$ is *statically* assigned to a *designated* summand-sharing party through some *deterministic* assignment, which is possible since $[a]_p$ and $[b]_q$ are held by the parties in $(S_p \cap S_q)$. This is *non-empty*, since the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition holds. Since the parties in $\mathcal{GD}$ are *already* known to be corrupted, all the shares $[a]_p, [b]_p$ held by the parties in $\mathcal{GD}$ are *publicly reconstructed* and *instead* of letting the parties in $\mathcal{GD}$ secret-share their assigned summands, the parties take the "default" secret-sharing of these summands.

$\Pi_{\mathsf{BasicMult}}$ in the Asynchronous Setting with $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ Condition [5]. The idea of *statically* designating each summand $[a]_p \cdot [b]_q$ to a *unique* party in $\mathcal{P} \setminus \mathcal{GD}$ *need not* work in the *asynchronous* setting, since the designated party may be *corrupt* and *need not* secret-share any summand, thus resulting in an *endless* wait. To deal with this challenge, [5] *dynamically* selects the summand-sharing parties for each summand. In more detail, let $\mathcal{Z}_a = \{\mathsf{Z}_1, \ldots, \mathsf{Z}_{|\mathcal{Z}_a|}\}$ and $\mathbb{S}_{\mathcal{Z}_a} = \{\mathsf{S}_1, \ldots, \mathsf{S}_{|\mathcal{Z}_a|}\}$, where each $\mathsf{S}_r = \mathcal{P} \setminus \mathsf{Z}_r$. Since the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition is satisfied and $\mathcal{GD} \in \mathcal{Z}_a$, it follows that $(\mathsf{S}_p \cap \mathsf{S}_q) \setminus \mathcal{GD} \neq \emptyset$. This implies that there exists at least one non-faulty party in $(\mathsf{S}_p \cap \mathsf{S}_q)$ who can secret-share the summand $[a]_p \cdot [b]_q$. Hence, *every* party in $\mathcal{P} \setminus \mathcal{GD}$ is allowed to secret-share *all* the summands it is "capable" of, with special care taken to ensure that each summand $[a]_p \cdot [b]_q$ is considered *exactly once*. For this, the protocol now proceeds in "hops", where in each hop all the parties in $\mathcal{P} \setminus \mathcal{GD}$ secret-share all the summands they are capable of, *but* a *single* summand sharing party is finally selected for the hop through ACS. Then, all the summands which have been shared by the elected summand-sharing party are "marked" as shared and not considered for sharing in the future hops. Moreover, a party who has already served as a summand-sharing party is *not* selected in the future hops.

Technical Challenges in the Network Agnostic Setting. The *asynchronous* $\Pi_{\mathsf{BasicMult}}$ based on *dynamically* selecting summand-sharing parties will *fail* in the *synchronous* network, since the $\mathbb{Q}^{(3)}$ condition *need not* be satisfied. On the other hand, *synchronous* $\Pi_{\mathsf{BasicMult}}$ based on *statically* selecting summand-sharing parties will *fail* if a designated summand-sharing party *does not* secret-share the required summands, resulting in an endless wait. The way out is to select summand-sharing parties in *three* phases. We *first* select summand-sharing parties dynamically in *hops*, following the approach of [5], till we find a subset of parties from $\mathbb{S}_{\mathcal{Z}_s}$ which have shared *all* the summands they are capable of. Then in the *second* phase, the *remaining* summands which are *not* yet secret-shared are *statically* assigned and shared by the respective designated summand-sharing parties. To avoid an endless wait in this phase, the parties wait only for a "fixed" time required for the parties to secret-share the assigned summands (corresponding to the time taken in a *synchronous* network) and run instances of BA to identify which of the designated summand-sharing parties have shared their summands up during the second phase. During the *third* phase, any "leftover" summand which is *not yet* shared is *publicly* reconstructed by reconstructing the corresponding shares and a default sharing is taken for such summands.

The idea here is the following: *all* non-faulty parties will share the summands which are assigned to them, either statically or dynamically, *irrespective* of the network type. Consequently, the first phase will be always over, since the set consisting of *only* non-faulty parties always constitutes a candidate set of summand-sharing parties which the parties look for to complete of the first phase. Once the first phase is over, the second phase is bound to be over since the parties wait *only* for a fixed time. The third phase is always bound to be over, once the first two phases are over, since it involves publicly reconstructing the leftover summands. The way summands are assigned across the three phases, it will be always guaranteed that every summand is considered for sharing once in exactly one of the three phases and no summand will be left out. The crucial point here is that the the shares held *only* by the non-faulty parties *never* get publicly reconstructed, thus guaranteeing that the adversary *does not* learn any additional information about a and b. This is obviously true in a *synchronous* network because we always have the *second* phase where every non-faulty party who is *not* selected as a summand-sharing party during the first phase will get the opportunity to secret-share its assigned summands. On the other hand, in an *asynchronous* network, it can be shown that *all* the summands which involve any share held by the non-faulty parties would have been secret-shared during the *first* phase itself. In more detail, let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Moreover, let $S_h \in \mathbb{S}_{\mathcal{Z}_s}$ be the group consisting of *only* non-faulty parties which hold the shares $[a]_h$ and $[b]_h$. Consider an *arbitrary* summand $[a]_h \cdot [b]_q$. Suppose the first phase gets over because every party in $S_\ell \in \mathbb{S}_{\mathcal{Z}_s}$ has been selected as a summand-sharing party during the *first* phase. Then consider the set $(S_\ell \cap \mathcal{H} \cap S_q)$, which is *not* empty due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Hence, there exists *some* $P_j \in (\mathcal{H} \cap S_\ell \cap S_q)$, who would have shared $[a]_h \cdot [b]_q$ when selected as a summand-sharing party during some hop in the first phase. Due to a similar reason, any summand of the form $[a]_q \cdot [b]_h$ would have been secret-shared during the first phase itself.

1.3 Other Related Works

The domain of network agnostic cryptographic protocols is relatively new and almost all the existing works have considered *threshold* adversaries. The work of [16] presents a network agnostic *cryptographically-secure* atomic broadcast protocol. The work of [45] studies Byzantine fault tolerance and state machine replication protocols for multiple thresholds, including t_s and t_a. The work of [35] presents a network agnostic protocol for the task of approximate agreement using the condition $2t_s + t_a < n$. The same condition has been used to design a network agnostic distributed key-generation (DKG) protocol in [6]. A recent work [28] has studied the problem of network agnostic perfectly-secure message transmission (PSMT) [29] over *incomplete* graphs.

1.4 Open Problems

There are several interesting directions to explore for network agnostic MPC protocols. Here we mention few of them. It is *not* known whether the condition $3t_s + t_a < n$ (resp. $\mathbb{Q}^{(3,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$) is *necessary* for the network agnostic MPC with *perfect* security against threshold (resp. non-threshold) adversary. An abundant amount of research effort has been spent to improve both the theoretical as well as practical efficiency of (unconditionally-secure) SMPC and AMPC protocols. The works of [3,4] and this work just focus on the *possibility* of unconditionally-secure network agnostic MPC. Upgrading the efficiency of these protocols to those of state of the art SMPC and AMPC protocols seems to require a significant research effort. Even though the complexity of our MPC protocol is polynomial in n and $|\mathcal{Z}_s|$, when instantiated for *threshold* adversaries (where $|\mathcal{Z}_s|$ has all subsets of $\mathcal{P}$ of size up to t_s), it may require an *exponential* (in n) amount of computation and communication. This is *unlike* the case for *perfect security*, where we have a network agnostic MPC protocol against threshold adversaries with a complexity polynomial (in n) [3]. Hence, designing network agnostic MPC protocol against *threshold* adversaries with *statistical* security and *polynomial* complexity is left as a challenging open problem.

2 Preliminaries and Definitions

We assume the *pair-wise secure channel* model, where 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 network type. In a *synchronous* network, every message sent is delivered within a *known* time Δ. In an *asynchronous* network, messages can be delayed arbitrarily, but finitely, with every message sent being delivered *eventually*. The distrust among $\mathcal{P}$ is modelled by a *malicious* (Byzantine) adversary Adv, 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 parties *not* under the control of Adv are called *honest*. We assume the adversary to be *static*, who decides the set of corrupt parties at the beginning of the protocol execution. As our main goal is to show the possibility of statistically-secure network agnostic MPC, we keep the formalities to a bare minimum and prove the security of our protocols using the *property-based* definition, by listing the security properties achieved by our protocols. However, our protocols can also be proven to be secure using the more rigorous *Universal Composability* (UC) definitional framework [20], without affecting their efficiency.

Adversary Adv can corrupt any one subset of parties from $\mathcal{Z}_s$ and $\mathcal{Z}_a$ in *synchronous* and *asynchronous* network respectively. The adversary structures are *monotone*, implying that if $Z \in \mathcal{Z}_s$ ($Z \in \mathcal{Z}_a$ resp.), then every subset of Z also belongs to $\mathcal{Z}_s$ (resp. $\mathcal{Z}_a$). We assume that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the conditions $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ respectively, which are *necessary* for statistically-secure MPC in the synchronous and asynchronous network respectively. Additionally, we assume that $\mathcal{Z}_a \subset \mathcal{Z}_s$. Moreover, $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.

In our protocols, all computations are done over a finite field $\mathbb{F}$, where $|\mathbb{F}| > n^5 \cdot 2^{\mathsf{ssec}}$ and ssec is the underlying statistical security parameter. Looking ahead, this will ensure that the error probability in our MPC protocol is upper bounded by $2^{-\mathsf{ssec}}$. Without loss of generality, we assume that each P_i has an input $x_i \in \mathbb{F}$, and the parties want to securely compute a function $f : \mathbb{F}^n \rightarrow \mathbb{F}$, represented by an arithmetic circuit ckt over $\mathbb{F}$, consisting of linear and non-linear (multiplication) gates, where ckt has c_M multiplication gates and a multiplicative depth of D_M.

We assume the existence of an *unconditionally-secure public-key infrastructure* (PKI), for an unconditionally-secure signature scheme, also called *pseudo-signature* [50,34]. We briefly explain the requirements from such a setup and refer to [34] for complete formal details. There exists a *publicly known* vector of public keys $(pk_1, \ldots, pk_n)$, where each *honest* P_i holds the generated secret key sk_i, associated with pk_i.[7] A *valid* signature τ on message m from P_i is one for which $\mathsf{Verify}_{pk_i}(m, \tau) = 1$, where Verify is the *verification function* of the underlying signature scheme. For simplicity, we make the standard convention of treating signatures as *idealized* objects during our protocol analysis; i.e., we assume that

[7] Corrupt parties may choose their keys arbitrarily.

the signatures are *perfectly unforgeable* and hence Adv will fail to forge signature of an honest party on any message, which is not signed by the party. We also assume that the signatures are *transferable* and any party upon receiving a valid signature from a party can send and get it verified by any other party. However, *unlike* the standard digital signatures which are *computationally-secure* and offer arbitrary number of transfers, pseudo-signatures offer a "limited" number of transfers, which is typically bounded as a function of the number of parties in the protocol where pseudo-signature is used as a primitive. We assume that the given setup supports the required number of transfers demanded by our protocols. We use $|\sigma|$ to represent the size of a pseudo-signature in bits. If P_i signs a message m, then we denote the resultant signed message as $\langle m \rangle_i$.

Termination Guarantees of Our Sub-Protocols: As done in [3,5], 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* computing 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 [23] which ensures that the honest parties (eventually) obtain their respective output *almost-surely* with probability 1. This means that the probability that an honest party obtains its output after participating for *infinitely* many rounds approaches 1 *asymptotically* [1,47,7]. That is:

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

where the probability is over the random coins of the honest parties and the adversary in the protocol. The property of *almost-surely* obtaining the output carries over to the "higher" level protocols, where ABA is used as a building block. We will say that the "*honest parties obtain some output almost-surely from protocol Π*" to mean that every *honest* P_i *asymptotically* obtains its output in Π with probability 1, in the above sense.

3 Network Agnostic Unconditionally Secure Byzantine Agreement

We recall the definition of BA from [4], which is adapted from [15,3].

Definition 3.1 (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 $\{0,1,\perp\}$. Moreover, let* Adv *be a computationally-unbounded adversary, characterized by adversary structure $\mathcal{Z}$, where* Adv *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 $\perp$.*
– $\mathcal{Z}$**-Consistency**: *Π has $\mathcal{Z}$-consistency if all honest parties with an output, output the same value (which can be $\perp$).*
– $\mathcal{Z}$**-Weak Consistency**: *Π has $\mathcal{Z}$-weak consistency if all honest parties with an output, output either a common $v \in \{0, 1\}$ or $\perp$.*

Π is called a $\mathcal{Z}$-secure synchronous BA (SBA) if, in a synchronous network, it achieves $\mathcal{Z}$-guaranteed liveness, $\mathcal{Z}$-validity, and $\mathcal{Z}$-consistency. Π is called a $\mathcal{Z}$-secure asynchronous BA (ABA) if, in an asynchronous network it has $\mathcal{Z}$-almost-surely liveness, $\mathcal{Z}$-validity and $\mathcal{Z}$-consistency.[8]

To design our network agnostic BA protocol, we will be using a special type of broadcast protocol. We next review the definition of broadcast from [4] which is further adapted from [15,3].

Definition 3.2 (Broadcast). *Let Π be a protocol, where a designated sender* $\mathsf{Sen} \in \mathcal{P}$ *has input $m \in \{0, 1\}^\ell$, and parties obtain a possible output, including $\perp$. Moreover, let* Adv *be a computationally-unbounded adversary, characterized by an adversary structure $\mathcal{Z}$, where* Adv *can corrupt any subset from $\mathcal{Z}$ during 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* Sen *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* Sen *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* Sen *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* Sen *is corrupt, then every honest party with an output, outputs a common $m^\star \in \{0, 1\}^\ell$ or $\perp$.*

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

We next recall a blueprint for network agnostic BA from [15,3].

3.1 A Blueprint for Network Agnostic BA [15,3]

To design our network agnostic BA, we *assume* the existence of the following sub-protocols.

– *Synchronous BA with Asynchronous Guarantees*: we assume the existence of a protocol Π_{SBA}, which is a $\mathcal{Z}_s$-secure SBA and which has $\mathcal{Z}_a$-weak validity and $\mathcal{Z}_a$-guaranteed liveness in the *asynchronous* network. At (local) time

[8] The seminal FLP impossibility result [32] rules out the possibility of *any* deterministic ABA, where there always exists a "bad" execution in which the honest parties may keep on running the protocol forever, without obtaining any output. To circumvent this, one can opt for randomized ABA protocols and hope that the bad executions occur asmptotically with probability 0.

T_{SBA}, all honest parties will have an output (which could be $\perp$), *irrespective* of the network type.

– *Asynchronous BA with Synchronous Guarantees*: we assume the existence of a protocol Π_{ABA}, which is a $\mathcal{Z}_a$-secure ABA. Moreover, in a *synchronous* network, the protocol has $\mathcal{Z}_s$-validity, with all honest parties computing their output within time $T_{\mathsf{ABA}} = c \cdot \Delta$ in this case, for some *known constant* c.[9]

Based on Π_{SBA} and Π_{ABA}, one can design a network agnostic BA protocol Π_{BA} as follows, following the blueprint of [15,3]. The parties first invoke an instance of Π_{SBA}, *assuming* a *synchronous* network. If the network is indeed synchronous, then the (honest) parties should have a *binary* output at time T_{SBA}. The parties check the same and either switch their input to the output of Π_{SBA}, if it is *not* $\perp$, or stick to their original input. The parties then invoke an instance of Π_{ABA} with "updated" inputs and the output of Π_{ABA} is set to be the overall output. The description of Π_{BA}, taken from [3], is presented in Fig 1.

Protocol Π_{BA}

– On having input b_i, participate in an instance of Π_{SBA} with input b_i and wait till the local time becomes T_{SBA}. Let v_i be the output from Π_{SBA} at time T_{SBA}. If $v_i \neq \perp$, then set $v_i^\star = v_i$. Else set $v_i^\star = b_i$.
– Participate in an instance of Π_{ABA} with input $v_i^\star$. Output the result of Π_{ABA}, when it is available.

Fig. 1: Network agnostic BA protocol from Π_{SBA} and Π_{ABA}. The above code is executed by P_i.

Theorem 3.1, follows from [3], given that Π_{SBA} and Π_{ABA} achieve the stated properties. For completeness, the theorem is proved in Appendix A.

Theorem 3.1. *Let $\mathcal{Z}_a \subset \mathcal{Z}_s$ such that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the conditions $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ respectively. Moreover, let $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Then protocol Π_{BA} achieves the following.*[10]

– **Synchronous Network**: *the protocol is a $\mathcal{Z}_s$-secure SBA, where all honest parties get their output at time $T_{\mathsf{BA}} = T_{\mathsf{SBA}} + T_{\mathsf{ABA}}$.*
– **Asynchronous Network**: *the protocol is a $\mathcal{Z}_a$-secure ABA.*

We now proceed to instantiate protocols Π_{SBA} and Π_{ABA}.

3.2 Π_{SBA}: Synchronous BA with Asynchronous Weak Validity and Guaranteed Liveness

To design protocol Π_{SBA}, we again follow the blueprint of [3], which design Π_{SBA} based on *three* components.

[9] Thus Π_{ABA} has $\mathcal{Z}_s$-guaranteed liveness in the *synchronous* network, *if* all honest parties have the *same* input. However, if the honest parties have *different* inputs, then Π_{ABA} *need not* provide guaranteed liveness or consistency guarantees in the *synchronous* network.

[10] As the number of invocations of Π_{BA} in our MPC protocol will be *independent* of $|\mathsf{ckt}|$, we *do not* focus on its *exact* complexity. However, we confirm that it will be polynomial in n and $|\mathcal{Z}_s|$.

3.2.1 SBA with Asynchronous Guaranteed Liveness

The *first* component for designing Π_{SBA} is an SBA protocol, which has *just* guaranteed liveness in an *asynchronous* network. In [50], the authors have presented an SBA protocol against *threshold* adversaries, tolerating up to $t_s < n/2$ faults. The protocol which we denote as Π_{PW}, modifies the Dolev-Strong BA protocol [30], by *replacing* digital signatures with pseudo-signatures. We note that Π_{PW} can be easily generalized, if $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition. To achieve guaranteed liveness in an *asynchronous* network, the parties run the protocol till the supposed timeout in the synchronous network and check if any output is computed and in case no output is computed, $\bot$ is taken as the output. Since the protocol is an easy generalization of the existing protocol against threshold adversaries, protocol Π_{PW} and proof of Lemma 3.1 are available in Appendix A.

Lemma 3.1. *Protocol Π_{PW} achieves the following, where t is the cardinality of the maximum-sized subset in $\mathcal{Z}_s$.*

- **Synchronous Network**: *The protocol is a $\mathcal{Z}_s$-secure SBA protocol, where all honest parties compute their output at time $T_{\mathsf{PW}} \stackrel{def}{=} (t+1) \cdot \Delta$.*
- **Asynchronous Network**: *The protocol achieves $\mathcal{Z}_a$-guaranteed liveness, where all honest parties compute their output at time T_{PW}.*
- **Communication Complexity**: *$\mathcal{O}(n^4 \cdot \ell \cdot |\sigma|)$ bits are sent by the honest parties, if the inputs of the parties are of size ℓ bits.*

3.2.2 Asynchronous Broadcast with Synchronous Guarantees

The *second* component for designing Π_{SBA} is an *asynchronous* broadcast protocol Π_{Acast} (also called Acast), which provides *liveness, validity* and a "variant" of consistency in a *synchronous* network. The variant guarantees that if Sen is *corrupt* and the honest parties compute an output in a *synchronous* network, then they *may not* do it at the *same* time and there might be a *gap* in the time at which the honest parties compute an output. In [43,4], an instantiation of Π_{Acast} is provided against $\mathbb{Q}^{(3)}$ adversary structures.[11] Unfortunately, the protocol *fails* to provide *any* security guarantees in a *synchronous* network against $\mathbb{Q}^{(2)}$ adversary structures. So we provide a *different* instantiation of Acast for our setting (see Fig 2). The protocol is obtained by generalizing the ideas used in the broadcast protocol of [46]. The protocol of [46] uses a computational PKI, which we replace with an unconditional PKI. The protocol consists of three phases and each (honest) party executes a phase *at most* once.

Protocol $\Pi_{\mathsf{Acast}}(\mathsf{Sen}, m, \mathcal{Z}_s, \mathcal{Z}_a)$

Each $P_i \in \mathcal{P}$ executes each of the following phases *at most* once.

- **(Propose):** If $P_i = \mathsf{Sen}$, then on having the input m, send $\langle(\mathsf{propose}, m)\rangle_{\mathsf{Sen}}$ to all the parties.

[11] The protocol is a generalization of the classic Bracha's *threshold* Acast protocol [18], tolerating $t < n/3$ corruptions.

- **(Vote):** Upon receiving the *first* propose message $\langle(\mathsf{propose}, m)\rangle_{\mathsf{Sen}}$ from Sen with valid signature, send $\langle(\mathsf{propose}, m)\rangle_{\mathsf{Sen}}$ to all the parties and wait till the local time increases by Δ. If $\langle(\mathsf{propose}, m')\rangle_{\mathsf{Sen}}$ is not received from any party where $m' \neq m$, then send $\langle(\mathsf{vote}, m)\rangle_i$ to all the parties.
- **(Output):** Upon receiving a vote message $\langle(\mathsf{vote}, m)\rangle_j$ with valid signature corresponding to every $P_j \in \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, do the following:
  - Let $\mathcal{C}(m)$ denote the collection of signed $\langle(\mathsf{vote}, m)\rangle_j$ messages.
  - Send $\mathcal{C}(m)$ to all the parties and output m.

Fig. 2: Asynchronous broadcast with synchronous guarantees. The above code is executed by each $P_i \in \mathcal{P}$ including the designated sender Sen

The proof of Lemma 3.2 is available in Appendix A.

Lemma 3.2. *Protocol Π_{Acast} achieves the following properties.*
- *Asynchronous Network: The protocol is a $\mathcal{Z}_a$-secure broadcast protocol.*
- *Synchronous Network:* **(a)** **$\mathcal{Z}_s$-Liveness***: If Sen is honest, then all honest parties obtain an output within time 3Δ.* **(b)** **$\mathcal{Z}_s$-Validity***: If Sen is honest, then every honest party with an output, outputs m.* **(c)** **$\mathcal{Z}_s$-Consistency***: If Sen 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 $T + \Delta$.*
- *Communication Complexity: $\mathcal{O}(n^3 \cdot \ell \cdot |\sigma|)$ bits are communicated by the honest parties, where ℓ is the size of Sen's input.*

Terminologies for Using Π_{Acast}. In the protocol Π_{Acast}, any party from $\mathcal{P}$ can be designated as Sen. In the rest of the paper we will say that "P_i *Acasts* m" to mean that P_i acts as Sen and invokes an instance of Π_{Acast} with input m, and the parties participate in this instance. Similarly, "P_j *receives* m *from the Acast of* P_i" means that P_j outputs m in the corresponding instance of Π_{Acast}.

3.2.3 Synchronous Broadcast with Asynchronous Guarantees

The *third* component for designing Π_{SBA} is a broadcast protocol Π_{BC}, which is secure in a *synchronous* network and which also provides liveness, weak validity and weak consistency in an *asynchronous* network. Note that the guarantees of Π_{BC} are *different* from that of Π_{Acast}. The design of Π_{BC} is based on the idea from [4], by carefully combining protocols Π_{Acast} and Π_{PW}. In the protocol, Sen 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_{PW}, with whatever output has been obtained from the Π_{Acast} instance at time 3Δ; in case no output is obtained, then the input is $\perp$. Finally, at time $3\Delta + T_{\mathsf{PW}}$, the parties output an $m^\star$, if it is the output of the Π_{Acast} instance *as well as* the output of Π_{PW}, else the output of the parties will be $\perp$. We recall the description of Π_{BC} from [4] and present it in Fig 3.

Protocol Π_{BC} as a Network Agnostic Secure Broadcast. Protocol Π_{BC} *only* guarantees *weak validity* and *weak consistency* in an *asynchronous* network, since only a subset of *honest* parties may receive Sen's message from the Acast of Sen within time $3\Delta + T_{\mathsf{PW}}$. Note that maintaining the time-out is *essential*, as we need *liveness* from Π_{BC} (*irrespective* of the network type) when used later in protocol Π_{SBA}. Looking ahead, we will use Π_{BC} in our VSS protocol for broadcasting values. The weak validity and consistency 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 $3\Delta + T_{\mathsf{PW}}$, 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}. Following [3,4], we make a provision for this in Π_{BC}. Namely, each P_i who outputs $\perp$ at time $3\Delta + T_{\mathsf{PW}}$ "switches" its output to $m^\star$, if P_i *eventually* receives $m^\star$ from Sen's Acast. We stress that this switching is *only* for the parties who obtained $\perp$ at time $3\Delta + T_{\mathsf{PW}}$. To differentiate between the two ways of obtaining output, we use the terms *regular-mode* and *fallback-mode*. Regular-mode is the process of deciding the output at time $3\Delta + T_{\mathsf{PW}}$, while fallback-mode is the process of deciding the output beyond time $3\Delta + T_{\mathsf{PW}}$.

Protocol $\Pi_{\mathsf{BC}}(\mathsf{Sen}, m, \mathcal{Z}_s, \mathcal{Z}_a)$

(Regular Mode)

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

(Fallback Mode)

– Each $P_i \in \mathcal{P}$ who outputs $\perp$ at time $3\Delta + T_{\mathsf{PW}}$, changes it to $m^\star$, if $m^\star$ is received from Acast of Sen.

Fig. 3: Synchronous broadcast with asynchronous guarantees.

Theorem 3.2 follows from [4] and is proved in Appendix A.

Theorem 3.2. *Protocol Π_{BC} achieves the following, with a communication complexity of $\mathcal{O}(n^4 \cdot \ell \cdot |\sigma|)$ bits, where $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{PW}}$.*

– *Synchronous network:* **(a)** $\mathcal{Z}_s$**-Liveness***: At time T_{BC}, each honest party has an output.* **(b)** $\mathcal{Z}_s$**-Validity***: If Sen is honest, then at time T_{BC}, each honest party outputs m.* **(c)** $\mathcal{Z}_s$**-Consistency***: If Sen is corrupt, then the output of every honest party is the same at time T_{BC}.* **(d)** $\mathcal{Z}_s$**-Fallback Consistency***: If Sen is corrupt, and some honest party outputs $m^\star \neq \perp$ at time T through fallback-mode, then every honest party outputs $m^\star$ by time $T + \Delta$.*
– *Asynchronous Network:* **(a)** $\mathcal{Z}_a$**-Liveness***: At time T_{BC}, each honest party has an output.* **(b)** $\mathcal{Z}_a$**-Weak Validity***: If Sen is honest, then at time T_{BC}, each honest party outputs m or $\perp$.* **(c)** $\mathcal{Z}_a$**-Fallback Validity***: If Sen is honest, then each honest party with output $\perp$ at time T_{BC}, eventually outputs m through fallback-mode.* **(d)** $\mathcal{Z}_a$**-Weak Consistency***: If Sen is corrupt, then there exists an $m^\star \neq \perp$, such that at time T_{BC}, each honest party outputs $m^\star$ or $\perp$.* **(e)** $\mathcal{Z}_a$**-Fallback Consistency***: If Sen is corrupt, and*

some honest party outputs $m^\star \neq \bot$ at time $T \geq T_{\mathsf{BC}}$, 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 Sen 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}.

3.2.4 $\Pi_{\mathsf{BC}} \rightarrow \Pi_{\mathsf{SBA}}$

Finally, using Π_{BC} we instantiate protocol Π_{SBA}, following the blueprint of [4]. In the protocol, every party broadcasts its input bit (for Π_{SBA}) through an instance of Π_{BC}. At time T_{BC}, the parties check if "sufficiently many" instances of Π_{BC} have produced a binary output (which should have happened in a *synchronous* network) and if so, they output the "majority" of those values. Otherwise, the network is *asynchronous*, in which case the parties output $\bot$. The description of Π_{SBA} is recalled from [4] and presented in Fig 4.

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

- 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 $\mathcal{SV}$ if $b_i^{(j)} \neq \bot$.
  - If $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, then compute the output as follows.
    - If there exists a subset of parties $\mathcal{SV}_i \subseteq \mathcal{SV}$, such that $\mathcal{SV} \setminus \mathcal{SV}_i \in \mathcal{Z}_s$ and $b_i^{(j)} = b$ for all the parties $P_j \in \mathcal{SV}_i$, then output b.[a]
    - Else output 1.
  - Else output $\bot$.

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

Fig. 4: Synchronous BA with asynchronous guaranteed liveness and weak validity. The above code is executed by every $P_i \in \mathcal{P}$.

Theorem 3.3 follows from [4] and is proved in Appendix A.

Theorem 3.3. *Protocol Π_{SBA} achieves the following where $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$, incurring a communication of $\mathcal{O}(n^5 \cdot |\sigma|)$ bits.*
- *Synchronous Network: the protocol is a $\mathcal{Z}_s$-secure SBA protocol where honest parties have an output, different from $\bot$, at time T_{SBA}.*
- *Asynchronous Network: the protocol achieves $\mathcal{Z}_a$-guaranteed liveness and $\mathcal{Z}_a$-weak validity, such that all honest parties have an output at (local) time T_{SBA}.*

3.3 Π_{ABA}: Asynchronous BA with Synchronous Validity

To the best of our knowledge, the *only* known *unconditionally-secure* ABA protocol is due to [23], which generalizes the framework of randomized ABA

[51,11,21,31] against general adversaries. The protocol of [23] uses a *graded agreement* (GA) protocol (also known as the vote protocol), along with a secure coin-flip protocol. Unfortunately, both these primitives provide security (in an *asynchronous* network) against $\mathbb{Q}^{(3)}$ adversary structures and *fail* to provide any security guarantees against $\mathbb{Q}^{(2)}$ adversary structures in a *synchronous* network. Consequently, the ABA protocol of [23] *fails* to provide any security guarantees in a *synchronous* network against $\mathbb{Q}^{(2)}$ adversary structures.[12] We give a different instantiation of Π_{ABA} by generalizing a few ideas used in [15]. Our instantiation of Π_{ABA} is based on following *two* components.

Component I: Asynchronous Graded Agreement with Synchronous Validity. We assume the existence of a GA protocol Π_{GA}, where each party has a binary input. The output for each party is a value from $\{0, 1, \bot\}$, along with a grade from $\{0, 1, 2\}$. The protocol achieves the following properties.

- *Asynchronous Network* — The following properties are achieved, even if the adversary corrupts any subset from $\mathcal{Z}_a$: **(a): $\mathcal{Z}_a$-Liveness:** If all honest parties participate in the protocol, then each honest party eventually obtains an output. **(b) $\mathcal{Z}_a$-Graded Validity:** If every honest party's input is b, then all honest parties with an output, output $(b, 2)$. **(c) $\mathcal{Z}_a$-Graded Consistency:** If two honest parties output grades g, g', then $|g - g'| \leq 1$ holds; moreover, if two honest parties output (v, g) and (v', g') with $g, g' \geq 1$, then $v = v'$.
- *Synchronous Network* — The following properties are achieved, even if the adversary corrupts any subset from $\mathcal{Z}_s$: **(a): $\mathcal{Z}_s$-Liveness:** If all honest parties participate in the protocol with the *same* input, then after some fixed time T_{GA}, all honest parties obtain an output. **(b) $\mathcal{Z}_s$-Graded Validity:** If every honest party's input is b, then all honest parties with an output, output $(b, 2)$.

Note that we *do not* require any form of consistency from Π_{GA} in the *synchronous* network. We give an instantiation of Π_{GA} with the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, by generalizing the *threshold* GA protocol of [15] with condition $2t_s + t_a < n$, such that $T_{\mathsf{GA}} = 4 \cdot \Delta$. The protocol of [15] uses digital signatures (hence, is *computationally secure*), which we replace with pseudo-signatures. For the description of Π_{GA} and its properties, see Appendix A.

Component II: Asynchronous Coin-Flipping with Synchronous Liveness. We assume the existence of a p-coin-flipping protocol Π_{CoinFlip}, where $0 < p < 1$ is a parameter. In the protocol, the parties participate with random inputs and the output of each party is a bit satisfying the following properties.

- *Asynchronous Network*: The following properties are achieved even if the adversary corrupts any subset from $\mathcal{Z}_a$: **(a): $\mathcal{Z}_a$-Almost-Surely Liveness:** If all honest parties participate in the protocol, then almost-surely, all honest parties eventually get an output. **(b): $(\mathcal{Z}_a, p)$-Commonness:** With probability p, the output of *all* honest parties is a *random* bit $b \in \{0, 1\}$.

[12] Recall that we need *validity*, coupled with *guaranteed liveness* from Π_{ABA}, when used in our network agnostic BA protocol Π_{BA}.

– *Synchronous Network*: The following property is achieved even if the adversary corrupts any subset from $\mathcal{Z}_s$: **(a): $\mathcal{Z}_s$-Guaranteed Liveness:** If all honest parties participate in the protocol, then all honest parties get an output, after some *fixed* time T_{CoinFlip}.

In [23], the authors presented an instantiation of Π_{CoinFlip}, which achieves $\mathcal{Z}_a$-*Almost-Surely Liveness* as well as $(\mathcal{Z}_a, p)$-*Commonness* in an asynchronous network, where $p = \frac{1}{n}$, provided $\mathcal{Z}_a$ satisfies the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition. The protocol incurs an *expected* communication of $\mathcal{O}(\mathsf{poly}(n, |\mathcal{Z}_s|, \log |\mathbb{F}|))$ bits. Interestingly, the protocol also achieves $\mathcal{Z}_s$-*Guaranteed Liveness* in a *synchronous* network, *irrespective* of $\mathcal{Z}_s$. Namely, after time $T_{\mathsf{CoinFlip}} = 20 \cdot \Delta$, all honest parties will have an output, with (honest) parties communicating $\mathcal{O}(\mathsf{poly}(n, |\mathcal{Z}_s|, \log |\mathbb{F}|))$ bits.

3.3.1 $\Pi_{\mathsf{GA}} + \Pi_{\mathsf{CoinFlip}} \rightarrow \Pi_{\mathsf{ABA}}$

Once we have instantiations of Π_{GA} and Π_{CoinFlip}, we can easily combine it using the framework of [51,11,21,31] to get the protocol Π_{ABA}. The protocol consists of several iterations, where in each iteration, the parties run two instances of Π_{GA}, along with an instance of Π_{CoinFlip}. Using the first instance of Π_{GA}, the parties check if they all have the same input. Independent of this finding, they then run an instance of Π_{CoinFlip}. Finally, they again run an instance of Π_{GA}, with inputs being carefully chosen. Namely, if a party obtained an output with the *highest* grade from the *first* instance of Π_{GA}, then it participates with this input, else it participates with the coin-output. Finally, based on the output received from the second instance of Π_{GA}, the parties update their input for the next iteration as follows: if a bit with a non-zero grade is obtained, then it is set as the updated input, else the updated input is set to the input of the second instance of Π_{GA}. During each iteration, the parties keep a tab on whether they have received an output bit with the *highest* grade from the *second* instance of Π_{GA}, in which case, they indicate it to the others by sending a signed ready message and the bit. Once "sufficiently many" parties send the same signed ready bit, it is taken as the output of the protocol.

The idea here is that if the honest parties start an iteration with the *same* input bit b, then the output of Π_{CoinFlip} is *not* considered (*irrespective* of the network type) and *all* instances of Π_{GA} output $(b, 2)$. Thus, all honest parties will send a signed ready message for b. Consequently, all honest parties will output b. This ensures *validity*, coupled with *guaranteed liveness*, both in *synchronous* and *asynchronous* networks. On the other hand, if the honest parties start an iteration with *different* inputs, then with probability at least $p \cdot \frac{1}{2} = \frac{1}{n} \cdot \frac{1}{2}$, all of them will have the *same* input for the second instance of Π_{GA}. And consequently, all honest parties will have the same input from the next iteration onward and consistency is achieved (in the *asynchronous* network). The description of Π_{ABA} based on Π_{GA} and Π_{CoinFlip} is recalled from [23] and presented in Fig 5.

Protocol $\Pi_{\mathsf{ABA}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$

- **Initialisation**: Set $b = b_i$, $\mathsf{committed} = \mathsf{false}$ and $k = 1$. Then do the following.

1. Participate in an instance of Π_{GA} protocol with input b and wait for time T_{GA}.
2. Once an output (b, g) is received from the instance of Π_{GA}, participate in an instance of Π_{CoinFlip} and wait for time T_{CoinFlip}. Let Coin_k denote the output received from Π_{CoinFlip}.
3. If $g < 2$, then set $b = \mathsf{Coin}_k$.
4. Participate in an instance of Π_{GA} protocol with input b and wait for time T_{GA}. Let (b', g') be the output received. If $g' > 0$, then set $b = b'$.
5. If $g' = 2$ and $\mathsf{committed} = \mathsf{false}$, then set $\mathsf{committed} = \mathsf{true}$ and send $\langle(\mathsf{ready}, b)\rangle_i$ to all the parties.
6. Set $k = k + 1$ and repeat from 1.

- **Output Computation**:
  – Upon receiving $\langle(\mathsf{ready}, b)\rangle_j$ messages with valid signatures corresponding to every $P_j \in \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, send $(b, \mathcal{C}(b))$ to all the parties and output b. Here $\mathcal{C}(b)$ denotes the collection of signed $\langle(\mathsf{ready}, b)\rangle_j$ messages.
  – Upon receiving $(b, \mathcal{C}(b))$ where $\mathcal{C}(b)$ has valid signatures corresponding to every $P_j \in \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, send $(b, \mathcal{C}(b))$ to all the parties and output b.

Fig. 5: The ABA protocol. The above code is executed by every $P_i \in \mathcal{P}$ with input b_i

Theorem 3.4 follows from [15] and is proved in Appendix A.

Theorem 3.4. *Protocol* Π_{ABA} *achieves the following where* $T_{\mathsf{ABA}} = T_{\mathsf{CoinFlip}} + 2T_{\mathsf{GA}} + \Delta$.

– *Synchronous Network: If all honest parties have the same input* $b \in \{0, 1\}$*, then all honest parties output* b*, at time* T_{ABA}*. Moreover,* $\mathcal{O}(\mathsf{poly}(n, |\mathcal{Z}_s|, \log |\mathbb{F}|))$ *bits are communicated by the honest parties.*
– *Asynchronous Network: the protocol is a* $\mathcal{Z}_a$*-secure ABA, incurring an expected communication of* $\mathcal{O}(\mathsf{poly}(n, |\mathcal{Z}_s|, \log |\mathbb{F}|))$ *bits.*

4 Network Agnostic Information Checking Protocol

In this section, we present our network agnostic ICP protocol (Fig 6). A detailed overview of the protocol has been already presented in Section 1.2.2. The protocol consists of two subprotocols Π_{Auth} and Π_{Reveal}, implementing the authentication and revelation phase respectively, where the parties participate in the revelation phase only upon completing the authentication phase. During the authentication phase, S distributes the authentication and verification information, followed by parties publicly verifying the consistency of distributed information and once the consistency is established, the authentication phase is over. During the revelation phase, I reveals the IC-signature which is verified by R with respect to the verification information revealed by a "selected" subset of the verifiers.

Protocol $\Pi_{\mathsf{ICP}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathsf{S}, \mathsf{I}, \mathsf{R})$

Protocol $\Pi_{\mathsf{Auth}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathsf{S}, \mathsf{I}, \mathsf{R}, s)$: $t \overset{def}{=} \max\{|Z| : Z \in \mathcal{Z}_s\}$

- **Distributing Data**: S executes the following steps.
  - Randomly select t-degree *signing-polynomial* $F(x)$ and t-degree *masking-polynomial* $M(x)$, where $F(0) = s$. For $i = 1, \ldots, n$, randomly select $\alpha_i \in \mathbb{F} \setminus \{0\}$, and compute $v_i = F(\alpha_i)$ and $m_i = M(\alpha_i)$.
  - Send $(F(x), M(x))$ to I. For $i = 1, \ldots, n$, send (α_i, v_i, m_i) to party P_i.
- **Confirming Receipt of Verification Points**: Each party P_i (including S, I and R), upon receiving (α_i, v_i, m_i) from S, broadcasts $(\mathsf{Received}, i)$.
- **Announcing Set of Supporting Verifiers**: only S does the following.
  - Initialize the set of *supporting verifiers* $\mathcal{SV}$ to $\emptyset$, and wait till the local time is $\Delta + T_{\mathsf{BC}}$. Upon receiving $(\mathsf{Received}, i)$ from the broadcast of P_i, add P_i to $\mathcal{SV}$. Once $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, broadcast the set $\mathcal{SV}$.
- **Announcing Masked Polynomial**: only I does the following.
  - Wait till the local time is $\Delta + 2T_{\mathsf{BC}}$. Upon receiving $\mathcal{SV}$ from the broadcast of S such that $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, wait till $(\mathsf{Received}, i)$ is received from the broadcast of every $P_i \in \mathcal{SV}$. Then randomly pick $d \in \mathbb{F} \setminus \{0\}$ and broadcast $(d, B(x))$, where $B(x) \overset{def}{=} dF(x) + M(x)$.
- **Announcing Validity of Masked Polynomial**: only S does the following.
  - Wait till the local time is $\Delta + 3T_{\mathsf{BC}}$. Upon receiving $(d, B(x))$ from the broadcast of I, broadcast OK, if $B(x)$ is a t-degree polynomial and if $dv_j + m_j = B(\alpha_j)$ holds for every $P_j \in \mathcal{SV}$.
- **Deciding Whether Authentication is Successful**: each $P_i \in \mathcal{P}$ (including S, I and R) waits till the local time is $\Delta + 4T_{\mathsf{BC}}$. Upon receiving $\mathcal{SV}$ and $(d, B(x))$ from the broadcast of S and I respectively, where $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, it set the variable $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ to 1 if OK is received from the broadcast of S. Upon setting $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ to 1, I sets $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s) = F(x)$.

Protocol $\Pi_{\mathsf{Reveal}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathsf{S}, \mathsf{I}, \mathsf{R}, s)$

- **Revealing Signing Polynomial and Verification Points**: Each party P_i (including S, I and R) does the following, if $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ is set to 1.
  - If $P_i = \mathsf{I}$ then send $F(x)$ to R, if $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ is set to $F(x)$ during Π_{Auth}.
  - If $P_i \in \mathcal{SV}$, then send (α_i, v_i, m_i) to R.
- **Accepting the IC-Signature**: The following steps are executed only by R, if $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ is set to 1 during the protocol Π_{Auth}.
  - Wait till the local time becomes a multiple of Δ. Upon receiving $F(x)$ from I, where $F(x)$ is a t-degree polynomial, proceed as follows.
    1. If (α_i, v_i, m_i) is received from $P_i \in \mathcal{SV}$, then *accept* (α_i, v_i, m_i) if either $v_i = F(\alpha_i)$ or $B(\alpha_i) \neq dv_i + m_i$, where $B(x)$ is received from the broadcast of I during Π_{Auth}. Otherwise, *reject* (α_i, v_i, m_i).
    2. Wait till a subset of parties $\mathcal{SV}' \subseteq \mathcal{SV}$ is found, such that $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$, and for every $P_i \in \mathcal{SV}'$, the corresponding revealed point (α_i, v_i, m_i) is *accepted*. Then, output $s = F(0)$.

Fig. 6: The network-agnostic ICP

The proof of Theorem 4.1 is available in Appendix B.

Theorem 4.1. *Protocols* $(\Pi_{\mathsf{Auth}}, \Pi_{\mathsf{Reveal}})$ *satisfy the following properties, except with probability at most* $\epsilon_{\mathsf{ICP}} \stackrel{def}{=} \frac{nt}{|\mathbb{F}|-1}$, *where* $t = \max\{|Z| : Z \in \mathcal{Z}_s\}$.

- *If* S, I *and* R *are honest, then the following hold.*
  - $\mathcal{Z}_s$**-Correctness***: In a synchronous network, each honest party sets* $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ *to* 1 *during* Π_{Auth} *at time* $T_{\mathsf{Auth}} = \Delta + 4T_{\mathsf{BC}}$. *Moreover* R *outputs* s *during* Π_{Reveal} *which takes* $T_{\mathsf{Reveal}} = \Delta$ *time.*
  - $\mathcal{Z}_a$**-Correctness***: In an asynchronous network, each honest party eventually sets* $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ *to* 1 *during* Π_{Auth} *and* R *eventually outputs* s *during* Π_{Reveal}.
  - **Privacy***: The view of* Adv *is independent of* s*, irrespective of the network.*
- **Unforgeability***: If* S, R *are honest,* I *is corrupt and if* R *outputs* $s' \in \mathbb{F}$ *during* Π_{Reveal}, *then* $s' = s$ *holds, irrespective of the network type.*
- *If* S *is corrupt,* I, R *are honest and if* I *sets* $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s) = F(x)$ *during* Π_{Auth}, *then the following holds.*
  - $\mathcal{Z}_s$**-Non-Repudiation***: In a synchronous network,* R *outputs* $s = F(0)$ *during during* Π_{Reveal}, *which takes* $T_{\mathsf{Reveal}} = \Delta$ *time.*
  - $\mathcal{Z}_a$**-Non-Repudiation***: In an asynchronous network,* R *eventually outputs* $s = F(0)$ *during during* Π_{Reveal}.
- **Communication Complexity***: Irrespective of the network type,* Π_{Auth} *incurs a communication of* $\mathcal{O}(n^5 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ *bits, while* Π_{Reveal} *incurs a communication of* $\mathcal{O}(n \cdot \log |\mathbb{F}|)$ *bits.*

Looking ahead, in our VSS protocols, there will be several instances of ICP running, with different parties playing the role of S, I and R. It will be convenient to use the following notations while invoking instances of ICP.

Notation 4.2 (for ICP) *While using* $(\Pi_{\mathsf{Auth}}, \Pi_{\mathsf{Reveal}})$*, we will say that:*

- *"*P_i *gives* $\mathsf{ICSig}(P_i, P_j, P_k, s)$ *to* P_j*" to mean that* P_i *acts as* S *and invokes an instance of* Π_{Auth} *with input* s*, where* P_j *and* P_k *play the role of* I *and* R *respectively.*
- *"*P_j *receives* $\mathsf{ICSig}(P_i, P_j, P_k, s)$ *from* P_i*" to mean that* P_j*, as* I*, has set* $\mathsf{authCompleted}_{(P_i,P_j,P_k)}$ *to* 1 *and* $\mathsf{ICSig}(P_i, P_j, P_k, s)$ *to some* t*-degree polynomial with* s *as the constant term during the instance of* Π_{Auth}*, where* P_i *and* P_k *play the role of* S *and* R *respectively.*
- *"*P_j *reveals* $\mathsf{ICSig}(P_i, P_j, P_k, s)$ *to* P_k*" to mean* P_j*, as* I*, invokes an instance of* Π_{Reveal}*, with* P_i *and* P_k *playing the role of* S *and* R *respectively.*
- *"*P_k *accepts* $\mathsf{ICSig}(P_i, P_j, P_k, s)$*" to mean that* P_k*, as* R*, outputs* s *during the instance of* Π_{Reveal}*, invoked by* P_j *as* I*, with* P_i *playing the role of* S.

4.1 Linearity of IC Signature

Our ICP satisfies the *linearity* property, provided "special care" is taken while generating the IC-signatures. In more detail, consider a *fixed* S, I and R and let s_a and s_b be two values, such that I has received $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a)$ and $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$ from S, through instances $\Pi^{(a)}_{\mathsf{Auth}}$ and $\Pi^{(b)}_{\mathsf{Auth}}$ of Π_{Auth} respectively, where *all* the following conditions are satisfied.

– Supporting verifiers $\mathcal{SV}_a$ and $\mathcal{SV}_b$, during $\Pi^{(a)}_{\mathsf{Auth}}$ and $\Pi^{(b)}_{\mathsf{Auth}}$, are the *same*.
– For $i = 1, \ldots, n$, corresponding to the verifier P_i, S has used the *same* $\alpha_i \in \mathbb{F} \setminus \{0\}$, to compute the verification points, during $\Pi^{(a)}_{\mathsf{Auth}}$ and $\Pi^{(b)}_{\mathsf{Auth}}$
– I has used the *same* linear combiner $d \in \mathbb{F} \setminus \{0\}$ during the instances $\Pi^{(a)}_{\mathsf{Auth}}$ and $\Pi^{(b)}_{\mathsf{Auth}}$, to compute the linearly-combined masked polynomials.

Let $s \stackrel{def}{=} c_1 \cdot s_a + c_2 \cdot s_b$, where c_1, c_2 are *publicly known* constants from $\mathbb{F}$. It then follows that if all the above conditions are satisfied, then I can *locally* compute $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ from $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a)$ and $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$. Namely, I can set $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s) = c_1 \cdot \mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a) + c_2 \cdot \mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$. On the other hand, let the verifier $P_i \in \mathcal{SV}_a$ hold the verification points $(\alpha_i, v_{a,i}, m_{a,i})$ and $(\alpha_i, v_{b,i}, m_{b,i})$, corresponding to $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_a)$ and $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s_b)$ respectively. Then P_i can *locally* compute (α_i, v_i, m_i) as its verification point corresponding to $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$, where $v_i = c_1 \cdot v_{a,i} + c_2 \cdot v_{b,i}$ and $m_i = c_1 \cdot m_{a,i} + c_2 \cdot m_{b,i}$. During the protocol Π_{Reveal}, to reveal $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$, the intermediary I can reveal $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to R, while each verifier $P_i \in \mathcal{SV}_a$ can reveal the verification information (α_i, v_i, m_i). To accept (α_i, v_i, m_i), the receiver R either checks for the "consistency" of (α_i, v_i) with $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$, or the "inconsistency" of masked polynomial $B(x) \stackrel{def}{=} c_1 \cdot B_a(x) + c_2 \cdot B_b(x)$ with (d, v_i, m_i); here $B_a(x)$ and $B_b(x)$ denote the masked polynomials, made public by I, during the instances $\Pi^{(a)}_{\mathsf{Auth}}$ and $\Pi^{(b)}_{\mathsf{Auth}}$ respectively, *both* computed with respect to the linear combiner d.

Looking ahead, we will require the linearity property from ICP, when used in our VSS protocols, where there will be multiple instances of Π_{Auth} running, involving the *same* $(\mathsf{S}, \mathsf{I}, \mathsf{R})$ triplet. To achieve this, we will ensure that in all the Π_{Auth} instances invoked during VSS involving the *same* triplet $(\mathsf{S}, \mathsf{I}, \mathsf{R})$, the signer uses the same non-zero evaluation point $\alpha_{\mathsf{S},\mathsf{I},\mathsf{R},i}$ for the verifier P_i, while distributing verification information to P_i, as part of the respective Π_{Auth} instances. Similarly, S should find and make public a *common* set of supporting verifiers $\mathcal{SV}$, on behalf of *all* the instances of Π_{Auth}. And finally, I should use the same non-zero random linear combiner d, to compute the masked polynomials for all the instances of Π_{Auth} and once computed, it should *together* make public d and the masked polynomials for *all* the instances of Π_{Auth}.

In the rest of the paper, we will use the term "*parties follow linearity principle while generating IC-signatures*", to mean that the underlying instances of Π_{Auth} are invoked as above.

4.2 Default IC Signature

In our VSS protocols, we will also encounter situations where some *publicly known* value s and a triplet $(\mathsf{S}, \mathsf{I}, \mathsf{R})$ exist. Then I can *locally* compute $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ by setting $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to the *constant* polynomial $F(x) = s$. *Each* verifier $P_i \in \mathcal{P}$ *locally* sets $(\alpha_{\mathsf{S},\mathsf{I},\mathsf{R},i}, v_i, m_i)$ as its verification information, where $v_i = m_i = s$. Moreover, the set of supporting verifiers $\mathcal{SV}$ is set as $\mathcal{P}$. Notice that the way in which $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ and the verification information is set guarantees that, later, if an *honest* I reveals $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to an *honest* R during Π_{Reveal}, then R *always* outputs s.

In the rest of the paper, we will use the term "*parties set* $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ *to the default value*", to mean the above.

5 Network Agnostic Verifiable Secret Sharing (VSS)

This section presents our network-agnostic VSS protocol, which allows a designated dealer to generate a linear secret-sharing with IC-signatures for its input. We first define the notion of linear secret-sharing with IC-signatures.

Definition 5.1 (Linear Secret Sharing with IC-Signatures). *A value $s \in \mathbb{F}$ is said to be linearly secret-shared with IC-signatures, if there exist shares $s_1, \ldots, s_{|\mathcal{Z}_s|} \in \mathbb{F}$ where $s = s_1 + \ldots + s_{|\mathcal{Z}_s|}$. Moreover, for $q = 1, \ldots, |\mathcal{Z}_s|$, there exists some publicly-known core-set $\mathcal{W}_q \subseteq S_q$, such that all the following hold.*

- *$\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition and all (honest) parties in the set S_q have the share s_q.*
- *Every honest $P_i \in \mathcal{W}_q$ has the IC-signature $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$ of every $P_j \in \mathcal{W}_q$ for every $P_k \notin S_q$. Moreover, if any corrupt $P_j \in \mathcal{W}_q$ has $\mathsf{ICSig}(P_j, P_i, P_k, s'_q)$ of any honest $P_i \in \mathcal{W}_q$ for any $P_k \notin S_q$, then $s'_q = s_q$ holds. Furthermore, all the underlying IC-signatures satisfy the linearity property.*

The vector of information corresponding to a linear secret-sharing with IC-signature of s is denoted by $[s]$, which includes the share $s_1, \ldots, s_{|\mathcal{Z}_s|}$, the core sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$ and IC-signatures $\{\mathsf{ICSig}(P_j, P_i, P_k, s_q)\}_{P_j, P_i \in \mathcal{W}_q, P_k \notin S_q}$. For convenience, we denote the q^{th} share of s, corresponding to S_q, by $[s]_q$.

A vector of values $\vec{S} = (s^{(1)}, \ldots, s^{(L)})$ where $L \geq 1$ is said to be linearly secret-shared with IC-signatures, if each $s^{(\ell)} \in \vec{S}$ is linearly secret-shared with IC-signatures and if there exist common core sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$, corresponding to the secret-sharings $[s^{(1)}], \ldots, [s^{(\ell)}]$.

If $\vec{S} = (s^{(1)}, \ldots, s^{(L)})$ are linearly secret-shared with IC-signatures, then the parties can *locally* compute any publicly-known function of these secret-shared values. In more detail, let $c_1, \ldots, c_L \in \mathbb{F}$ be publicly-known constants and let $s \stackrel{def}{=} c_1 \cdot s^{(1)} + \ldots + c_L \cdot s^{(L)}$. Then the following holds:

$$c_1 \cdot [s^{(1)}] + \ldots + c_L \cdot [s^{(L)}] = [s],$$

where the core-sets corresponding to $[s]$ are $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$. And corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, the share $[s]_q$ for each (honest) party in S_q can be computed locally as $c_1 \cdot [s^{(1)}]_q + \ldots + c_\ell \cdot [s^{(\ell)}]_q$. Moreover, every (honest) $P_i \in \mathcal{W}_q$ can compute the IC-signature $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ of every $P_j \in \mathcal{W}_q$ for every $P_k \notin S_q$, from $\mathsf{ICSig}(P_j, P_i, P_k, [s^{(1)}]_q), \ldots, \mathsf{ICSig}(P_j, P_i, P_k, [s^{(\ell)}]_q)$. In the rest of the paper, we will say that the "*parties in $\mathcal{P}$ locally compute* $[c_1 \cdot s^{(1)} + \ldots + c_\ell \cdot s^{(\ell)}]$ *from* $[s^{(1)}], \ldots, [s^{(\ell)}]$" to mean the above.

5.1 The VSS Protocol

We present a network-agnostic VSS protocol Π_{VSS} (Fig 7). In the protocol, there exists a designated *dealer* $\mathsf{D} \in \mathcal{P}$ with input s (the protocol can be easily generalized if D has L inputs). The protocol allows D to "verifiably" generate a linear secret-sharing of s with IC-signatures. In a *synchronous* network, the (honest) parties output $[s]$ after a "fixed" time, while in an *asynchronous* network, they do so *eventually*, such that s remains private. The verifiability here guarantees that if D is *corrupt* and some honest party gets an output, then there exists some value, say $s^\star$ (which could be different from s), such that $s^\star$ is linearly secret-shared with IC-signatures. Note that in this case, we *cannot* bound the time within which $s^\star$ will be secret-shared, since a potentially *corrupt* D may delay sending the required messages and the parties will *not* be knowing the exact network type. A detailed overview of the protocol has been already presented in Section 1.2.3 and so we directly present the protocol.

Protocol $\Pi_{\mathsf{VSS}}(\mathsf{D}, \mathcal{Z}_s, \mathcal{Z}_a, s, \mathbb{S}_{\mathcal{Z}_s})$

- **Distribution of Shares**: D, on having input s, randomly chooses $s_1, \ldots, s_{|\mathcal{Z}_s|} \in \mathbb{F}$, such that $s = s_1 + \cdots + s_{|\mathcal{Z}_s|}$. It then sends s_q to all $P_i \in S_q$, for $q = 1, \ldots, |\mathcal{Z}_s|$.
- **Exchanging IC-Signed Values**: Each $P_i \in \mathcal{P}$ (including D), waits till the local time becomes Δ. Then, for each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ such that $P_i \in S_q$, upon receiving s_{qi} from D, give $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$, such that the parties follow the linearity principle while generating IC-signatures (see Section 4.1).
- **Announcing Results of Pairwise Consistency Tests**: Each $P_i \in \mathcal{P}$ (including D) waits till the local time becomes $\Delta + T_{\mathsf{Auth}}$ and then does the following.
  - Upon receiving $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ from P_j for each $S_q \in \mathbb{S}$ such that $P_j, P_i \in S_q$, corresponding to every $P_k \in \mathcal{P}$, broadcast $\mathsf{OK}(i, j)$, if $s_{qi} = s_{qj}$ holds.
  - Corresponding to every $P_j \in \mathcal{P}$, participate in any instance of Π_{BC} initiated by P_j as a sender, to broadcast any $\mathsf{OK}(P_j, \star)$ message.
- **Constructing Consistency Graph**: Each $P_i \in \mathcal{P}$ (including D) waits till the local time becomes $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$ and then constructs an undirected consistency graph $G^{(i)}$ with $\mathcal{P}$ as the vertex set, where the edge (P_j, P_k) is added to $G^{(i)}$, provided $\mathsf{OK}(j, k)$ and $\mathsf{OK}(k, j)$ is received from the broadcast of P_j and P_k respectively (through any mode).
- **Identification of Core Sets and Public Announcement by the Dealer**: D waits till its local time is $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, and then executes the following steps to compute core sets.
  - Once any $S_p \in \mathbb{S}_{\mathcal{Z}_s}$ forms a clique in the graph $G^{(\mathsf{D})}$, then for $q = 1, \ldots, |\mathcal{Z}_s|$, compute core-set $\mathcal{W}_q$ and broadcast-set $\mathcal{BS}$ with respect to S_p as follows, followed by broadcasting $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$.[a]
    - If S_q constitutes a clique in the graph $G^{(\mathsf{D})}$, then set $\mathcal{W}_q = S_q$.
    - Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\mathsf{D})}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then set $\mathcal{W}_q = (S_p \cap S_q)$.
    - Else set $\mathcal{W}_q = S_q$ and include q to $\mathcal{BS}$.
- **Identifying Valid Core Sets**: Each $P_i \in \mathcal{P}$ waits till its local time is $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$ and then initializes a set $\mathcal{C}_i = \emptyset$. For $p = 1, \ldots, |\mathcal{Z}_s|$, party P_i includes (D, S_p) to $\mathcal{C}_i$ (initialized to $\emptyset$), provided *all* the following hold.

– $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ is received from the broadcast of D, such that for $q = 1, \ldots, |\mathcal{Z}_s|$, the following hold.
  – If $q \in \mathcal{BS}$, then the set $\mathcal{W}_q = S_q$.
  – If $(q \notin \mathcal{BS})$, then $\mathcal{W}_q$ is either S_q or $(S_p \cap S_q)$, such that:
    – If $\mathcal{W}_q = S_q$, then S_q constitutes a clique in $G^{(i)}$.
    – Else if $\mathcal{W}_q = (S_p \cap S_q)$, then $(S_p \cap S_q)$ constitutes a clique in $G^{(i)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition.

– **Computing Output**: Each $P_i \in \mathcal{P}$ does the following, once $\mathcal{C}_i \neq \emptyset$.
  – For every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ such that $P_i \in \mathcal{W}_q$, corresponding to every $P_j \in \mathcal{W}_q$, reveal $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ to every $P_k \in \mathsf{S}_q \setminus \mathcal{W}_q$ upon computing $[s]_q$ and $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ as follows.
    – If $q \in \mathcal{BS}$, then set $[s]_q = s_q$, where s_q is received from the broadcast of D, as part of the message $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$. Moreover, for every $P_j \in S_q$ and every $P_k \in \mathcal{P}$, set $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ to the default value.
    – Else, set $[s]_q$ to s_{qi}, where s_{qi} was received from D. Moreover, for every $P_j \in \mathcal{W}_q$ and every $P_k \in \mathcal{P}$, set $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ to $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$, received from P_j.
  – For every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ such that $P_i \in S_q \setminus W_q$, compute $[s]_q$ as follows.
    – Check if there exists any $P_j \in \mathcal{W}_q$ and a value s_{qj}, such that P_i has accepted $\mathsf{ICSig}(P_k, P_j, P_i, s_{qj})$, corresponding to every $P_k \in \mathcal{W}_q$. Upon finding such a P_j, set $[s]_q = s_{qj}$.
  – Wait till the local time becomes $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$. Upon setting $\{[s]_q\}_{P_i \in S_q}$ to some value, output $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$, $\{[s]_q\}_{P_i \in S_q}$ and $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)_{P_j, P_i \in \mathcal{W}_q, P_k \notin S_q}$.

[a] If there are multiple S_p from $\mathbb{S}_{\mathcal{Z}_s}$ which constitute a clique in $G^{(\mathsf{D})}$, then consider the one with the smallest index.

Fig. 7: The network agnostic VSS protocol

The properties of Π_{VSS} as stated in Theorem 5.1 are proved in Appendix C.

Theorem 5.1. *Protocol Π_{VSS} achieves the following, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$, where D has input $s \in \mathbb{F}$ for Π_{VSS} and where $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$.*

– *If D is honest, then the following hold.*
  – **$\mathcal{Z}_s$-correctness**: *In a synchronous network, the honest parties output $[s]$ at time T_{VSS}.*
  – **$\mathcal{Z}_a$-correctness**: *In an asynchronous network, the honest parties eventually output $[s]$.*
  – **Privacy**: *Adversary's view remains independent of s in any network.*
– *If D is corrupt, then the following hold.*
  – **$\mathcal{Z}_s$-commitment**: *In a synchronous network, either no honest party computes any output or there exists some $s^\star \in \mathbb{F}$, such that the honest parties output $[s^\star]$. Moreover, if any honest party computes its output at time T, then all honest parties compute their required output by time $T + \Delta$.*

- $\mathcal{Z}_a$**-commitment**: *In an asynchronous network, either no honest party computes any output or there exists some $s^\star \in \mathbb{F}$, such that the honest parties eventually output $[s^\star]$.*
- **Communication Complexity**: $\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ *bits are communicated by the honest parties.*

6 Network Agnostic Reconstruction Protocols and Secure Verifiable Multicast

Let s be a value which is linearly secret-shared with IC-signatures and let $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. Moreover, let $\mathcal{R} \subseteq \mathcal{P}$ be a designated set. Then protocol $\Pi_{\mathsf{RecShare}}([s], S_q, \mathcal{R})$ allows all the (honest) parties in $\mathcal{R}$ to reconstruct the share $[s]_q$ without disclosing any additional information. For this, every $P_i \in \mathcal{W}_q$ reveals $[s]_q$ to *all* the parties *outside* $\mathcal{W}_q$, who are in $\mathcal{R}$ (the parties in $\mathcal{W}_q$ who are in $\mathcal{R}$ *already* have $[s]_q$). To ensure that P_i *does not* cheat, P_i actually reveals the IC-signature of *every* party in $\mathcal{W}_q$ on the revealed $[s]_q$. The idea here is that since $\mathcal{W}_q$ has *at least* one *honest* party (irrespective of the network type), a potentially *corrupt* P_i will fail to reveal the signature of an *honest* party from $\mathcal{W}_q$ on an *incorrect* $[s]_q$. On the other hand, an *honest* P_i will be able to reveal the signature of *all* the parties in $\mathcal{W}_q$ on $[s]_q$.

Based on Π_{RecShare}, we design another protocol $\Pi_{\mathsf{Rec}}([s], \mathcal{R})$, which allows all the (honest) parties in $\mathcal{R}$ to reconstruct s. The idea is to run an instance of Π_{RecShare} for *every* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. Since the protocols are standard, we present them and prove there properties (Lemma 6.1 and Lemma 6.2) in Appendix D.

Lemma 6.1. *Let s be a value which is linearly secret-shared with IC signatures, let $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ be a designated set and let $\mathcal{R} \subseteq \mathcal{P}$ be a designated set of receivers. Then protocol Π_{RecShare} achieves the following.*

- $\mathcal{Z}_s$**-correctness**: *In a synchronous network, all honest parties in $\mathcal{R}$ output $[s]_q$ after time $T_{\mathsf{RecShare}} = T_{\mathsf{Reveal}}$, except with a probability of $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- $\mathcal{Z}_a$**-correctness**: *In an asynchronous network, all honest parties in $\mathcal{R}$ eventually output $[s]_q$, except with a probability of $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- **Privacy**: *If $\mathcal{R}$ consists of only honest parties, then the view of the adversary remains independent of $[s]_q$.*
- **Communication Complexity**: $\mathcal{O}(|\mathcal{R}| \cdot n^3 \cdot \log |\mathbb{F}|)$ *bits are communicated.*

Lemma 6.2. *Let s be a value which is linearly secret-shared with IC signatures and let $\mathcal{R} \subseteq \mathcal{P}$ be a set of designated receivers. Then protocol Π_{Rec} achieves the following.*

- $\mathcal{Z}_s$**-correctness**: *In a synchronous network, all honest parties in $\mathcal{R}$ output s after time $T_{\mathsf{Rec}} = T_{\mathsf{RecShare}}$, except with probability $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- $\mathcal{Z}_a$**-correctness**: *In an asynchronous network, all honest parties in $\mathcal{R}$ eventually output s, except with probability $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- **Privacy**: *If $\mathcal{R}$ consists of only honest parties, then the view of the adversary remains independent of s.*
- **Communication Complexity**: $\mathcal{O}(|\mathcal{Z}_s| \cdot |\mathcal{R}| \cdot n^3 \cdot \log |\mathbb{F}|)$ *bits are communicated.*

6.1 Π_{VSS} and Reconstruction Protocol for Superpolynomial $|\mathcal{Z}_s|$

From Theorem 5.1 and Lemma 6.2, the error probability of Π_{VSS} and Π_{Rec} depend linearly on $|\mathbb{S}_{\mathcal{Z}_s}|$, which is the same as $|\mathcal{Z}_s|$. This is because there are $\Omega(|\mathcal{Z}_s|)$ instances of $\Pi_{\mathsf{Auth}}/\Pi_{\mathsf{Reveal}}$ in which the *unforgeability/non-repudiation* properties might get violated with probability ϵ_{ICP}. This might be problematic for a "large-sized" $\mathcal{Z}_s$. To avoid this, we use the idea of *local dispute control* used in [40,4], which ensures that *irrespective* of the number of instances of $\Pi_{\mathsf{Auth}}/\Pi_{\mathsf{Reveal}}$, the *overall* error probability is *only* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. This is done by ensuring that the *unforgeability/non-repudiation* properties get violated *only* $\mathcal{O}(n^3)$ times across all these instances. The idea here is that the parties start *locally discarding* corrupt parties the "moment" they are caught cheating during *any* instance of Π_{Auth} or Π_{Reveal}. Once a party P_j is locally discarded by any party P_i, then P_i "behaves" as if P_j has *certainly* behaved maliciously in all the "future" instances of Π_{Auth} or Π_{Reveal}, even if this is *not* the case. This restricts the number of attempts of cheating for the adversary to a "fixed" number and consequently, the *total* error probability of *arbitrary* many instances of $\Pi_{\mathsf{Auth}}/\Pi_{\mathsf{Reveal}}$ will *no* longer depend on $|\mathcal{Z}_s|$. To incorporate the above idea, each party P_i now maintains a list of *locally discarded* parties $\mathcal{LD}^{(i)}$, which it keeps populating across all the instances of Π_{Auth} and Π_{Reveal}, as soon as P_i identifies any party cheating. It will be ensured that an *honest* P_i *never* includes an *honest* P_j to $\mathcal{LD}^{(i)}$. We next discuss the modifications in Π_{Auth} and Π_{Reveal} and how the parties populate their $\mathcal{LD}$ sets across instances of Π_{Auth} and Π_{Reveal}.

Populating $\mathcal{LD}$ Sets During Instances of Π_{Auth}. In any instance of Π_{Auth}, if P_i is present in the corresponding set of supporting verifiers $\mathcal{SV}$ (i.e. $P_i \in \mathcal{SV}$), then P_i includes the corresponding *signer* P_j of the Π_{Auth} instance to $\mathcal{LD}^{(i)}$ if the following condition holds during Π_{Auth}:

$$(P_j \text{ broadcasts } \mathsf{OK}) \quad \wedge \quad (B(\alpha_i) \neq dv_i + m_i),$$

where $B(x)$ is the masked polynomial broadcasted by the corresponding I of the Π_{Auth} instance. The idea here is that if P_i is *honest* and if the above condition holds, then clearly the signer P_j is *corrupt* and is trying to break the *non-repudiation* property. Once $P_j \in \mathcal{LD}^{(i)}$, then in any pair of $(\Pi_{\mathsf{Auth}}, \Pi_{\mathsf{Reveal}})$ instances involving P_j as the *signer*, if P_i is present in the corresponding set $\mathcal{SV}$, then in the Π_{Reveal} instance, P_i reveals $\perp$ as its verification information to the corresponding *receiver* R.[13] Upon receiving $\perp$ as the verification information, the strategy for R is to *always* accepts it *without* doing any verification, *irrespective* of the polynomial $F(x)$ revealed as ICSig by the corresponding I.

The above modification ensures that if in any instance of Π_{Auth} involving a *corrupt signer* P_j and an *honest* I, P_j distributes an inconsistent verification point to an *honest verifier* P_i from the corresponding $\mathcal{SV}$ set and *still* broadcasts an OK message during Π_{Auth}, then P_j will locally be discarded by the verifier P_i, except with probability ϵ_{ICP} (follows from the *non-repudiation* property of ICP). From

[13] This serves as an indicator for R that P_i is in conflict with the signer P_j.

then onwards, in *all* the instances of ($\Pi_{\mathsf{Auth}}, \Pi_{\mathsf{Reveal}}$), involving P_j as the *signer*, if the verifier P_i is added to the $\mathcal{SV}$ set, then the "special" verification information revealed by P_i during Π_{Reveal} will *always* be considered as accepted, irrespective of what verification information it actually receives from P_j during Π_{Auth}. Hence, P_j will *not* have any chance of cheating the verifier P_i in any Π_{Auth} instance. By considering all possibilities for a *corrupt* S and an *honest* verifier P_i, along with an *honest* I, it follows that except with probability at most $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the verification-points of *all* honest verifiers from corresponding $\mathcal{SV}$, will be *accepted* by every *honest* R, during *all* the instances of Π_{Reveal}, in any instance of Π_{VSS} or Π_{Rec}. Consequently, except with probability at most $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the signatures revealed by all *honest* I will be *always accepted*.

We stress that the above modification *does not* help a *corrupt* I to break the *unforgeability* property for an *honest* S and an *honest* R, with the help of potentially *corrupt* verifiers.

Populating $\mathcal{LD}$ Sets During Instances of Π_{Reveal}. Consider an instance of Π_{Reveal} involving P_i as R and P_j as I. If P_i finds that P_j has tried to *forge* signature on an *incorrect* value, then P_i adds P_j to $\mathcal{LD}^{(i)}$. To achieve this goal, during Π_{Reveal}, P_i (as R) now *additionally* checks if there exists a subset of verifiers $\mathcal{SV}'' \subseteq \mathcal{SV}$, where $\mathcal{SV} \setminus \mathcal{SV}'' \in \mathcal{Z}_a$, such that the verification-points of *all* the parties in $\mathcal{SV}''$ are *rejected*. If such a subset $\mathcal{SV}''$ exists, then clearly P_j (as I) has cheated and tried to break the *unforgeability* property, since $\mathcal{SV}''$ is bound to contain at least one *honest* verifier. If the verification point of an honest verifier is *rejected*, then clearly P_j is *corrupt*. Once $P_j \in \mathcal{LD}^{(i)}$, from then onwards, in any instance of Π_{Reveal} involving P_j as I and P_i as R, party P_i *always rejects* any IC-signature revealed by P_i.

The above modification ensures that if in any instance of Π_{Reveal} involving an *honest* signer, a *corrupt intermediary* P_j and an *honest receiver* P_i, P_j tries to reveal an incorrect signature during Π_{Reveal}, then except with probability ϵ_{ICP}, the intermediary P_j will be locally discarded by the receiver P_i (follows from the *unforgeability* property of ICP). From then onwards, in *all* the instances of ($\Pi_{\mathsf{Auth}}, \Pi_{\mathsf{Reveal}}$), involving P_j as the intermediary and P_i as the receiver, the signature revealed by P_j during Π_{Reveal} will *always* be rejected, irrespective of what data is actually revealed by P_j. Hence, by considering all possibilities for a *corrupt* I, *honest* S and *honest* R, it follows that except with probability at most $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, no *corrupt* I will be able to forge an *honest* S's signature to any *honest* R, in any instance of Π_{Reveal}, during any instance of Π_{VSS} or Π_{Rec}.

6.2 Network Agnostic Secure Multicast

Based on protocols Π_{VSS} and Π_{Rec}, we design a secure verifiable multicast protocol Π_{SVM}. In the protocol, there exists a designed *sender* $\mathsf{Sen} \in \mathcal{P}$ with input $v \in \mathbb{F}$ and a designated set of *receivers* $\mathcal{R}$. The goal is to let every party in $\mathcal{R}$ receive v, without revealing any additional information to the adversary.[14] While

[14] Note that the requirements here are different from broadcast since we need the *privacy* of v if Sen is *honest* and if $\mathcal{R}$ consists of *only* honest parties.

in a *synchronous* network, the (honest) parties in $\mathcal{R}$ get v after a "fixed" time, in an *asynchronous* network, they do so eventually. Note that if Sen is *corrupt*, then the parties *need not* obtain any output, as Sen *may not* invoke the protocol. However, if *any* honest party in $\mathcal{R}$ computes an output $v^\star$ (which could be *different* from v), then *all* honest parties in $\mathcal{R}$ will also output $v^\star$. The "verifiability" here guarantees that in case the honest parties in $\mathcal{R}$ get any output, then *all* the (honest) parties in $\mathcal{P}$ will be "aware" of this; namely there will be a Boolean variable $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$, which all the honest parties will set to 1.

The idea behind Π_{SVM} is very simple. The parties participate in an instance of Π_{VSS}, where Sen plays the role of the *dealer* with input v. Once any (honest) party computes an output during Π_{VSS} (implying that Sen is committed to some value $v^\star$ which is the same as v for an *honest* Sen), then it turns $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1. Once $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ is turned to 1, the parties invoke an instance of Π_{Rec} to let *only* the parties in $\mathcal{R}$ reconstruct the committed value. Protocol Π_{SVM} and proof of its properties (stated in Lemma 6.3), are available in Appendix D.

Lemma 6.3. *Protocol Π_{SVM} achieves the following, where Sen participates with input v and where each honest party initializes $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 0.*

- **Synchronous Network***: If Sen is honest, then all honest parties set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 at time T_{VSS} and except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ output v, after time $T_{\mathsf{SVM}} = T_{\mathsf{VSS}} + T_{\mathsf{Rec}}$. Moreover, if $\mathcal{R}$ consists of only honest parties, then the view of Adv remains independent of v. If Sen is corrupt and some honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1, then there exists some $v^\star$ such that, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ output $v^\star$. Moreover, if any honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 at time T, then all honest parties in $\mathcal{R}$ output $v^\star$ by time $T + 2\Delta$.*
- **Asynchronous Network***: If Sen is honest, then all honest parties eventually set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 and except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ eventually output v. Moreover, if $\mathcal{R}$ consists of only honest parties, then the view of the adversary remains independent of v. If Sen is corrupt and some honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1, then there exists some $v^\star$ such that , except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ eventually output $v^\star$.*
- **Communication Complexity***: $\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits are communicated.*

7 Network Agnostic Protocol for Generating Linearly Secret-Shared Random Values with IC-Signatures

In this section, we present a network agnostic protocol Π_{Rand}, which allows the parties to jointly generate linear secret-sharing of random values with IC-signatures. To design the protocol Π_{Rand}, we first design a subprotocol Π_{MDVSS}.

7.1 Network Agnostic VSS for Multiple Dealers

Protocol Π_{MDVSS} (Fig 8) is a multi-dealer VSS. In the protocol, each party $P_\ell \in \mathcal{P}$ participates as a dealer with some input $s^{(\ell)}$. Then, *irrespective* of the network type, the protocol outputs a *common* subset of dealers $\mathsf{CORE} \subseteq \mathcal{P}$, which is *guaranteed* to have at least one *honest* dealer. Moreover, corresponding to every dealer $P_\ell \in \mathsf{CORE}$, there will be some value, say $s^{\star(\ell)}$, which will be the same as $s^{(\ell)}$ for an *honest* P_ℓ, such that the values $\{s^{\star(\ell)}\}_{P_\ell \in \mathsf{CORE}}$ are linearly secret-shared with IC-signatures. While in a *synchronous* network, $\{[s^{\star(\ell)}]\}_{P_\ell \in \mathsf{CORE}}$ is generated after a "fixed" time, in an *asynchronous* network, $\{[s^{\star(\ell)}]\}_{P_\ell \in \mathsf{CORE}}$ is generated eventually.

The high level overview of Π_{MDVSS} has been already discussed in detail in Section 1.2.5.[15] The idea is to let every dealer P_ℓ to invoke an instance of Π_{VSS} to secret-share its input. However, we need to take special care to ensure that the inputs of all the dealers in CORE are secret-shared with *common* core-sets. For this, each individual dealer in its instance of Π_{VSS} computes and publishes as many "legitimate" core-sets as possible and the parties run instances of *agreement on common subset* (ACS) to identify whether "sufficiently many" dealers have published the same legitimate core-sets in their respective instances of Π_{VSS}. Moreover, to ensure that all the underlying IC-signatures satisfy the linearity property, we first need to identify the dealers who distribute shares as part of their respective Π_{VSS} instances. For this, we let each dealer distribute shares in its instance of Π_{VSS} through instances of Π_{SVM}. This enables the parties to identify a set of *committed dealers* $\mathcal{CD}$ who have indeed distributed shares as part of their Π_{VSS} instances through instances of Π_{SVM}.

Protocol $\Pi_{\mathsf{MDVSS}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, (s^{(1)}, \ldots, s^{(n)}), \mathbb{S}_{\mathcal{Z}_s})$

- **Committing Shares**: Each $P_i \in \mathcal{P}$ executes the following steps.
  - On having input $s^{(i)}$, randomly choose $s^{(i)}_1, \ldots, s^{(i)}_{|\mathcal{Z}_s|}$, such that $s^{(i)} = s^{(i)}_1 + \cdots + s^{(i)}_{|\mathcal{Z}_s|}$. Act as Sen and invoke instances $\Pi_{\mathsf{SVM}}(P_i, s^{(i)}_1, S_1), \ldots, \Pi_{\mathsf{SVM}}(P_i, s^{(i)}_{|\mathcal{Z}_s|}, S_{|\mathcal{Z}_s|})$ of Π_{SVM}.
  - Corresponding to every dealer $P_\ell \in \mathcal{P}$, participate in the instances of Π_{SVM}, invoked by P_ℓ as a Sen and wait till the local time becomes T_{SVM}. For $q = 1, \ldots, |\mathcal{Z}_s|$, let $\mathsf{flag}^{(P_\ell, S_q)}$ be the Boolean flag, corresponding to the instance $\Pi_{\mathsf{SVM}}(P_\ell, s^{(\ell)}_q, S_q)$, invoked by P_ℓ.
- **Identifying the Set of Committed Dealers Through ACS**: Each $P_i \in \mathcal{P}$ does the following.
  - For $\ell = 1, \ldots, n$, participate in an instance $\Pi^{(\ell)}_{\mathsf{BA}}$ of Π_{BA} with input 1, provided P_i has set $\mathsf{flag}^{(P_\ell, S_q)} = 1$, for $q = 1, \ldots, |\mathcal{Z}_s|$.
  - Once there exists a subset of dealers $\mathcal{CD}_i$ where $\mathcal{P} \setminus \mathcal{CD}_i \in \mathcal{Z}_s$, such that corresponding to every dealer $P_\ell \in \mathcal{CD}_i$, the instance $\Pi^{(\ell)}_{\mathsf{BA}}$ has produced output 1, then participate with input 0 in all the BA instances $\Pi^{(\star)}_{\mathsf{BA}}$, for which no input is provided yet.

[15] Actually, the overview was for the protocol Π_{Rand}, but the same idea is also used in the protocol Π_{MDVSS}.

- Once all the n instances of $\Pi_{\mathsf{BA}}^{(\star)}$ have produced a binary output, set $\mathcal{CD}$ to be the set of dealers P_ℓ, such that $\Pi_{\mathsf{BA}}^{(\ell)}$ has produced output 1.

- **Exchanging IC-Signed Values**: Each $P_i \in \mathcal{P}$ waits till the local time becomes $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$. Then corresponding to each dealer $P_\ell \in \mathcal{CD}$, does the following.
  - For each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ such that $P_i \in S_q$, upon computing an output $s_{qi}^{(\ell)}$ during $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$, give $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi}^{(\ell)})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$, where the parties follow the linearity principle while generating IC-signatures.
- **Announcing Results of Pairwise Consistency Tests**: Each $P_i \in \mathcal{P}$ waits till the local time becomes $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}}$ and then does the following, corresponding to each dealer $P_\ell \in \mathcal{CD}$.
  - Upon receiving $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj}^{(\ell)})$ from P_j for each $S_q \in \mathbb{S}$ such that $P_j, P_i \in S_q$, corresponding to every $P_k \in \mathcal{P}$, broadcast $\mathsf{OK}^{(\ell)}(i, j)$, if $s_{qi}^{(\ell)} = s_{qj}^{(\ell)}$ holds.
  - Corresponding to every $P_j \in \mathcal{P}$, participate in any instance of Π_{BC} initiated by P_j as a sender, to broadcast any $\mathsf{OK}^{(\ell)}(P_j, \star)$ message.
- **Constructing Consistency Graphs**: Each $P_i \in \mathcal{P}$ waits till the local time becomes $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$ and then does the following, corresponding to each dealer $P_\ell \in \mathcal{CD}$.
  - Construct an undirected consistency graph $G^{(\ell,i)}$ with $\mathcal{P}$ as the vertex set, where the edge (P_j, P_k) is added to $G^{(\ell,i)}$, provided $\mathsf{OK}^{(\ell)}(j,k)$ and $\mathsf{OK}^{(\ell)}(k,j)$ is received from the broadcast of P_j and P_k respectively (through any mode).
- **Public Announcement of Core Sets by the Committed Dealers**: Each dealer $P_\ell \in \mathcal{CD}$ waits till its local time is $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, and then executes the following steps to compute core sets.
  - $\forall S_p \in \mathbb{S}_{\mathcal{Z}_s}$, once S_p forms a clique in $G^{(\ell,\ell)}$, then for $q = 1, \ldots, |\mathcal{Z}_s|$, compute core-set $\mathcal{W}_{p,q}^{(\ell)}$ and broadcast-set $\mathcal{BS}_p^{(\ell)}$ with respect to S_p as follows, followed by broadcasting $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$.
    - If S_q constitutes a clique in the graph $G^{(\ell,\ell)}$, then set $\mathcal{W}_{p,q}^{(\ell)} = S_q$.
    - Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,\ell)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then set $\mathcal{W}_{p,q}^{(\ell)} = (S_p \cap S_q)$.
    - Else set $\mathcal{W}_{p,q}^{(\ell)} = S_q$ and include q to $\mathcal{BS}_p^{(\ell)}$.
- **Identifying Valid Core Sets**: Each $P_i \in \mathcal{P}$ waits for time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$ and then initializes a set $\mathcal{C}_i = \emptyset$. Corresponding to $P_\ell \in \mathcal{CD}$ and $p = 1, \ldots, |\mathcal{Z}_s|$, party P_i includes (P_ℓ, S_p) to $\mathcal{C}_i$, provided *all* the following hold.
  - $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ is received from the broadcast of P_ℓ, such that for $q = 1, \ldots, |\mathcal{Z}_s|$, the following hold.
    - If $q \in \mathcal{BS}_p^{(\ell)}$, then the set $\mathcal{W}_{p,q}^{(\ell)} = S_q$.
    - If $(q \notin \mathcal{BS}_p^{(\ell)})$, then $\mathcal{W}_{p,q}^{(\ell)}$ is either S_q or $(S_p \cap S_q)$, such that:
      - If $\mathcal{W}_{p,q}^{(\ell)} = S_q$, then S_q constitutes a clique in $G^{(\ell,i)}$.
      - Else if $\mathcal{W}_{p,q}^{(\ell)} = (S_p \cap S_q)$, then $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,i)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition.
- **Selecting the Common Committed Dealers and Core Sets through ACS**: Each party $P_i \in \mathcal{P}$ does the following.

- For $p = 1, \ldots, |\mathcal{Z}_s|$, participate in an instance $\Pi_{\mathsf{BA}}^{(1,p)}$ of Π_{BA} with input 1, provided there exists a set of dealers $\mathcal{A}_{p,i} \subseteq \mathcal{CD}$ where $\mathcal{CD} \setminus \mathcal{A}_{p,i} \in \mathcal{Z}_s$ and where $(P_\ell, S_p) \in \mathcal{C}_i$ for every $P_\ell \in \mathcal{A}_{p,i}$.
- Once any instance of $\Pi_{\mathsf{BA}}^{(1,\star)}$ has produced an output 1, participate with input 0 in all the BA instances $\Pi_{\mathsf{BA}}^{(1,\star)}$, for which no input is provided yet.
- Once all the $|\mathcal{Z}_s|$ instances of $\Pi_{\mathsf{BA}}^{(1,\star)}$ have produced a binary output, set q_{core} to be the least index among $\{1, \ldots, |\mathcal{Z}_s|\}$, such that $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$ has produced output 1.
- Once q_{core} is computed, then corresponding to each $P_j \in \mathcal{CD}$, participate in an instance $\Pi_{\mathsf{BA}}^{(2,j)}$ of Π_{BA} with input 1, provided $(P_j, S_{q_{\mathsf{core}}}) \in \mathcal{C}_i$.
- Once there exists a set of parties $\mathcal{B}_i \subseteq \mathcal{CD}$, such that $\mathcal{CD} \setminus \mathcal{B}_i \in \mathcal{Z}_s$ and $\Pi_{\mathsf{BA}}^{(2,j)}$ has produced output 1, corresponding to each $P_j \in \mathcal{B}_i$, participate with input 0 in all the instances of $\Pi_{\mathsf{BA}}^{(2,\star)}$, for which no input is provided yet.
- Once all the $|\mathcal{CD}|$ instances of $\Pi_{\mathsf{BA}}^{(2,\star)}$ have produced a binary output, include all the parties P_j from $\mathcal{CD}$ in CORE (initialized to $\emptyset$), such that $\Pi_{\mathsf{BA}}^{(2,j)}$ has produced output 1.

– **Computing Output**: Each $P_i \in \mathcal{P}$ does the following, after computing CORE and q_{core}.
  - If $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(j)}})$ is not yet received from the broadcast of P_ℓ for for any $P_\ell \in \mathsf{CORE}$, then wait to receive it from the broadcast of P_ℓ through fallback-mode.
  - Once $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(j)}})$ is available for every $P_\ell \in \mathsf{CORE}$, compute $\mathcal{W}_q$ for $q = 1, \ldots, |\mathcal{Z}_s|$ as follows.
    - If $\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)} = S_q$ for every $P_\ell \in \mathsf{CORE}$, then set $\mathcal{W}_q = S_q$.
    - Else set $\mathcal{W}_q = (S_{q_{\mathsf{core}}} \cap S_q)$.
  - Corresponding to every $P_\ell \in \mathsf{CORE}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ such that $P_i \in S_q$, compute the output as follows.
    - If $q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}$, then set $[s^{(\ell)}]_q = s_q^{(\ell)}$, where $s_q^{(\ell)}$ was received from the broadcast of P_ℓ, as part of $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(j)}})$. Moreover, for every $P_j \in \mathcal{W}_q$ and every $P_k \in \mathcal{P}$, set $\mathsf{ICSig}(P_j, P_i, P_k, [s^{(\ell)}]_q)$ to the default value.
    - Else, set $[s^{(\ell)}]_q$ to $s_{qi}^{(\ell)}$, where $s_{qi}^{(\ell)}$ was computed as output during $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$. Moreover, if $P_i \in \mathcal{W}_q$, then for every $P_j \in \mathcal{W}_q$ and every $P_k \in \mathcal{P}$, set $\mathsf{ICSig}(P_j, P_i, P_k, [s^{(\ell)}]_q)$ to $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj}^{(\ell)})$, received from P_j.

    Output CORE, the core sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$, shares $\{[s^{(\ell)}]_q\}_{P_\ell \in \mathsf{CORE} \wedge P_i \in S_q}$ and the IC-signatures $\mathsf{ICSig}(P_j, P_i, P_k, [s^{(\ell)}]_q)_{P_\ell \in \mathsf{CORE} \wedge P_j, P_i \in \mathcal{W}_q, P_k \in \mathcal{P}}$.

Fig. 8: The statistically-secure VSS protocol for multiple dealers to generate linearly secret-shared values with IC-signatures

The properties of Π_{MDVSS}, stated in Theorem 7.1 are proved in Appendix E.

Theorem 7.1. *Protocol Π_{MDVSS} achieves the following where each P_ℓ participates with input $s^{(\ell)}$ and where $T_{\mathsf{MDVSS}} = T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 6T_{\mathsf{BA}}$.*

– $\mathcal{Z}_s$**-Correctness&Commitment**: *If the network is synchronous, then except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, at time* T_{MDVSS}*, all honest parties output a common set* $\mathsf{CORE} \subseteq \mathcal{P}$ *such that at least one honest party will be present in* CORE*. Moreover, corresponding to every* $P_\ell \in \mathsf{CORE}$*, there exists some* $s^{\star(\ell)}$*, where* $s^{\star(\ell)} = s^{(\ell)}$ *for an honest* P_ℓ*, such that the values* $\{s^{\star(\ell)}\}_{P_\ell \in \mathsf{CORE}}$ *are linearly secret-shared with IC-signatures.*
– $\mathcal{Z}_a$**-Correctness&Commitment**: *If the network is asynchronous, then except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, almost-surely all honest parties output a common set* $\mathsf{CORE} \subseteq \mathcal{P}$ *eventually such that at least one honest party will be present in* CORE*. Moreover, corresponding to every* $P_\ell \in \mathsf{CORE}$*, there exists some* $s^{\star(\ell)}$*, where* $s^{\star(\ell)} = s^{(\ell)}$ *for an honest* P_ℓ*, such that the values* $\{s^{\star(\ell)}\}_{P_\ell \in \mathsf{CORE}}$ *are eventually linearly secret-shared with IC-signatures.*
– **Privacy**: *Irrespective of the network type, the view of the adversary remains independent of* $s^{(\ell)}$*, corresponding to every honest* $P_\ell \in \mathsf{CORE}$*.*
– **Communication Complexity**: $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n^9 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ *bits are communicated by the honest parties. In addition,* $\mathcal{O}(|\mathcal{Z}_s| + n)$ *instances of* Π_{BA} *are invoked.*

Protocol Π_{MDVSS} with L Values for Each Dealer. In protocol Π_{MDVSS}, each dealer $P_\ell \in \mathcal{P}$ participates with a *single* input. Consider a scenario where *each* P_ℓ participates with L inputs $\overrightarrow{S^{(\ell)}} = (s^{(\ell,1)}, \ldots, s^{(\ell,L)})$, where $L \geq 1$. The goal is to identify a *common* subset of dealers $\mathsf{CORE} \subseteq \mathcal{P}$ which is *guaranteed* to have at least one *honest* dealer, irrespective of the network type. Corresponding to every dealer $P_\ell \in \mathsf{CORE}$, there exist L values, say $\overrightarrow{S^{\star(\ell)}} = (s^{\star(\ell,1)}, \ldots, s^{\star(\ell,L)})$, which will be the same as $\overrightarrow{S^{(\ell)}}$ for an *honest* P_ℓ, where all the values in $\{\overrightarrow{S^{\star(\ell)}}\}_{P_\ell \in \mathsf{CORE}}$ are linearly secret-shared with IC-signatures. To achieve this, we run the protocol Π_{MDVSS} with the following modifications, so that the number of instances of Π_{BA} in the protocol *still* remains to be $\mathcal{O}(|\mathcal{Z}_s| + n)$, which is *independent* of L.

Corresponding to each $s^{(\ell)} \in \overrightarrow{S^{(L)}}$, the dealer P_ℓ will pick $|\mathcal{Z}_s|$ random shares (which sum up to $s^{(\ell)}$) and the shares corresponding to the group $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$ are communicated through an instance of Π_{SVM}; hence $|\mathcal{Z}_s| \cdot L$ instances are invoked by P_ℓ as a Sen. Then, while identifying the set of committed dealers $\mathcal{CD}$, parties vote 1 for P_ℓ in the instance $\Pi_{\mathsf{BA}}^{(\ell)}$ provided the underlying flag variable is set to 1 in *all* the $|Z_s| \cdot L$ instances of Π_{SVM} invoked by P_ℓ. The rest of the steps for identifying $\mathcal{CD}$ remains the same. This way, by executing *only* $\mathcal{O}(n)$ instances of Π_{BA}, we identify the set $\mathcal{CD}$.

Next, the parties exchange IC-signatures on their supposedly common shares for each group, corresponding to *all* the L values shared by each dealer from $\mathcal{CD}$. However, each P_i now broadcasts a *single* $\mathsf{OK}^{(\ell)}(i, j)$ message, corresponding to each $P_j \in S_q$, provided P_i receives IC-signed common share from P_j on the behalf of *all* the L values, shared by P_ℓ. This ensures that, for each P_ℓ, every P_i constructs a *single* consistency graph. Next, each dealer P_ℓ computes and broadcasts the candidate core-sets and broadcast-sets, as and when they are ready. The parties identify the CORE set by running $\mathcal{O}(|\mathcal{Z}_s|+n)$ instances of Π_{BA}.

To avoid repetition, we do not present the formal steps of the modified Π_{MDVSS} protocol. The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot L \cdot n^9 \cdot \log|\mathbb{F}| \cdot |\sigma|)$ bits, apart from $\mathcal{O}(|\mathcal{Z}_s| + n)$ instances of Π_{BA}.

7.2 Protocol for Generating Secret-Shared Random Values

Protocol Π_{Rand} (Fig 9) allows the parties to jointly generate linear secret-sharings of L values with IC-signatures, where $L \geq 1$, which are random for the adversary. For this, the parties invoke an instance of the (modified) Π_{MDVSS} where each dealer $P_\ell \in \mathcal{P}$ participates with a random vector of L values. Let CORE be the set of common dealers identified during the instance of Π_{MDVSS}. Then for $\mathfrak{l} = 1, \ldots, L$, the parties output the sum of $\mathfrak{l}^{th}$ value shared by *all* the dealers in CORE. Since there will be at least one *honest* dealer in CORE whose shared values will be random for the adversary, it follows that the resultant values also remain random for the adversary.

In the rest of the paper, we will refer to the core sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$ obtained during Π_{Rand} as *global core-sets* and denote them by $\mathcal{GW}_{\mathbf{1}}, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$. From now onwards, all the secret-shared values will be generated with respect to these global core-sets.

Protocol $\Pi_{\mathsf{Rand}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, L)$

- **Secret-Sharing Random Values**: Each $P_\ell \in \mathcal{P}$ picks L random values $\overrightarrow{R^{(\ell)}} = (r^{(\ell,1)}, \ldots, r^{(\ell,L)})$ and participates in an instance of Π_{MDVSS} with input $\overrightarrow{R^{(\ell)}}$ and waits for time T_{MDVSS}.
- **Computing Output**: Let $(\mathsf{CORE}, \mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}, \{([r^{\star(\ell,1)}], \ldots, [r^{\star(\ell,L)}])\}_{P_\ell \in \mathsf{CORE}})$ be the output from the instance of Π_{MDVSS}. For $\mathfrak{l} = 1, \ldots, L$, the parties locally compute $[r^{(\mathfrak{l})}] = \sum_{P_\ell \in \mathsf{CORE}} [r^{\star(\ell,\mathfrak{l})}]$ from $\{[r^{\star(\ell,\mathfrak{l})}]\}_{P_\ell \in \mathsf{CORE}}$. The parties then output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \{[r^{(\mathfrak{l})}]\}_{\mathfrak{l}=1,\ldots,L})$, where $\mathcal{GW}_q = \mathcal{W}_q$ for $q = 1, \ldots, |\mathcal{Z}_s|$.

Fig. 9: Protocol for generating linearly secret-shared random values with IC-signatures

Theorem 7.2 follows easily from the above discussion.

Theorem 7.2. *Protocol Π_{Rand} achieves the following where $T_{\mathsf{Rand}} = T_{\mathsf{MDVSS}} = T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 6T_{\mathsf{BA}}$ and $L \geq 1$.*

- $\mathcal{Z}_s$**-correctness**: *If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, at the time T_{Rand}, there exist values $r^{(1)}, \ldots, r^{(L)}$, which are linearly secret-shared with IC-signatures, where the core-sets are $\mathcal{GW}_{\mathbf{1}}, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$.*
- $\mathcal{Z}_a$**-correctness**: *If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, there exist values $r^{(1)}, \ldots, r^{(L)}$, which are almost-surely linearly secret-shared with IC-signatures, where the core-sets are $\mathcal{GW}_{\mathbf{1}}, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$.*
- **Privacy**: *Irrespective of the network type, the view of the adversary remains independent of $r^{(1)}, \ldots, r^{(L)}$.*
- **Communication Complexity**: *The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot L \cdot n^9 \cdot \log|\mathbb{F}| \cdot |\sigma|)$ bits, apart from $\mathcal{O}(|\mathcal{Z}_s| + n)$ instances of Π_{BA}.*

8 Network Agnostic Protocol for Generating Random Multiplication Triples

In this section, we present our network-agnostic triple-generation protocol, which generates random and private multiplication-triples which are linearly secret-shared with IC-signatures. The protocol is based on several sub-protocols which we present next. Throughout this section, we will assume the existence of *global* core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, where $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{GW}_q, \mathcal{Z}_s)$ condition for $q = 1, \ldots, |\mathcal{Z}_s|$. Looking ahead, these core-sets will be generated by first running the protocol Π_{Rand}, using an appropriate value of L, which will be determined across all the sub-protocols which we will be discussing next. All the secret-shared values in the various sub-protocols in the sequel will have $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ as underlying core-sets.

8.1 Verifiably Generating Linear Secret Sharing of a Value with IC-signatures

In protocol Π_{LSh} (Fig 10), there exists a designated dealer $\mathsf{D} \in \mathcal{P}$ with private input s. In addition, there is a random value $r \in \mathbb{F}$, which is linearly secret-shared with IC-signatures, such that the underlying core-sets are $\mathcal{GW}_{\mathbf{1}}, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$ (the value r will *not* be known to D at the beginning of the protocol). The protocol allows the parties to let D verifiably generate a linear secret-sharing of s with IC-signatures, such that the underlying core-sets are $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$, where s remains private for an *honest* D. The verifiability guarantees that even if D is *corrupt*, if any (honest) party computes an output, then there exists *some* value, say $s^\star$, which is linearly secret-shared with IC-signatures, such that the underlying core-sets are $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}$.

The protocol idea is very simple and standard. We first let D reconstruct the value r, which is then used as a *one-time pad* (OTP) by D to make public an OTP-encryption of s. Then, using the linearity property of secret-sharing, the parties locally remove the OTP from the OTP-encryption.

Protocol $\Pi_{\mathsf{LSh}}(\mathsf{D}, s, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [r], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|})$

- **Reconstructing the OTP Towards the Dealer**: The parties in $\mathcal{P}$ invoke an instance $\Pi_{\mathsf{Rec}}([r], \{\mathsf{D}\})$ of Π_{Rec} to let D reconstruct r and wait for time T_{Rec}.
- **Making the OTP-encryption Public**: D, upon computing the output r from the instance of Π_{Rec}, broadcasts $\mathbf{s} = s + r$.
- **Computing the Output**: The parties in $\mathcal{P}$ wait till the local time becomes $T_{\mathsf{Rec}} + T_{\mathsf{BC}}$. Then upon receiving $\mathbf{s}$ from the broadcast of D, the parties in $\mathcal{P}$ locally compute $[\mathbf{s} - r]$ from $[\mathbf{s}]$ and $[r]$. Here $[\mathbf{s}]$ denotes the *default linear secret-sharing* of $\mathbf{s}$ with IC-signatures and core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, where $[\mathbf{s}]_1 = \mathbf{s}$ and $[\mathbf{s}]_2 = \ldots = [\mathbf{s}]_{|\mathcal{Z}_s|} = 0$, and where the parties set $\mathsf{ICSig}(P_j, P_i, P_k, [\mathbf{s}]_q)_{P_j, P_i \in \mathcal{GW}_q, P_k \in \mathcal{P}}$ to the default value. The parties then output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_{\mathbf{s}}|}, [\mathbf{s} - r])$.

Fig. 10: VSS for verifiably generating a linear secret-sharing of a value with IC-signatures with respect to given global core-sets

The properties of the protocol Π_{LSh} stated in Lemma 8.1 are proved in Appendix F.

Lemma 8.1. *Let r be a random value which is linearly secret-shared with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Then protocol Π_{LSh} achieves the following where D participates with the input s.*

- *If D is honest, then the following hold, where $T_{\mathsf{LSh}} = T_{\mathsf{Rec}} + T_{\mathsf{BC}}$.*
  - **$\mathcal{Z}_s$-Correctness***: If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties output $[s]$ at the time T_{LSh}, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
  - **$\mathcal{Z}_a$-Correctness***: If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties eventually output $[s]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
  - **Privacy***: Irrespective of the network type, the view of the adversary remains independent of s.*
- *If D is corrupt then either no honest party computes any output or there exists some value, say $s^\star$, such that the following hold.*
  - **$\mathcal{Z}_s$-Commitment***: If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties output $[s^\star]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, if any honest party computes its output at the time T, then all honest parties will have their respective output by the time $T + \Delta$.*
  - **$\mathcal{Z}_a$-Commitment***: If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties eventually output $[s^\star]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
- **Communication Complexity***: $\mathcal{O}(|\mathcal{Z}_s| \cdot n^3 \cdot \log |\mathbb{F}| + n^4 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits are communicated by the honest parties.*

We end this section with some notations which we use while invoking the protocol Π_{LSh} in the rest of the paper.

Notation 8.1 (Notations for Using Protocol Π_{LSh}) *Let $P_i \in \mathcal{P}$. In the rest of the paper we will say that "P_i invokes an instance of Π_{LSh} with input s" to mean that P_i acts as D and invokes an instance $\Pi_{\mathsf{LSh}}(\mathsf{D}, s, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [r], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|})$ of Π_{LSh}. Here, r will be the corresponding random "pad" for this instance of Π_{LSh}, which will already be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. If there are multiple instances of Π_{LSh} invoked by D, then corresponding to each instance, there will be a random secret-shared pad available to the parties beforehand. The parties will be knowing which secret-shared pad is associated with which instance of Π_{LSh}. This will be ensured by upper-bounding the maximum number of Π_{LSh} instances $\mathsf{L}_{\mathsf{max}}$ invoked across all our protocols. The parties then generate $\mathsf{L}_{\mathsf{max}}$ number of linearly secret-shared random values with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, by running the protocol Π_{Rand} beforehand with $L = \mathsf{L}_{\mathsf{max}}$.*

8.2 Non-Robust Multiplication Protocol

Protocol $\Pi_{\mathsf{BasicMult}}$ (Fig 11) takes input a and b, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets and a *publicly known* subset $\mathcal{GD} \subset \mathcal{P}$, consisting of *only corrupt* parties. The parties output a linear secret-sharing of c with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. If all the parties in $\mathcal{P} \setminus \mathcal{GD}$ behave *honestly*, then $c = a \cdot b$, else $c = a \cdot b + \delta$, where $\delta \neq 0$. Moreover, the adversary does not learn anything additional about a and b in the protocol. The protocol also takes input an *iteration number* iter and all the sets computed in the protocol are tagged with iter. Looking ahead, our *robust* triple-generation protocol will be executed *iteratively*, with each iteration invoking instances of $\Pi_{\mathsf{BasicMult}}$.

A detailed overview of the protocol $\Pi_{\mathsf{BasicMult}}$ has been already presented in Section 1.2.6. The idea is to let each summand $[a]_p \cdot [b]_q$ be linearly secret-shared by exactly one *summand-sharing* party. A secret-sharing of $a \cdot b$ then follows from the secret-sharing of each summand $[a]_p \cdot [b]_q$ owing to the linearity property of the secret-sharing. To deal with the network agnostic condition, the summand-sharing parties are selected in two phases: *first*, we select them *dynamically*, *without* pre-assigning any summand to any designated party. Once there exists a subset of parties from $\mathbb{S}_{\mathcal{Z}_s}$ who have served the role of summand-sharing parties, we go to the *second* phase, where each remaining summand is designated to the left-over parties through some publicly known assignment. Strict timeouts are maintained to ensure that we don't stuck forever during the second phase. Finally, if there are still any remaining summands which are not yet secret-shared, they are publicly reconstructed and the default sharing is taken on their behalf. Throughout, the parties in $\mathcal{GD}$ are not let to secret-share any summand, since they are *already known* to be *corrupt* and at the same time, it is ensured that the shares of the honest parties are *never* publicly reconstructed.

Protocol $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a], [b], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$

– **Initialization**: The parties in $\mathcal{P}$ do the following.
- Initialize the *summand-index-set* of indices of *all* summands:

$$\mathsf{SIS}_{\mathsf{iter}} = \{(p, q)\}_{p,q=1,\ldots,|\mathbb{S}_{\mathcal{Z}_s}|}.$$

- Initialize the *summand-index-set* corresponding to each $P_j \in \mathcal{P} \setminus \mathcal{GD}$:

$$\mathsf{SIS}^{(j)}_{\mathsf{iter}} = \{(p, q)\}_{P_j \in S_p \cap S_q}.$$

- Initialize the *summand-index-set* corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$:

$$\mathsf{SIS}^{(S_q)}_{\mathsf{iter}} = \cup_{P_j \in S_q} \mathsf{SIS}^{(j)}_{\mathsf{iter}}.$$

- Initialize the set of summands-sharing parties:

$$\mathsf{Selected}_{\mathsf{iter}} = \emptyset.$$

- Initialize the hop number:

$$\mathsf{hop} = 1.$$

Phase I: Sharing Summands Through Dynamic Assignment

– While there exists *no* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, where $\mathsf{SIS}_{\mathsf{iter}}^{(S_q)} = \emptyset$, the parties do the following:

- **Sharing Sum of Eligible Summands**: Every $P_i \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$ invokes an instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phI},\mathsf{hop},i)}$ of Π_{LSh} with input $c_{\mathsf{iter}}^{(i)}$, where

$$c_{\mathsf{iter}}^{(i)} = \sum_{(p,q) \in \mathsf{SIS}_{\mathsf{iter}}^{(i)}} [a]_p [b]_q.$$

  Corresponding to every $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, the parties in $\mathcal{P}$ participate in the instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phI},\mathsf{hop},j)}$, if invoked by P_j.

- **Selecting Summand-Sharing Party for the Hop Through ACS**: The parties in $\mathcal{P}$ wait for time T_{LSh} and then do the following.
  – For $j = 1, \ldots, n$, participate in an instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$ of Π_{BA} corresponding to $P_j \in \mathcal{P}$ with input 1 if *all* the following hold:
    – $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$;
    – An output $[c_{\mathsf{iter}}^{(j)}]$ is computed during the instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phI},\mathsf{hop},j)}$.
  – Upon computing an output 1 during the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$ corresponding to some $P_j \in \mathcal{P}$, participate with input 0 in the instances $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},k)}$ corresponding to parties $P_k \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, for which no input has been provided yet.
  – Upon computing outputs during the instances $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},i)}$ corresponding to each $P_i \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, let P_j be the least-indexed party, such that the output 1 is computed during the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$. Then update the following.
    – $\mathsf{Selected}_{\mathsf{iter}} = \mathsf{Selected}_{\mathsf{iter}} \cup \{P_j\}$.
    – $\mathsf{SIS}_{\mathsf{iter}} = \mathsf{SIS}_{\mathsf{iter}} \setminus \mathsf{SIS}_{\mathsf{iter}}^{(j)}$.
    – $\forall P_k \in \mathcal{P} \setminus \{\mathcal{GD} \cup \mathsf{Selected}_{\mathsf{iter}}\}$: $\mathsf{SIS}_{\mathsf{iter}}^{(k)} = \mathsf{SIS}_{\mathsf{iter}}^{(k)} \setminus \mathsf{SIS}_{\mathsf{iter}}^{(j)}$.
    – For each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, $\mathsf{SIS}_{\mathsf{iter}}^{(S_q)} = \mathsf{SIS}_{\mathsf{iter}}^{(S_q)} \setminus \mathsf{SIS}_{\mathsf{iter}}^{(j)}$.
    – Set $\mathsf{hop} = \mathsf{hop} + 1$.

Phase II: Sharing Remaining Summands Through Static Assignment

- **Re-assigning the Summand-Index-Set of Each Party**: Corresponding to each $P_j \in \mathcal{P} \setminus \mathsf{Selected}_{\mathsf{iter}}$, the parties in $\mathcal{P}$ set $\mathsf{SIS}_{\mathsf{iter}}^{(j)}$ as

$$\mathsf{SIS}_{\mathsf{iter}}^{(j)} = \mathsf{SIS}_{\mathsf{iter}} \cap \{(p,q)\}_{P_j = \min(S_p \cap S_q)},$$

  where $\min(S_p \cap S_q)$ denotes the minimum indexed party in $(S_p \cap S_q)$.

- **Sharing Sum of Assigned Summands**: Every party $P_i \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$ invokes an instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phII},i)}$ of Π_{LSh} with input $c_{\mathsf{iter}}^{(i)}$, where

$$c_{\mathsf{iter}}^{(i)} = \sum_{(p,q) \in \mathsf{SIS}_{\mathsf{iter}}^{(i)}} [a]_p [b]_q.$$

  Corresponding to every $P_j \in \mathcal{P} \setminus (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, the parties in $\mathcal{P}$ participate in the instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phII},j)}$, if invoked by P_j.

- **Agreeing on the Summand-Sharing parties of the Second Phase**: The parties in $\mathcal{P}$ wait for T_{LSh} time after the beginning of the second phase. Then for

each $P_j \in \mathcal{P}$, participate in an instance $\Pi_{\mathsf{BA}}^{(\mathsf{phII},j)}$ of Π_{BA} with input 1, if *all* the following hold, otherwise participate with input 0.
– $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$;
– An output $[c_{\mathsf{iter}}^{(j)}]$ is computed during the instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phII},j)}$.

- **Updating the Sets for the Second Phase**: Corresponding to each $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, such that 1 is computed as the output during $\Pi_{\mathsf{BA}}^{(\mathsf{phII},j)}$, update
  – $\mathsf{SIS}_{\mathsf{iter}} = \mathsf{SIS}_{\mathsf{iter}} \setminus \mathsf{SIS}_{\mathsf{iter}}^{(j)}$;
  – $\mathsf{Selected}_{\mathsf{iter}} = \mathsf{Selected}_{\mathsf{iter}} \cup \{P_j\}$.

Phase III: Reconstructing the Remaining Summands

- **Reconstructing the Remaining Summands and Taking the Default Sharing**: The parties in $\mathcal{P}$ do the following.
  – Corresponding to each $[a]_p$ such that $(p, \star) \in \mathsf{SIS}_{\mathsf{iter}}$, participate in the instance $\Pi_{\mathsf{RecShare}}([a], S_p, \mathcal{P})$ of Π_{RecShare} to publicly reconstruct $[a]_p$
  – Corresponding to each $[b]_q$ such that $(\star, q) \in \mathsf{SIS}_{\mathsf{iter}}$, participate in the instance $\Pi_{\mathsf{RecShare}}([b], S_q, \mathcal{P})$ of Π_{RecShare} to publicly reconstruct $[b]_q$.
  – Corresponding to every $P_j \in \mathcal{P} \setminus \mathsf{Selected}_{\mathsf{iter}}$, take the *default linear secret-sharing* of the public input $c_{\mathsf{iter}}^{(j)}$ with IC-signatures and core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, where[a]

$$c_{\mathsf{iter}}^{(j)} = \sum_{(p,q) \in \mathsf{SIS}_{\mathsf{iter}}^{(j)}} [a]_p [b]_q.$$

- **Output Computation**: The parties output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [c_{\mathsf{iter}}^{(1)}], \ldots, [c_{\mathsf{iter}}^{(n)}], [c_{\mathsf{iter}}])$, where $c_{\mathsf{iter}} \stackrel{def}{=} c_{\mathsf{iter}}^{(1)} + \ldots + c_{\mathsf{iter}}^{(n)}$.

[a] The default linear secret-sharing of $c_{\mathsf{iter}}^{(j)}$ is computed in a similar way as done in the protocol Π_{Rand} for $\mathbf{s}$ (see Fig 9).

Fig. 11: Network-agnostic non-robust multiplication protocol

The properties of the protocol $\Pi_{\mathsf{BasicMult}}$ are claimed in the following lemmas, which are proved in Appendix F.

Lemma 8.2. *During any instance* $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a], [b], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$ *of* $\Pi_{\mathsf{BasicMult}}$*, if* $P_j \in \mathsf{Selected}_{\mathsf{iter}}$ *then* $P_j \notin \mathcal{GD}$*, irrespective of the network type.*

Lemma 8.3. *Suppose that no honest party is present in* $\mathcal{GD}$*. If the honest parties start participating during hop number* hop *of Phase I of* $\Pi_{\mathsf{BasicMult}}$ *with iteration number* iter*, then except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, the hop takes* $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$ *time to complete in a synchronous network, or almost-surely completes eventually in an asynchronous network.*

Lemma 8.4. *If no honest party is present in* $\mathcal{GD}$*, then in protocol* $\Pi_{\mathsf{BasicMult}}$*, except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, all honest parties compute some output by the time* $T_{\mathsf{BasicMult}} = (2n+1) \cdot T_{\mathsf{BA}} + (n+1) \cdot T_{\mathsf{LSh}} + T_{\mathsf{Rec}}$ *in a synchronous network, or almost-surely, eventually in an asynchronous network.*

Lemma 8.5. *If no honest party is present in $\mathcal{GD}$, then the view of the adversary remains independent of a and b throughout the protocol, irrespective of the network type.*

Lemma 8.6. *If no honest party is present in $\mathcal{GD}$ and if all parties in $\mathcal{P} \setminus \mathcal{GD}$ behave honestly, then in protocol $\Pi_{\mathsf{BasicMult}}$, the honest parties output a linear secret-sharing of $a \cdot b$ with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, irrespective of the network type.*

Lemma 8.7. *Protocol $\Pi_{\mathsf{BasicMult}}$ incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot \log |\mathbb{F}| + n^6 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^2)$ calls to Π_{BA}.*

As a corollary of Lemma 8.7, we can derive the following corollary, which determines the *maximum* number of instances of Π_{LSh} which are invoked during an instance of $\Pi_{\mathsf{BasicMult}}$. Looking ahead, this will be useful to later calculate the maximum number of instances of Π_{LSh} which need to be invoked as part of our final multiplication protocol. This will be further useful to determine the number of linearly secret-shared values with IC-signatures and core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, which need to be generated through the protocol Π_{Rand} beforehand.

Corollary 8.1. *During any instance of $\Pi_{\mathsf{BasicMult}}$, there can be at most $n^2 + n$ instances of Π_{LSh} invoked.*

Protocol $\Pi_{\mathsf{BasicMult}}$ for L Pairs of Inputs. Protocol $\Pi_{\mathsf{BasicMult}}$ can be easily generalized, if there are L pairs of inputs $\{(a^\ell, b^\ell)\}_{\ell=1,\ldots,L}$, all of which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. However, with a slight modification during Phase I and Phase II, we can ensure that the number of instances of Π_{BA} remain *only* $\mathcal{O}(n^2)$, which is *independent* of L. Consider Phase I. During hop number hop, every party $P_i \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$ invokes L instances of Π_{LSh} to linearly secret-share L candidate summand-sums. Now while selecting the summand-sharing party through ACS for this hop, the parties vote for a candidate $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, provided an output is computed in *all* the L instances of Π_{LSh} invoked by P_j. Consequently, the number of instances of Π_{BA} during Phase I will be $\mathcal{O}(n^2)$. Similarly during Phase II, each party *outside* $(\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$ invokes L instances of Π_{LSh} to linearly secret-share L candidate re-assigned summand-sums. And then the parties vote for a candidate P_j as a summand-sharing party, if an output is computed in *all* the L instances of Π_{LSh} invoked by P_j. Finally, during Phase III, the default linear secret-sharing with IC-signatures is taken for the sum of all the summands, which are not yet secret-shared by any party, by making public all these summands. The resultant protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^3 \cdot n^4 \cdot L \cdot \log |\mathbb{F}| + |\mathcal{Z}_s| \cdot n^5 \cdot L \cdot \log |\mathbb{F}| + n^6 \cdot L \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^2)$ calls to Π_{BA}. We also note that there will be at most $n^2 \cdot L + n \cdot L$ instances of Π_{LSh} invoked in the generalized protocol. To avoid repetition, we do not present the steps of the generalized protocol here.

8.3 Network Agnostic Random Triple Generation with Cheater Identification

The network-agnostic protocol $\Pi_{\mathsf{RandMultCI}}$ (Fig 12) takes an iteration number iter and a publicly known subset of parties $\mathcal{GD}$, who are guaranteed to be *corrupt*. If *all* the parties in $\mathcal{P} \setminus \mathcal{GD}$ behave *honestly*, then the protocol outputs a random linearly secret-shared multiplication-triple with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core sets. Otherwise, with a high probability, the honest parties identify a *new corrupt* party, which is added to $\mathcal{GD}$.

Protocol $\Pi_{\mathsf{RandMultCI}}$ is based on [40] and consists of two stages: during the *first* stage, the parties jointly generate a pair of random values, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core sets. During the second stage, the parties run an instance of $\Pi_{\mathsf{BasicMult}}$ to compute the product of the pair of secret-shared random values from the first stage. To check whether any cheating has occurred during the instance of $\Pi_{\mathsf{BasicMult}}$, the parties then run a probabilistic test, namely the "sacrificing trick" [27], for which the parties need *additional* secret-shared random values, which are generated during the first stage itself.

Protocol $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$

- **Generating Linear Secret Sharing of Random Values with IC-signatures**: Each $P_i \in \mathcal{P}$ does the following.
  - Invoke instances of Π_{LSh} with randomly chosen inputs $a^{(i)}_{\mathsf{iter}}, b^{(i)}_{\mathsf{iter}}, b'^{(i)}_{\mathsf{iter}}, r^{(i)}_{\mathsf{iter}} \in \mathbb{F}$.
  - Corresponding to every $P_j \in \mathcal{P}$, participate in the instances of Π_{LSh} invoked by P_j (if any) and wait for time T_{LSh}. Initialize a set $\mathcal{C}_i = \emptyset$ after local time T_{LSh} and include P_j in $\mathcal{C}_i$, if any output is computed in *all* the instances of Π_{LSh} invoked by P_j.
  - Corresponding to every $P_j \in \mathcal{P}$, participate in an instance of $\Pi^{(j)}_{\mathsf{BA}}$ of Π_{BA} with input 1, if $P_j \in \mathcal{C}_i$.
  - Once 1 has been computed as the output from instances of Π_{BA} corresponding to a set of parties in $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, participate with input 0 in all the Π_{BA} instances $\Pi^{(j)}_{\mathsf{BA}}$, such that $P_j \notin \mathcal{C}_i$.
  - Once a binary output is computed in all the instances of Π_{BA} corresponding to the parties in $\mathcal{P}$, compute $\mathcal{CS}$, which is the set of parties $P_j \in \mathcal{P}$, such that 1 is computed as the output in the instance $\Pi^{(j)}_{\mathsf{BA}}$.

  Once $\mathcal{CS}$ is computed, the parties in $\mathcal{P}$ locally compute $[a_{\mathsf{iter}}]$, $[b_{\mathsf{iter}}]$, $[b'_{\mathsf{iter}}]$ and $[r_{\mathsf{iter}}]$ from $\{[a^{(j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}}, \{[b^{(j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}}, \{[b'^{(j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}}$ and $\{[r^{(j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}}$ respectively as follows:

$$[a_{\mathsf{iter}}] = \sum_{P_j \in \mathcal{CS}} [a^{(j)}_{\mathsf{iter}}],\ [b_{\mathsf{iter}}] = \sum_{P_j \in \mathcal{CS}} [b^{(j)}_{\mathsf{iter}}],\ [b'_{\mathsf{iter}}] = \sum_{P_j \in \mathcal{CS}} [b'^{(j)}_{\mathsf{iter}}],\ [r_{\mathsf{iter}}] = \sum_{P_j \in \mathcal{CS}} [r^{(j)}_{\mathsf{iter}}].$$

- **Computing Secret-Shared Products**: The parties in $\mathcal{P}$ do the following.
  - Participate in instances $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$ and $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a], [b'_{\mathsf{iter}}], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$ of $\Pi_{\mathsf{BasicMult}}$ to compute the outputs $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [c^{(1)}_{\mathsf{iter}}], \ldots, [c^{(n)}_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ and $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [c'^{(1)}_{\mathsf{iter}}], \ldots, [c'^{(n)}_{\mathsf{iter}}], [c'_{\mathsf{iter}}])$ respectively. Let $\mathsf{Selected}_{\mathsf{iter},c}$ and $\mathsf{Selected}_{\mathsf{iter},c'}$ be the summand-sharing parties for the two in-

stances respectively. Moreover, for each $P_j \in \mathsf{Selected}_{\mathsf{iter},c}$, let $\mathsf{SIS}^{(j)}_{\mathsf{iter},c}$ be the set of ordered pairs of indices corresponding to the summands whose sum has been shared by P_j during the instance $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$. And similarly, for each $P_j \in \mathsf{Selected}_{\mathsf{iter},c'}$, let $\mathsf{SIS}^{(j)}_{\mathsf{iter},c'}$ be the set of ordered pairs of indices corresponding to the summands whose sum has been shared by P_j during the instance $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a], [b'_{\mathsf{iter}}], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$.

- **Error Detection in the Instances of** $\Pi_{\mathsf{BasicMult}}$: The parties in $\mathcal{P}$ do the following.
  - Upon computing outputs from the instances of $\Pi_{\mathsf{BasicMult}}$, participate in an instance $\Pi_{\mathsf{Rec}}([r_{\mathsf{iter}}], \mathcal{P})$ of Π_{Rec} to publicly reconstruct r_{iter}.
  - Locally compute $[e_{\mathsf{iter}}] \stackrel{def}{=} r_{\mathsf{iter}}[b_{\mathsf{iter}}] + [b'_{\mathsf{iter}}]$ from $[b_{\mathsf{iter}}]$ and $[b'_{\mathsf{iter}}]$. Participate in an instance $\Pi_{\mathsf{Rec}}([e_{\mathsf{iter}}], \mathcal{P})$ of Π_{Rec} to publicly reconstruct e_{iter}.
  - Locally compute $[d_{\mathsf{iter}}] \stackrel{def}{=} e_{\mathsf{iter}}[a_{\mathsf{iter}}] - r_{\mathsf{iter}}[c_{\mathsf{iter}}] - [c'_{\mathsf{iter}}]$ from $[a_{\mathsf{iter}}]$, $[c_{\mathsf{iter}}]$ and $[c'_{\mathsf{iter}}]$. Participate in an instance $\Pi_{\mathsf{Rec}}([d_{\mathsf{iter}}], \mathcal{P})$ of Π_{Rec} to publicly reconstruct d_{iter}.
  - **Output Computation in Case of Success:** If $d_{\mathsf{iter}} = 0$, then set the boolean variable $\mathsf{flag}_{\mathsf{iter}} = 0$ and output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$.
  - **Cheater Identification in Case of Failure:** If $d_{\mathsf{iter}} \neq 0$, then set the boolean variable $\mathsf{flag}_{\mathsf{iter}} = 1$ and proceed as follows.
    - For each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, participate in instances $\Pi_{\mathsf{RecShare}}([a_{\mathsf{iter}}], S_q, \mathcal{P})$, $\Pi_{\mathsf{RecShare}}([b_{\mathsf{iter}}], S_q, \mathcal{P})$ and $\Pi_{\mathsf{RecShare}}([b'_{\mathsf{iter}}], S_q, \mathcal{P})$ of Π_{RecShare} to publicly reconstruct the shares $\{[a_{\mathsf{iter}}]_q, [b_{\mathsf{iter}}]_q, [b'_{\mathsf{iter}}]_q\}_{S_q \in \mathbb{S}_{\mathcal{Z}_s}}$. In addition, for $i = 1, \ldots, n$, participate in instances $\Pi_{\mathsf{Rec}}(c^{(i)}_{\mathsf{iter}}, \mathcal{P})$ and $\Pi_{\mathsf{Rec}}(c'^{(i)}_{\mathsf{iter}}, \mathcal{P})$ to publicly reconstruct $c^{(i)}_{\mathsf{iter}}$ and $c'^{(i)}_{\mathsf{iter}}$.
    - Set $\mathcal{GD} = \mathcal{GD} \cup \{P_j\}$, if $P_j \in \mathsf{Selected}_{\mathsf{iter},c} \cup \mathsf{Selected}_{\mathsf{iter},c'}$ and the following holds for P_j:

$$r_{\mathsf{iter}} \cdot c^{(j)}_{\mathsf{iter}} + c'^{(j)}_{\mathsf{iter}} \neq r_{\mathsf{iter}} \cdot \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q + \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

Fig. 12: Network-agnostic protocol for generating secret-shared random multiplication-triple with cheater identification.

The properties of the protocol $\Pi_{\mathsf{RandMultCI}}$ are claimed in the following lemmas, which are proved in Appendix F.

Lemma 8.8. *In protocol* $\Pi_{\mathsf{RandMultCI}}$*, the following hold.*
- **Synchronous Network***: Except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, honest parties will have linearly secret-shared* $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ *and* r_{iter} *with IC-signatures, with* $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ *being the underlying core-sets, by the time* $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$*. Moreover, adversary's view is independent of* $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ *and* r_{iter}*.*
- **Asynchronous Network***: Except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, almost-surely, honest parties will eventually have linearly secret-shared* $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ *and* r_{iter} *with IC-signatures, with* $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ *being the underlying core-sets. Moreover, adversary's view is independent of* $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ *and* r_{iter}*.*

Lemma 8.9. *Consider an arbitrary* iter*, such that all honest parties participate in the instance* $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$*, where no* *honest party is present in $\mathcal{GD}$. Then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties reconstruct a (common) value d_{iter} and set $\mathsf{flag}_{\mathsf{iter}}$ to a common Boolean value, at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network, or eventually in an asynchronous network.*

Lemma 8.10. *Consider an arbitrary iter, such that all honest parties participate in the instance $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$, where no honest party is present in $\mathcal{GD}$. If no party in $\mathcal{P} \setminus \mathcal{GD}$ behaves maliciously, then $d_{\mathsf{iter}} = 0$ and the honest parties output $([a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network, where $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ and where $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ are the underlying core-sets*

Lemma 8.11. *Consider an arbitrary iter, such that all honest parties participate in the instance $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$, where no honest party is present in $\mathcal{GD}$. If $d_{\mathsf{iter}} \neq 0$, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties update $\mathcal{GD}$ by adding a new maliciously-corrupt party in $\mathcal{GD}$, either at the time $T_{\mathsf{RandMultCI}} = T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 4T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network.*

Lemma 8.12. *Consider an arbitrary iter, such that all honest parties participate in the instance $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$, where no honest party is present in $\mathcal{GD}$. If $d_{\mathsf{iter}} = 0$, then the honest parties output linearly secret-shared $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$ with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network where, except with probability $\frac{1}{|\mathbb{F}|}$, the condition $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ holds. Moreover, the view of Adv will be independent of $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$.*

Lemma 8.13. *Protocol $\Pi_{\mathsf{RandMultCI}}$ incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot \log |\mathbb{F}| + n^6 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^2)$ calls to Π_{BA}.*

Protocol $\Pi_{\mathsf{RandMultCI}}$ for L Triples. Protocol $\Pi_{\mathsf{RandMultCI}}$ can be easily generalized to generate L triples with cheater identification, such that the number of instances of Π_{BA} is *independent* of L. To begin with, every party P_i now picks $3L+1$ random values $(r^{(i)}_{\mathsf{iter}}, \{a^{(\ell,i)}_{\mathsf{iter}}, b^{(\ell,i)}_{\mathsf{iter}}, {b'}^{(\ell,i)}_{\mathsf{iter}}\}_{\ell=1,\ldots,L})$ and invokes $3L+1$ instances of Π_{LSh} to generate linear secret-sharing of these values with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Next, to determine $\mathcal{CS}$, the parties invoke *only* n instances of Π_{BA}, where the j^{th} instance is used to decide whether P_j should be included in $\mathcal{CS}$, the criteria being whether some output is computed in *all* the $3L+1$ instances of Π_{LSh} invoked by P_j. Once $\mathcal{CS}$ is decided, the parties locally compute $\{[a^{(\ell)}_{\mathsf{iter}}]\}_{\ell=1,\ldots,L}$, $\{[b^{(\ell)}_{\mathsf{iter}}]\}_{\ell=1,\ldots,L}$, $\{[{b'}^{(\ell)}_{\mathsf{iter}}]\}_{\ell=1,\ldots,L}$ and $[r_{\mathsf{iter}}]$ from $\{[a^{(\ell,j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}, \ell=1,\ldots,L}$, $\{[b^{(\ell,j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}, \ell=1,\ldots,L}$, $\{[{b'}^{(\ell,j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}, \ell=1,\ldots,L}$ and $\{[r^{(j)}_{\mathsf{iter}}]\}_{P_j \in \mathcal{CS}}$ respectively. The parties then invoke two instances of (generalized) $\Pi_{\mathsf{BasicMult}}$ protocol with L pairs of secret-shared inputs to compute $\{[c^{(\ell)}_{\mathsf{iter}}]\}_{\ell=1,\ldots,L}$ and $\{[{c'}^{(\ell)}_{\mathsf{iter}}]\}_{\ell=1,\ldots,L}$ respectively. Note that this requires *only*

$\mathcal{O}(n^2)$ instances of Π_{BA}, apart from $n^2 \cdot L + n \cdot L$ instances of Π_{LSh}. The rest of the protocol steps are then generalized to deal with L inputs. The resultant protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot L \cdot \log |\mathbb{F}| + n^6 \cdot L \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^2)$ calls to Π_{BA}. We also note that there will be at most $n^2 \cdot L + 4n \cdot L + n$ instances of Π_{LSh} invoked overall in the generalized protocol. To avoid repetition, we do not present the steps of the generalized protocol here.

8.4 The Multiplication-Triple Generation Protocol

Protocol Π_{TripGen} for generating a single secret-shared multiplication-triple is presented in Fig 13. The idea of the protocol is very simple and based on [40]. The parties iteratively run instances of $\Pi_{\mathsf{RandMultCI}}$, till they hit upon an instance when *no* cheating is detected. Corresponding to each "failed" instance of $\Pi_{\mathsf{RandMultCI}}$, the parties keep updating the set $\mathcal{GD}$. Since after each failed instance the set $\mathcal{GD}$ is updated with one *new* corrupt party, there will be at most $(t+1)$ iterations, where t is the cardinality of the largest-sized subset in $\mathcal{Z}_s$.

Protocol $\Pi_{\mathsf{TripGen}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|})$

- **Initialization**: The parties in $\mathcal{P}$ initialize $\mathcal{GD} = \emptyset$ and $\mathsf{iter} = 1$.
- **Triple Generation with Cheater Identification**: The parties in $\mathcal{P}$ participate in an instance $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$ of $\Pi_{\mathsf{RandMultCI}}$ and wait for its completion. Upon computing output from the instance, the parties proceed as follows.
  - **Positive Output**: If the Boolean variable $\mathsf{flag}_{\mathsf{iter}}$ is set to 0 during the instance of $\Pi_{\mathsf{RandMultCI}}$, then output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$, computed during the instance of $\Pi_{\mathsf{RandMultCI}}$.
  - **Negative Output**: Else set $\mathsf{iter} = \mathsf{iter} + 1$ and go to the step labelled **Triple Generation with Cheater Identification**.

Fig. 13: Network-agnostic protocol to generate a linear secret sharing with IC-signature of a single random multiplication-triple.

The properties of the protocol Π_{TripGen} are claimed in the following lemmas, which are proved in Appendix F.

Lemma 8.14. *Let t be the size of the largest set in $\mathcal{Z}_s$. Then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties compute an output during Π_{TripGen}, by the time $T_{\mathsf{TripGen}} = (t+1) \cdot T_{\mathsf{RandMultCI}}$ in a synchronous network, or almost-surely, eventually in an asynchronous network, where $T_{\mathsf{RandMultCI}} = T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 4T_{\mathsf{Rec}}$.*

Lemma 8.15. *If the honest parties output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ during the protocol Π_{TripGen}, then $a_{\mathsf{iter}}, b_{\mathsf{iter}}$ and c_{iter} are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, $c_{\mathsf{iter}} = a_{\mathsf{iter}} b_{\mathsf{iter}}$ holds, except with probability $\frac{1}{|\mathbb{F}|}$. Furthermore, the view of the adversary remains independent of $a_{\mathsf{iter}}, b_{\mathsf{iter}}$ and c_{iter}.*

Lemma 8.16. *Protocol Π_{TripGen} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^6 \cdot \log |\mathbb{F}| + n^7 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^3)$ calls to Π_{BA}.*

Protocol Π_{TripGen} for Generating L Multiplication-Triples. To generate L multiplication-triples, the parties now need to invoke an instance of the generalized (modified) $\Pi_{\mathsf{RandMultCI}}$ protocol in each iteration, which generates L triples with cheater identification. The rest of the protocol steps remain the same. To avoid repetition, we do not present the formal details here. The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^6 \cdot L \cdot \log |\mathbb{F}| + n^7 \cdot L \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^3)$ calls to Π_{BA}.

On the Maximum Number of Calls of Π_{LSh} in Π_{TripGen}. As discussed in the previous section, each instance of $\Pi_{\mathsf{RandMultCI}}$ for L triples requires at most $n^2 \cdot L + 4n \cdot L + n$ instances of Π_{LSh}. Now as there can be up to $t + 1 \approx n$ such instances of $\Pi_{\mathsf{RandMultCI}}$ in the protocol Π_{TripGen}, it follows that at most $n^3 \cdot L + 4n^2 \cdot L + n^2$ instances of Π_{LSh} are invoked in the protocol Π_{TripGen} for generating L multiplication-triples.

9 Network Agnostic Circuit-EvaluationProtocol

The network-agnostic circuit-evaluation protocol Π_{cktEval} is presented in Fig 14. The idea behind the protocol is to perform shared circuit-evaluation, where each value remains linearly secret-shared with IC-signatures and common core-sets Once the function-output is secret-shared, it is publicly reconstructed. To achieve this goal, the parties first secret-share their respective inputs for the function f through instances of Π_{LSh}. The parties then 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 linearly secret-shared with IC-signatures. If the network is *synchronous* then it will be ensured that *all honest* parties are present in $\mathcal{CS}$ and hence the inputs of *all honest* parties are considered for the circuit-evaluation. The linearity of secret-sharing ensures that the linear gates in ckt are evaluated non-interactively, while Beaver's trick is deployed for evaluating multiplication gates in ckt. For the latter, the parties need to have c_M number of random multiplication-triples apriori, which are linearly secret-shared with IC-signatures. This is achieved by apriori calling the protocol Π_{TripGen} with $L = c_M$, which in turn will require *at most* $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2$ number of instances of Π_{LSh}. As the *total* number of instances of Π_{LSh} across Π_{TripGen} and the input-phase is at most $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$, the parties first invoke an instance of Π_{Rand} by setting $L = n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$, to generate these many linearly secret-shared random values with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. This will ensure that all the values during the circuit-evaluation are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

Notice that if the network is *asynchronous* then different parties may be in the different phases of the protocol. And consequently, a party upon reconstructing the function output *cannot* afford to immediately terminate, as its presence may be required in the other parts of the protocol. Hence there is also a *termination phase*, which is executed concurrently, where the parties check if it is "safe" to terminate the protocol.

Protocol $\Pi_{\mathsf{cktEval}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathsf{ckt}, c_M)$

Pre-Processing Phase

- **Generating Linearly Secret-Shared Random Values with IC-signatures**: The parties invoke an instance $\Pi_{\mathsf{Rand}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, L)$ where $L = n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$ and compute output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \{[r^{(\mathfrak{l})}]\}_{\mathfrak{l}=1,\ldots,L})$
- **Generating Linearly Secret-Shared Random Multiplication-Triples with IC-signatures**: Upon computing an output in the instance of Π_{Rand}, the parties invoke an instance $\Pi_{\mathsf{TripGen}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|})$ with $L = c_M$ and compute output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \{[a^{(\ell)}], [b^{(\ell)}], [c^{(\ell)}]\}_{\ell=1,\ldots,c_M})$. During the instance of Π_{TripGen}, the secret-shared values $\{[r^{(\mathfrak{l})}]\}_{\mathfrak{l}=1,\ldots,n^3 \cdot c_M + 4n^2 \cdot c_M + n^2}$ are used as the corresponding pads in the underlying instances of Π_{LSh}, invoked as part of Π_{TripGen}.

Input Phase

Upon computing an output during the instance of Π_{TripGen}, each $P_i \in \mathcal{P}$ does the following.

- On having the input $x^{(i)}$, invoke an instance of Π_{LSh} with input $x^{(i)}$.[a]
- Corresponding to every $P_j \in \mathcal{P}$, participate in the instance of Π_{LSh} invoked by P_j and wait for local time T_{LSh} after starting the input phase. Then initialize a set $\mathcal{C}_i = \emptyset$ and include P_j in $\mathcal{C}_i$, if any output is computed in the instance of Π_{LSh} invoked by P_j.
- Corresponding to every $P_j \in \mathcal{P}$, 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 computed as the output from instances of Π_{BA} corresponding to a set of parties in $\mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, participate with input 0 in all the Π_{BA} instances $\Pi_{\mathsf{BA}}^{(j)}$, such that $P_j \notin \mathcal{C}_i$.
- Once a binary output is computed in all the instances of Π_{BA} corresponding to the parties in $\mathcal{P}$, compute $\mathcal{CS}$, which is the set of parties $P_j \in \mathcal{P}$, such that 1 is computed as the output in the instance $\Pi_{\mathsf{BA}}^{(j)}$.

Once $\mathcal{CS}$ is computed, corresponding to every $P_j \notin \mathcal{CS}$, the parties set $x^{(j)} = 0$ and take the *default* linear secret-sharing of 0 with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

Circuit-Evaluation Phase

- Evaluate each gate g in the circuit according to the topological ordering as follows, depending upon the type.
  - **Addition Gate**: If g is an addition gate with inputs x, y and output z, then the parties in $\mathcal{P}$ locally compute $[z] = [x + y]$ from $[x]$ and $[y]$.
  - **Multiplication Gate**: If g is the ℓ^{th} multiplication gate with inputs x, y and output z, where $\ell \in \{1, \ldots, M\}$, then the parties in $\mathcal{P}$ do the following:
    - Locally compute $[d^{(\ell)}] = [x - a^{(\ell)}]$ from $[x]$ and $[a^{(\ell)}]$ and $[e^{(\ell)}] = [y - b^{(\ell)}]$ from $[y]$ and $[b^{(\ell)}]$.
    - Participate in instances $\Pi_{\mathsf{Rec}}([d^{(\ell)}], \mathcal{P})$ and $\Pi_{\mathsf{Rec}}([e^{(\ell)}], \mathcal{P})$ of Π_{Rec} to publicly reconstruct $d^{(\ell)}$ and $e^{(\ell)}$, where $d^{(\ell)} \stackrel{def}{=} x - a^{(\ell)}$ and $e^{(\ell)} \stackrel{def}{=} y - b^{(\ell)}$.
    - Upon reconstructing $d^{(\ell)}$ and $e^{(\ell)}$, take the default linear secret-sharing of $d^{(\ell)} \cdot e^{(\ell)}$ with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying

core-sets. Then locally compute $[z] = [d^{(\ell)} \cdot e^{(\ell)}] + d^{(\ell)} \cdot [b^{(\ell)}] + e^{(\ell)} \cdot [a^{(\ell)}] + [c^{(\ell)}]$ from $[d^{(\ell)} \cdot e^{(\ell)}], [a^{(\ell)}], [b^{(\ell)}]$ and $[c^{(\ell)}]$.

– **Output Gate**: If g is the output gate with output y, then participate in an instance $\Pi_{\mathsf{Rec}}([y], \mathcal{P})$ of Π_{Rec} to publicly reconstruct y.

Termination Phase

Every $P_i \in \mathcal{P}$ concurrently executes the following steps during the protocol:

– Upon computing the circuit-output y, send the message (ready, P_i, y) to every party in $\mathcal{P}$.
– Upon receiving the message (ready, P_j, y) from a set of parties $\mathcal{A}$ such that $\mathcal{Z}_s$ satisfies $\mathbb{Q}^{(1)}(\mathcal{A}, \mathcal{Z}_s)$ condition, send (ready, P_i, y) to every party in $\mathcal{P}$, provided no $(\mathsf{ready}, P_i, \star)$ message has been sent yet.
– Upon receiving the message (ready, P_j, y) from a set of parties $\mathcal{W}$ such that $\mathcal{P} \setminus \mathcal{W} \in \mathcal{Z}_s$, output y and terminate.

[a] The secret-shared random value $r^{(n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + j)}$ serves as the pad for the instance of Π_{LSh} invoked by P_j in this phase.

Fig. 14: Network agnostic secure circuit-evaluation protocol.

The properties of the protocol Π_{cktEval} stated in Theorem 9.1, are proved in Appendix G.

Theorem 9.1. *Let $\mathcal{Z}_s$ and $\mathcal{Z}_a$ be monotone adversary structures where $\mathcal{Z}_a \subset \mathcal{Z}_s$, the set $\mathcal{Z}_s$ satisfy the condition $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$, the set $\mathcal{Z}_a$ satisfy the condition $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ and $\mathcal{Z}_s, \mathcal{Z}_a$ together satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| \geq n^5 \cdot 2^{\mathsf{ssec}}$ where ssec is the statistical security parameter. Moreover, let $y = f(x^{(1)}, \ldots, x^{(n)})$ be a publicly known function over $\mathbb{F}$ represented by an arithmetic circuit ckt over $\mathbb{F}$, where each $P_i \in \mathcal{P}$ has the input $x^{(i)}$. Furthermore, let c_M and D_M be the number of multiplication gates and the multiplicative depth of ckt respectively. Then given an unconditionally-secure PKI, protocol Π_{cktEval} achieves the following, where every honest P_j participates with the input $x^{(j)}$ and where t denotes the cardinality of the maximum sized subset in $\mathcal{Z}_s$.*

– **Synchronous Network**: *Except with probability $2^{-\mathsf{ssec}}$, all honest parties output $y = f(x^{(1)}, \ldots, x^{(n)})$ at the time $[(3n+5)t^2 + (74n+140)t + 69n + D_M + 438] \cdot \Delta$, where $x^{(j)} = 0$, for every $P_j \notin \mathcal{CS}$ and where every honest party is present in $\mathcal{CS}$, such that $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$.*
– **Asynchronous Network**: *Except with probability $2^{-\mathsf{ssec}}$, almost-surely, all honest parties eventually output $y = f(x^{(1)}, \ldots, x^{(n)})$ where $x^{(j)} = 0$, for every $P_j \notin CoreSet$ and where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$.*

The protocol incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n^{12} \cdot \log |\mathbb{F}|)$ bits and makes $\mathcal{O}(n^3)$ calls to Π_{BA}. Moreover, irrespective of the network type, the view of the adversary remains independent of the inputs of the honest parties in $\mathcal{CS}$.

10 Impossibility Result

Here we show the necessity of the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition for network agnostic MPC. In fact we show that the condition is even necessary for network agnostic BA. For this, we generalize the impossibility proof of [15] which shows the impossibility of network agnostic BA against threshold adversaries if $2t_s+t_a \geq n$.

Theorem 10.1. *Let $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ conditions respectively, where $\mathcal{Z}_a \subset \mathcal{Z}_s$.[16] Moreover, let the parties have access to the setup of an unconditional PKI. Furthermore, let Π be an n-party protocol, which is a $\mathcal{Z}_s$-secure SBA protocol in the synchronous network and which is a $\mathcal{Z}_a$-secure ABA protocol in the asynchronous network (as per Definition 3.1). Then Π exists only if $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.*

Proof. The proof is by contradiction. Let Π exist, even if $\mathcal{Z}_s$ and $\mathcal{Z}_a$ *do not* satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Then, there exist sets, say $Z_0, Z_1 \in \mathcal{Z}_s$ and $Z_2 \in \mathcal{Z}_a$ such that $Z_0 \cup Z_1 \cup Z_2 \supseteq \mathcal{P}$ holds. For simplicity and without loss of generality, assume that Z_0, Z_1 and Z_2 are *disjoint*. Now consider the following executions of Π. In *all* these executions, parties in Z_0 participate with input 0, and parties in Z_1 participate with input 1.

- **Execution** E_1: In this execution, the network is *synchronous*. All the parties in Z_0 are corrupted by the adversary and simply *abort*, and all parties in Z_2 participate with input 1. Since *all* the *honest* parties (namely the parties in $Z_1 \cup Z_2$) have input 1, from the $\mathcal{Z}_s$*-validity* of Π in the *synchronous* network, the parties in $Z_1 \cup Z_2$ should output 1 after some fixed time, say T_1.
- **Execution** E_2: In this execution, the network is *synchronous*. All the parties in Z_1 are corrupted by the adversary and simply *abort*, and all parties in Z_2 participate with input 0. Since *all* the *honest* parties (namely the parties in $Z_0 \cup Z_2$) have input 0, from the $\mathcal{Z}_s$*-validity* of Π in the *synchronous* network, the parties in $Z_0 \cup Z_2$ should output 0 after some fixed time, say T_2.
- **Execution** E_3: In this execution, the network is *asynchronous*, the adversary corrupts all the parties in Z_2 and behave as follows: the communication between the parties in Z_0 and Z_1 is delayed by at least time $\max(T_1, T_2)$. The adversary communicates with parties in Z_0 and Z_1, such that the views of the parties in Z_0 and Z_1 are identical to E_1 and E_2 respectively. For this, the adversary runs Π with input 0 when interacting with the parties in Z_0 and runs Π with input 1 when interacting with the parties in Z_1. Hence, the parties in Z_0 output 0, while the parties in Z_1 output 1, which violates the $\mathcal{Z}_a$*-consistency* of Π in the *asynchronous* network. This is a contradiction and hence, Π *does not* exist.

[16] The necessity of the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ and $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ conditions follow from the existing results on the impossibility of unconditionally-secure SBA and ABA respectively, without these conditions. The condition $\mathcal{Z}_a \subset \mathcal{Z}_s$ is also necessary since any potential corrupt subset which is tolerable in an *asynchronous* network should also be tolerable if the network is *synchronous*.

11 Acknowledgements

We would like to thank Anirudh Chandramouli for several helpful discussions during the early stages of this research.

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. I. Abraham, D. Dolev, and G. Stern. Revisiting Asynchronous Fault Tolerant Computation with Optimal Resilience. In *PODC*, pages 139–148. ACM, 2020.
3. A. Appan, A. Chandramouli, and A. Choudhury. Perfectly-Secure Synchronous MPC with Asynchronous Fallback Guarantees. In *PODC*, pages 92–102. ACM, 2022.
4. A. Appan, A. Chandramouli, and A. Choudhury. Perfectly Secure Synchronous MPC with Asynchronous Fallback Guarantees Against General Adversaries. *IACR Cryptol. ePrint Arch.*, page 1047, 2022.
5. A. Appan, A. Chandramouli, and A. Choudhury. Revisiting the Efficiency of Asynchronous MPC with Optimal Resilience Against General Adversaries. *J. Cryptol.*, 36(3):16, 2023.
6. R. Bacho, D. Collins, C. Liu-Zhang, and J. Loss. Network-Agnostic Security Comes for Free in DKG and MPC. Cryptology ePrint Archive, Paper 2022/1369, 2022.
7. 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.
8. 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.
9. Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer Verlag, 2006.
10. Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer Verlag, 2007.
11. M. Ben-Or. Another Advantage of Free Choice (Extended Abstract): Completely Asynchronous Agreement Protocols. In *PODC*, pages 27–30. ACM, 1983.
12. M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.
13. 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.
14. M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In *PODC*, pages 183–192. ACM, 1994.
15. 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.

16. 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.
17. E. Blum, C. Liu 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.
18. G. Bracha. An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In *PODC*, pages 154–162. ACM, 1984.
19. R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.
20. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.
21. R. Canetti and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *STOC*, pages 42–51, 1993.
22. 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.
23. A. Choudhury. Almost-Surely Terminating Asynchronous Byzantine Agreement Against General Adversaries with Optimal Resilience. In *ICDCN*, pages 167–176. ACM, 2023.
24. A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. *IEEE Trans. Information Theory*, 63(1):428–468, 2017.
25. A. Choudhury and A. Patra. On the Communication Efficiency of Statistically Secure Asynchronous MPC with Optimal Resilience. *J. Cryptol.*, 36(2):13, 2023.
26. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1999.
27. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.
28. G. Deligios and C. Liu-Zhang. Synchronous Perfectly Secure Message Transmission with Optimal Asynchronous Fallback Guarantees. *IACR Cryptol. ePrint Arch.*, page 1397, 2022.
29. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *J. ACM*, 40(1):17–47, 1993.
30. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.
31. Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.
32. M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382, 1985.
33. 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.
34. Matthias Fitzi. *Generalized Communication and Security Models in Byzantine Agreement*. PhD thesis, 2002.
35. D. Ghinea, C. Liu-Zhang, and R. Wattenhofer. Optimal Synchronous Approximate Agreement with Asynchronous Fallback. In *PODC*, pages 70–80. ACM, 2022.

36. 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.
37. V. Goyal, Y. Song, and C. Zhu. Guaranteed Output Delivery Comes Free in Honest Majority MPC. In *CRYPTO*, volume 12171 of *Lecture Notes in Computer Science*, pages 618–646. Springer, 2020.
38. Martin Hirt and Ueli Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC*, pages 25–34. ACM, 1997.
39. Martin Hirt and Ueli Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.
40. 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.
41. M. Ito, A. Saito, and T. Nishizeki. Secret Sharing Schemes Realizing General Access Structures). In *Global Telecommunication Conference, Globecom*, pages 99–102. IEEE Computer Society, 1987.
42. 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.
43. K. Kursawe and F. C. Freiling. Byzantine Fault Tolerance on General Hybrid Adversary Structures. Technical Report, RWTH Aachen, 2005.
44. U. M. Maurer. Secure Multi-party Computation Made Simple. In *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2002.
45. A. Momose and L. Ren. Multi-Threshold Byzantine Fault Tolerance. In *CCS*, pages 1686–1699. ACM, 2021.
46. Atsuki Momose and Ling Ren. Multi-threshold byzantine fault tolerance. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1686–1699, New York, NY, USA, 2021. Association for Computing Machinery.
47. A. Mostéfaoui, H. Moumen, and M. Raynal. Signature-Free Asynchronous Binary Byzantine Consensus with t < n/3, O(n2) Messages, and O(1) Expected Time. *J. ACM*, 62(4):31:1–31:21, 2015.
48. A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015.
49. Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.
50. B. Pfitzmann and M.Waidner. Information-theoretic Pseudosignatures and Byzantine Agreement for $t \geq n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.
51. Michael O. Rabin. Randomized Byzantine Generals. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 403–409, 1983.
52. T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pages 73–85. ACM, 1989.
53. A. C. Yao. Protocols for Secure Computations (Extended Abstract). In *FOCS*, pages 160–164. IEEE Computer Society, 1982.

A Properties of the Network Agnostic BA Protocol

In this section, we prove the properties of the network agnostic BA protocol (see Fig 1 for the protocol). We first formally present the sub-protocol Π_{PW} and prove its properties.

A.1 Protocol Π_{PW}: Synchronous BA with Asynchronous Guaranteed Liveness

Protocol Π_{PW} is presented in Fig 15, where for simplicity we assume that the input of each party is a bit. The protocol is very simple. Each party P_i uses a Dolev-Strong (DS) style protocol [30] to broadcast its input b_i. The protocol runs for $t+1$ "rounds", where t is the size of the largest set in $\mathcal{Z}_s$. Each party P_i accumulates values on the behalf of every party P_j in a set ACC_{ij}. A bit b is added to ACC_{ij} during round r only if P_i receives signatures on b from r distinct parties *including* P_j. Party P_i computes the final output by taking the "majority" among the accumulated values. This is done by computing a final set FIN_i of values based on each ACC_{ij} set. Since the DS protocol is designed for the *synchronous* network, for convenience, we present the protocol Π_{PW} in a round-based fashion, where the parties set the duration of each round to Δ and *will know* the beginning and end of each round.

Protocol $\Pi_{\mathsf{PW}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$

- **Initialization**: Initialize the following sets.
  - For $j = 1, \ldots, n$: the set of values accumulated on the behalf of P_j, $\mathsf{ACC}_{ij} = \emptyset$.
  - t: size of the largest set in $\mathcal{Z}_s$.
  - The final set of values to be considered while computing the output $\mathsf{FIN}_i = \emptyset$.
- **Round 0**: On having input bit b_i, sign b_i to obtain the signature σ_{ii}. Set $\mathsf{SET}_i = \{\sigma_{ii}\}$ and send (b_i, i, SET_i) to every party $P_j \in \mathcal{P}$.
- **Round $r = 0$ to $t+1$**: On receiving (b, j, SET_j) in round r, check if *all* the following hold.
  - SET_j contains valid signatures on b from r *distinct* parties, *including* P_j;
  - $b \notin \mathsf{ACC}_{ij}$.

  If all the above hold, then do the following.
  - Add b to ACC_{ij}.
  - If $r \neq (t+1)$, then compute a signature σ_{ij} on b and send $(b, j, \mathsf{SET}_j \cup \{\sigma_{ij}\})$ to every party $P_k \in \mathcal{P}$.
- **Output Computation**: At time $(t+1) \cdot \Delta$, do the following.
  - If ACC_{ij} contains exactly one value b, then add (j, b) to FIN_i. Else, add $(j, \bot)$ to FIN_i.
  - If there exists a set $Z \in \mathcal{Z}_s$ and a value $b \neq \bot$ such that, for every $P_j \in \mathcal{P} \setminus Z$, (j, b) belongs to FIN_i, then output b. Else, output $\bot$.

Fig. 15: Synchronous BA with asynchronous guaranteed liveness. The above code is executed by every party P_i with input b_i.

We next prove the properties of the protocol Π_{PW}, which are a straightforward generalization of the properties of the DS protocol.

Lemma A.1. *Protocol Π_{PW} achieves $\mathcal{Z}_s$-Consistency in a synchronous network.*

Proof. We claim that each *honest* party P_i computes the same set ACC_{ij} corresponding to *every* party P_j, by time $(t+1)\cdot\Delta$. Assuming the claim is true, the proof then follows from the fact that FIN_i is computed *deterministically* at the time $(t+1)\cdot\Delta$, based on the sets ACC_{ij}. To prove the claim, consider an *arbitrary* honest party P_i and an *arbitrary* P_j. If P_i includes b to ACC_{ij}, then we show that by the time $(t+1)\cdot\Delta$, the value b will be present in the set ACC_{kj} of *every* honest party P_k. For this, we consider the following two cases.

- **Case 1 - P_i added b to ACC_{ij} during round r where $r \leq t$**: In this case, P_i must have received (b, j, SET_j) in round r, where SET_j contained valid signatures on b from $r-1$ distinct parties *apart* from P_j. Party P_i then computes σ_{ij}, adds this to SET_j, and sends (b, j, SET_j) to every party. When P_k receives this during round $r+1$, it will find that SET_j contains r valid signatures on b apart from party P_j's, including σ_{ij}. Hence, P_k will add b to ACC_{kj}. Since $r+1 \leq t+1$, this happens by time $(t+1)\cdot\Delta$.
- **Case 2 - P_i added b to ACC_{ij} during round $r = t+1$**: In this case P_i must have received (b, j, SET_j) in round $t+1$, where SET_j contained valid signatures on b from t distinct parties *apart* from P_j. This means that, in total, P_i has received valid signatures on b from $t+1$ distinct parties. Among these, at least one party, say P_h, must be honest, as there can be at most t corrupt parties. This means that P_h must have added b to ACC_{hj} during some round r', where $r' \leq t$. Thus, as argued in the previous case, each party P_k also adds b to ACC_{kj} by round $t+1$ and hence time $(t+1)\cdot\Delta$.

Lemma A.2. *Protocol Π_{PW} achieves $\mathcal{Z}_s$-Validity in a synchronous network.*

Proof. Suppose that all honest parties have the same input b. Corresponding to each *honest* party P_j, every *honest* P_i sets $\mathsf{ACC}_{ij} = \{b\}$. This is because P_i would receive a valid signature on b from P_j during round 0, and adds b to ACC_{ij}. Further, P_i will not add any $b' \neq b$ to ACC_{ij} during any of the rounds, since the adversary *cannot* forge a signature on b' on the behalf of P_j. Thus, for each honest P_j, party P_i adds (j, b) to FIN_i. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Hence corresponding to every $P_j \in \mathcal{H}$, then value (j, b) is added to FIN_i of every $P_i \in \mathcal{H}$ by time $(t+1)\cdot\Delta$. Moreover, since $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ conditions holds, $\mathcal{P} \setminus Z^\star \notin \mathcal{Z}_s$. Consequently, as per the "majority" rule, every party in $\mathcal{H}$ outputs b.

Lemma A.3. *In protocol Π_{PW}, irrespective of the network type, all honest parties obtain an output at the time $(t+1)\cdot\Delta$.*

Proof. The proof follows from the fact that irrespective of the network type, the parties compute an output (which could be $\perp$) at the local time $(t+1)\cdot\Delta$.

If the inputs of the parties are of size ℓ bits, then we invoke ℓ instances of Π_{PW}. The following lemma describes the communication cost incurred while doing this.

Lemma A.4. *If the inputs of the parties are of size ℓ bits, then protocol Π_{PW} incurs a communication of $\mathcal{O}(n^4 \cdot \ell \cdot |\sigma|)$ bits from the honest parties.*

Proof. During round 0, each party signs its input and sends this to every other party, incurring a total communication of $\mathcal{O}(\ell \cdot n^2 \cdot |\sigma|)$ bits. During the next t rounds, each party P_i sends (b, j, SET_j) to every other party P_k at most *once*. This is because P_i sends this only if $b \notin \mathsf{ACC}_{ij}$ holds, and does not do this once it adds b to ACC_{ij}. Considering all possibilities for b, i, j and k, and taking into account that SET_j will contain $\mathcal{O}(n)$ signatures, the communication cost of this will be $\mathcal{O}(\ell \cdot n^4 \cdot |\sigma|)$ bits.

The proof of Lemma 3.1 now follows from Lemma A.1-A.4.

A.2 Protocool Π_{Acast}: Asynchronous Broadcast with Synchronous Guarantees

In this section, we prove the properties of the protocol Π_{Acast} (see Fig 2 for the protocol description).

Lemma 3.2. *Protocol Π_{Acast} achieves the following properties.*

- *Asynchronous Network: The protocol is a $\mathcal{Z}_a$-secure broadcast protocol.*
- *Synchronous Network:* **(a) $\mathcal{Z}_s$-Liveness***: If* Sen *is honest, then all honest parties obtain an output within time 3Δ.* **(b) $\mathcal{Z}_s$-Validity***: If* Sen *is honest, then every honest party with an output, outputs m.* **(c) $\mathcal{Z}_s$-Consistency***: If* Sen *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 $T + \Delta$.*
- *Communication Complexity: $\mathcal{O}(n^3 \cdot \ell \cdot |\sigma|)$ bits are communicated by the honest parties, where ℓ is the size of* Sen*'s input.*

Proof. We first consider a *synchronous* network, followed by an *asynchronous* network.

Properties in the Synchronous Network. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Suppose Sen is *honest*. Then by time Δ, each party in $\mathcal{H}$ receives $\langle(\mathsf{propose}, m)\rangle_{\mathsf{Sen}}$ from Sen and no honest party ever receives $\langle(\mathsf{propose}, m')\rangle_{\mathsf{Sen}}$ from any party by time 2Δ, for any $m' \neq m$, since signature of an *honest* Sen *cannot* be forged. Thus, every $P_j \in \mathcal{H}$ sends $\langle(\mathsf{vote}, m)\rangle_j$ by time 2Δ. Consequently, by time 3Δ, every $P_i \in \mathcal{H}$ will have a quorum $\mathcal{C}(m)$ of legitimately signed $\langle(\mathsf{vote}, m)\rangle_j$ messages corresponding to every $P_j \in \mathcal{H}$. The parties $P_k \in Z^\star$ may send signed $\langle(\mathsf{vote}, m')\rangle_k$ messages where $m' \neq m$ and consequently the parties in $\mathcal{H}$ may also have a quorum $\mathcal{C}(m')$ of legitimately signed $\langle(\mathsf{vote}, m')\rangle_k$ messages corresponding to every $P_k \in Z^\star$. However, since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition, $\mathcal{P} \setminus Z^\star = \mathcal{H} \notin \mathcal{Z}_s$. Consequently, the parties in $\mathcal{H}$ outputs m by time 3Δ, as the condition for outputting an $m' \neq m$ will be never satisfied for the parties in $\mathcal{H}$. This proves the $\mathcal{Z}_s$-liveness and $\mathcal{Z}_s$-validity.

We now consider Sen to be *corrupt*. We first show that no two parties in $\mathcal{H}$ can vote for different messages. On the contrary, let $P_i \in \mathcal{H}$ sends $\langle \mathsf{vote}, m') \rangle_i$ at time T_i, and let $P_j \in \mathcal{H}$ sends $\langle (\mathsf{vote}, m'') \rangle_j$ at time T_j, where $T_j \geq T_i$. This implies that P_i must have received $\langle (\mathsf{propose}, m') \rangle_{\mathsf{Sen}}$ from Sen within time $T_i - \Delta$, and would have sent $\langle (\mathsf{propose}, m') \rangle_{\mathsf{Sen}}$ to P_j. And P_j would have received $\langle (\mathsf{propose}, m') \rangle_{\mathsf{Sen}}$ from P_i within time T_i. Now since $T_i \leq T_j$, it implies that P_j *would not* have sent $\langle (\mathsf{vote}, m'') \rangle_j$ at time T_j, and this is a contradiction.

Now based on the above fact, we proceed to prove that $\mathcal{Z}_s$-consistency holds. Let $P_h \in \mathcal{H}$ outputs $m^\star$ at time T. This implies that at time T, there exists a subset $Z_\alpha \in \mathcal{Z}_s$, such that P_h has a quorum $\mathcal{C}(m^\star)$ of legitimately signed $\langle (\mathsf{vote}, m) \rangle_j$ messages, corresponding to every $P_j \in \mathcal{P} \setminus Z_\alpha$. Now since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition, it follows that $\mathcal{H} \cap (\mathcal{P} \setminus Z_\alpha) \neq \emptyset$. This implies that there exists at least one party in $\mathcal{H}$, say P_k, who has voted for $m^\star$ by sending a $\langle (\mathsf{vote}, m) \rangle_k$ message. Consequently, no other party in $\mathcal{H}$ every votes for $m^{\star\star} \neq m^\star$. The parties in $Z^\star$ may vote for $m^{\star\star} \neq m^\star$. But since $\mathcal{P} \setminus Z^\star \notin \mathcal{Z}_s$, it follows that no party in $\mathcal{H}$ will ever have a sufficiently large quorum of legitimately signed vote messages for $m^{\star\star}$ to output $m^{\star\star}$. Since P_h sends $\mathcal{C}(m^\star)$ to all parties at time T, every other party in $\mathcal{H}$ will receive $\mathcal{C}(m^\star)$ by time $T+\Delta$. Consequently, all the parties in $\mathcal{H}$ will output $m^\star$, latest by time $T + \Delta$.

Properties in the Asynchronous Network. We now consider an asynchronous network. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We first consider an *honest* Sen. Each party in $\mathcal{H}$ eventually receives $\langle (\mathsf{propose}, m) \rangle_{\mathsf{Sen}}$ from Sen. Furthermore, no party in $\mathcal{H}$ ever receives $\langle (\mathsf{propose}, m') \rangle_{\mathsf{Sen}}$ from any party, for any $m' \neq m$, since the signature of an honest Sen cannot be forged. Hence, each party in $\mathcal{H}$ eventually sends a signed vote message for m, which is eventually delivered to every party in $\mathcal{H}$. The parties in $Z^\star$ may send signed vote messages for $m' \neq m$. However, since $\mathcal{Z}_a$ satisfies the $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ condition, it follows that each party in $\mathcal{H}$ eventually outputs m and the conditions for outputting $m' \neq m$ will be never satisfied for any party in $\mathcal{H}$. This proves the $\mathcal{Z}_a$-liveness and $\mathcal{Z}_a$-consistency.

Next, consider a *corrupt* Sen. Let $P_h \in \mathcal{H}$ outputs $m^\star$. This implies there exists a subset $Z_\alpha \in \mathcal{Z}_s$, such that P_h has a quorum $\mathcal{C}(m^\star)$ of legitimately signed vote messages for $m^\star$, corresponding to every party in $\mathcal{P} \setminus Z_\alpha$. Now consider an *arbitrary* $P_i \in \mathcal{H}$, where $P_i \neq P_h$. We claim that for any $Z \in \mathcal{Z}_s$, party P_i will *never* have a quorum $\mathcal{C}(m^{\star\star})$ of legitimately signed vote messages for $m^{\star\star}$, corresponding to the parties in $\mathcal{P} \setminus Z$. On the contrary, let P_i eventually have a quorum $\mathcal{C}(m^{\star\star})$ of legitimately signed vote messages for $m^{\star\star}$, corresponding to every party in $\mathcal{P} \setminus Z_\beta$, for some $Z_\beta \in \mathcal{Z}_s$. Now since the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition is satisfied, it follows that $\mathcal{H} \cap (\mathcal{P} \setminus Z_\alpha) \cap (\mathcal{P} \setminus Z_\beta) \neq \emptyset$. This implies that there exists at least one party from $\mathcal{H}$, say P_k, such that P_k has voted *both* for $m^\star$, as well as $m^{\star\star}$, which is a contradiction. Consequently, P_i will never output $m^{\star\star}$. Now, since P_h sends $\mathcal{C}(m^\star)$ to all the parties, party P_i eventually receives $\mathcal{C}(m^\star)$ and outputs $m^\star$. This proves $\mathcal{Z}_a$-consistency.

Finally, the communication complexity follows from the fact that irrespective of the type of network, every party may have to send a quorum of up to $\mathcal{O}(n)$ signed vote messages, to every other party.

A.3 Properties of the Protocol Π_{BC}

In this section, we prove the properties of the protocol Π_{BC} (see Fig 3 for the formal description).

Theorem 3.2. *Protocol Π_{BC} achieves the following, with a communication complexity of $\mathcal{O}(n^4 \cdot \ell \cdot |\sigma|)$ bits, where $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{PW}}$.*
- *Synchronous network:*
  - **(a)** $\mathcal{Z}_s$**-Liveness***: At time T_{BC}, each honest party has an output.*
  - **(b)** $\mathcal{Z}_s$**-Validity***: If Sen is honest, then at time T_{BC}, each honest party outputs m.*
  - **(c)** $\mathcal{Z}_s$**-Consistency***: If Sen is corrupt, then the output of every honest party is the same at time T_{BC}.*
  - **(d)** $\mathcal{Z}_s$**-Fallback Consistency***: If Sen 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 $T + \Delta$.*
- *Asynchronous Network:*
  - **(a)** $\mathcal{Z}_a$**-Liveness***: At time T_{BC}, each honest party has an output.*
  - **(b)** $\mathcal{Z}_a$**-Weak Validity***: If Sen is honest, then at time T_{BC}, each honest party outputs m or $\bot$.*
  - **(c)** $\mathcal{Z}_a$**-Fallback Validity***: If Sen is honest, then each honest party with output $\bot$ at time T_{BC}, eventually outputs m through fallback-mode.*
  - **(d)** $\mathcal{Z}_a$**-Weak Consistency***: If Sen is corrupt, then there exists some $m^\star \neq \bot$, such that at time T_{BC}, each honest party outputs either $m^\star$ or $\bot$.*
  - **(e)** $\mathcal{Z}_a$**-Fallback Consistency***: If Sen 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 $\mathcal{Z}_s$-liveness and $\mathcal{Z}_a$-liveness properties follow 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.

Properties in the Synchronous Network. If Sen is *honest*, then due to the $\mathcal{Z}_s$*-liveness* and $\mathcal{Z}_s$*-validity* properties of Π_{Acast} in the *synchronous* network (see Lemma 3.2), all honest parties receive m from the Acast of Sen at time 3Δ. Consequently, all honest parties participate with input m in the instance of Π_{PW}. The $\mathcal{Z}_s$*-guaranteed liveness* and $\mathcal{Z}_s$*-validity* properties of Π_{PW} in the *synchronous* network (see Lemma 3.1) guarantees that at time $3\Delta + T_{\mathsf{PW}}$, all honest parties will have m as the output from the instance of Π_{PW}. As a result, all honest parties output m at time T_{BC}, thus proving the $\mathcal{Z}_s$*-validity* property.

To prove the $\mathcal{Z}_s$-*consistency* property, we consider a *corrupt* Sen. From the $\mathcal{Z}_s$-*consistency* property of Π_{PW} in the *synchronous* network (see Lemma 3.1), all honest parties will have the *same* output from the instance of Π_{PW} at time T_{BC}. If *all* honest parties have the output $\perp$ for Π_{BC} at time T_{BC}, then $\mathcal{Z}_s$-*consistency* holds trivially. So, consider the case when some *honest* party, say P_i, has the output $m^\star \neq \perp$ for Π_{BC} at time T_{BC}. This implies that all honest parties have the output $m^\star$ from the instance of Π_{PW}. Moreover, at time 3Δ, at least one *honest* party, say P_h, has received $m^\star$ from the Acast of Sen. If the latter does not hold, then all honest parties would have participated with input $\perp$ in the instance of Π_{PW}, and from the $\mathcal{Z}_s$-*validity* of Π_{PW} in the *synchronous* network (see Lemma 3.1), all honest parties would compute $\perp$ as the output during the instance of Π_{PW}, which is a contradiction. Since P_h has received $m^\star$ from Sen's Acast at time 3Δ, it follows from the $\mathcal{Z}_s$-*consistency* property of Π_{Acast} in the *synchronous* network (see Lemma 3.2) that *all* honest parties will receive $m^\star$ from Sen's Acast by time 4Δ. Moreover, $4\Delta < 3\Delta + T_{\mathsf{PW}}$ holds. Consequently, at time $3\Delta + T_{\mathsf{BC}}$, *all* honest parties will have $m^\star$ from Sen's Acast *and* as the output of Π_{PW}, implying that all honest parties output $m^\star$ for Π_{BC}.

We next prove the $\mathcal{Z}_s$-*fallback consistency* property for which we again consider a *corrupt* Sen. 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 $\mathcal{Z}_s$-*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 $m^\star$.[17] 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 Sen. It then follows from the $\mathcal{Z}_s$-*consistency* of Π_{Acast} in the *synchronous* network that every honest party will also receive $m^\star$ from the Acast of Sen, latest by time $T + \Delta$ and output $m^\star$. This completes the proof of all the properties in the *synchronous* network.

Properties in the Asynchronous Network. The $\mathcal{Z}_a$-*weak validity* property follows from the $\mathcal{Z}_a$-*validity* property of Π_{Acast} in the *asynchronous* network (see Lemma 3.2), which ensures that no honest party ever receives an m' from the Acast of Sen, 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 $\mathcal{Z}_a$-*weak consistency* property follows using similar arguments as used to prove $\mathcal{Z}_s$-*consistency* in the *synchronous* network; however we now rely on the $\mathcal{Z}_a$-*validity* and $\mathcal{Z}_a$-*consistency* properties of Π_{Acast} in the asynchronous network (see Lemma 3.2). The latter property ensures that for a *corrupt* Sen, two different honest parties never end up receiving m_1 and m_2 from the Acast of Sen, where $m_1 \neq m_2$.

For the $\mathcal{Z}_s$-*fallback validity* property, consider an *honest* Sen, and let P_i be an arbitrary *honest* party who outputs $\perp$ at (local) time T_{BC}. Since the parties keep

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

on participating in the protocol beyond time T_{BC}, it follows from the $\mathcal{Z}_a$-*liveness* and $\mathcal{Z}_a$-*validity* properties of Π_{Acast} in the *asynchronous* network (see Lemma 3.2) that party P_i will *eventually* receive m from the Acast of Sen, 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 $\mathcal{Z}_a$-*fallback consistency* property, we consider a *corrupt* Sen. 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 Sen. Now, consider an arbitrary *honest* P_i. From the $\mathcal{Z}_a$-*liveness* and $\mathcal{Z}_a$-*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 $\mathcal{Z}_a$-*consistency* property of Π_{Acast} in the *asynchronous* network (see Lemma 3.2), it follows that P_i will also eventually obtain $m^\star$ from the Acast of Sen, by executing the steps of the fallback-mode of Π_{BC}. Consequently, party P_i eventually changes its output from $\perp$ to $m^\star$.

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

A.4 Properties of the Protocol Π_{SBA}

In this section, we prove the properties of the protocol Π_{SBA} (see Fig 4 for the formal description).

Theorem 3.3. *Protocol Π_{SBA} achieves the following where $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$, incurring a communication of $\mathcal{O}(n^5 \cdot |\sigma|)$ bits.*

- *Synchronous Network: the protocol is a $\mathcal{Z}_s$-secure SBA protocol where honest parties have an output, different from $\perp$, at time T_{SBA}.*
- *Asynchronous Network: the protocol achieves $\mathcal{Z}_a$-guaranteed liveness and $\mathcal{Z}_a$-weak validity, such that all honest parties have an output at (local) time T_{SBA}.*

Proof. The communication complexity simply follows from the fact that n instances of Π_{BC} are invoked in the protocol. The *guaranteed liveness*, both in the synchronous and asynchronous network trivially follows from the $\mathcal{Z}_s$-*liveness* and $\mathcal{Z}_a$-*liveness* of Π_{BC} (see Theorem 3.2), which ensures that all the n instances of Π_{BC} produce *some* output within (local) time T_{BC}, both in a synchronous as well as an asynchronous network, through *regular* mode. Hence, at local time $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$, all honest parties will have some output. We next prove the rest of the properties in a *synchronous* network.

Properties in the Synchronous Network. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. In a *synchronous* network, the instances of Π_{BC} corresponding to the senders in $\mathcal{H}$, result in an output different from $\perp$ for all honest parties (follows from the $\mathcal{Z}_s$-*validity* property of Π_{BC} in the *synchronous* network, Theorem 3.2). Hence $\mathcal{H} \subseteq \mathcal{SV}$ will hold.

Moreover, from the $\mathcal{Z}_s$-*consistency* property of Π_{BC} in the *synchronous* network (Theorem 3.2), all the parties in $\mathcal{H}$ obtain a common output from the i^{th} instance of Π_{BC}, for $i = 1, \ldots, n$, at time T_{BC}. Hence, all honest parties output the *same* value, different from $\perp$, at time T_{BC}, proving the $\mathcal{Z}_s$-*consistency* of Π_{SBA}. Finally, if all parties in $\mathcal{H}$ have the same input bit b, then only the instances of Π_{BC} corresponding to the parties in $\mathcal{SV} \setminus \mathcal{H}$ may output $\bar{b} = 1 - b$. However, $\mathcal{SV} \setminus \mathcal{H} \in \mathcal{Z}_s$. Moreover, $\mathcal{SV} \setminus Z^\star \notin \mathcal{Z}_s$ (since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition). It then follows that all honest parties output b, proving $\mathcal{Z}_s$-*validity* of Π_{SBA}.

Properties in the Asynchronous Network. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Suppose all the parties in $\mathcal{H}$ have the same input bit b. Let $P_i \in \mathcal{H}$ be an *arbitrary* party, that obtains an output c, *different* from $\perp$, at time T_{BC}. This implies that there exists a subset of parties $\mathcal{SV}$ for P_i, where $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, such that P_i has obtained a Boolean output $b_i^{(j)}$ from the Π_{BC} instances, corresponding to every $P_j \in \mathcal{SV}$. Moreover, there also exists a subset of parties $\mathcal{SV}_i \subseteq \mathcal{SV}$, where $\mathcal{SV} \setminus \mathcal{SV}_i \in \mathcal{Z}_s$, such that the output $b_i^{(j)} = c$, corresponding to every $P_j \in \mathcal{SV}_i$. Now since the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition is satisfied, it follows that $\mathcal{Z}_a$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{SV}_i, \mathcal{Z}_a)$ condition and hence $\mathcal{SV}_i \cap \mathcal{H} \neq \emptyset$. Consequently, $c = b$ holds. This proves the $\mathcal{Z}_a$-*weak validity* in the asynchronous network.

A.5 Protocol Π_{GA}: Asynchronous Graded Agreement with Synchronous Validity

To design protocol Π_{GA}, we first design a sub-protocol Π_{Prop} for proposing values.

A.5.1 Π_{Prop}: A Network Agnostic Protocol for Proposing Values

Protocol Π_{Prop} takes an input value from each party from the set $\{0, 1, \lambda\}$ and outputs a set prop of proposed values for each party. Liveness is ensured in an *asynchronous* network as long as each honest party holds one of *two* inputs. In an asynchronous network, it will be ensured that each value in the output prop must be the input of some *honest* party. Moreover, if any two honest parties output a singleton set for prop, then they must output the *same* set. In a *synchronous* network, validity and liveness are ensured as long as each honest party participates with the same input. Protocol Π_{Prop} is presented in figure 16.

Protocol $\Pi_{\mathsf{Prop}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$

1. Set $\mathsf{vals} = \mathsf{prop} = \emptyset$.
2. On having the input $v \in \{0, 1, \lambda\}$, send $(\mathsf{prepare}, v)$ to every party $P_j \in \mathcal{P}$.
3. On receiving $(\mathsf{prepare}, b)$ for some $b \in \{0, 1, \lambda\}$ from a set of parties $S^{(b)}$ satisfying $\mathbb{Q}^1(S^{(b)}, \mathcal{Z}_s)$ condition, send $(\mathsf{prepare}, b)$ to all $P_j \in \mathcal{P}$, if not sent earlier.
4. Upon receiving the message $(\mathsf{prepare}, b)$ for some $b \in \{0, 1, \lambda\}$ from parties in set $\mathsf{PrepareSet}^{(b)} = \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$, set $\mathsf{vals} = \mathsf{vals} \cup \{b\}$.
5. Upon adding the first value b to vals, send $(\mathsf{propose}, b)$ to every party $P_j \in \mathcal{P}$.

6. Upon receiving $(\mathsf{propose}, b)$ messages from a set of parties $\mathsf{ProposeSet} = \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$ on values $b \in \mathsf{vals}$, let $\mathsf{prop} \subseteq \mathsf{vals}$ be the set of values carried by those messages. Output prop.

Fig. 16: Sub-protocol to propose values. The above code is executed by every party P_i with input $v \in \{0, 1, \lambda\}$

The guarantees provided by Π_{Prop} are proven in a series of lemmas below. In the below proofs, we assume that $Z^\star$ is the set of corrupt parties.

Lemma A.5. *Suppose that the network is asynchronous. If two honest parties P_i and P_j output $\{b\}$ and $\{b'\}$ respectively, then $b = b'$.*

Proof. Since P_i outputs $\{b\}$, it must have received $(\mathsf{propose}, b)$ from a set of parties $\mathsf{ProposeSet} = \mathcal{P} \setminus Z$ for some $Z \in \mathcal{Z}_s$. Similarly, since P_j outputs $\{b'\}$, *it* must have received $(\mathsf{propose}, b')$ from a set of parties $\mathsf{ProposeSet}' = \mathcal{P} \setminus Z'$ for some $Z' \in \mathcal{Z}_s$. Let $\mathsf{ProposeSet}_{\mathcal{H}}$ and $\mathsf{ProposeSet}_{\mathcal{H}}$ be the set of *honest* parties in $\mathsf{ProposeSet}$ and $\mathsf{ProposeSet}'$ respectively. Since $\mathcal{Z}_s, \mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition and $Z^\star \in \mathcal{Z}_a$, it follows that $\mathsf{ProposeSet}_{\mathcal{H}} \cap \mathsf{ProposeSet}'_{\mathcal{H}} \neq \emptyset$. Let $P_h \in \mathsf{ProposeSet}_{\mathcal{H}} \cap \mathsf{ProposeSet}'_{\mathcal{H}}$. If $b \neq b'$, this would mean that P_h has sent both $(\mathsf{propose}, b)$ and $(\mathsf{propose}, b')$, which is a contradiction, since an honest party sends *at most* one $\mathsf{propose}$ message as per the protocol.

Lemma A.6. *Suppose that the network is asynchronous. If no honest party has input v, then no honest party outputs* prop *containing v.*

Proof. If v was not input by any honest party, then no honest party sends $(\mathsf{prepare}, v)$ in step 2. Hence, no honest party receives $(\mathsf{prepare}, v)$ from a set of parties $S^{(v)}$ which satisfies the $\mathbb{Q}^1(S^{(v)}, \mathcal{Z}_s)$ condition during step 3, since such a set must contain at least one honest party. Consequently, no honest party sends $(\mathsf{propose}, v)$. Thus, no honest party adds v to vals and no honest party outputs prop containing v.

The following lemmas help prove liveness.

Lemma A.7. *Suppose that the network is asynchronous. If all honest parties hold one of two inputs, say v_0 and v_1, then all honest parties eventually compute an output.*

Proof. We first show that every honest party eventually sends a $\mathsf{propose}$ message. Let $\mathcal{H}_0$ and $\mathcal{H}_1$ be the sets of *honest* parties holding inputs v_0 and v_1 respectively and let $\mathcal{H}$ be the set of *honest* parties. We know that due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{H}, \mathcal{Z}_s)$ condition. Now, consider the following cases.

- **Case 1 - $\mathcal{H}_0$ satisfies the $\mathbb{Q}^1(\mathcal{H}_0, \mathcal{Z}_s)$ condition**: In this case, $\mathcal{H}_0$ is a candidate for the set $S^{(v_0)}$, since all the parties in $\mathcal{H}_0$ send $(\mathsf{prepare}, v_0)$ in step 2.

– **Case 2 - $\mathcal{H}_0$ does not satisfy the $\mathbb{Q}^1(\mathcal{H}_0, \mathcal{Z}_s)$ condition**: In this case, $\mathcal{Z}_s$ must satisfy the $\mathbb{Q}^1(\mathcal{H}_1, \mathcal{Z}_s)$ condition, since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(2)}(\mathcal{H}, \mathcal{Z}_s)$ condition. Then similar to what was argued in the previous case, $\mathcal{H}_1$ is a candidate for the set $S^{(v_1)}$.

Let v_b be the input corresponding to the candidate set $S^{(v_b)}$. Every honest party will now eventually send $(\mathsf{prepare}, v_b)$ in step 3. This would mean that the set of honest parties $\mathcal{H} = \mathcal{P} \setminus Z^\star$ is a candidate for the set $\mathsf{PrepareSet}^{(v_b)}$. Thus, every honest party eventually adds v_b to vals and sends a $\mathsf{propose}$ message for some value v_b. This way, every honest party eventually receives $\mathsf{propose}$ messages from every other honest party for v_b and thus, the set $\mathcal{H}$ also forms a candidate for the set $\mathsf{ProposeSet}$. Thus, all honest parties eventually compute an output.

Lemma A.8. *If the network is synchronous and if all honest parties participate with input v, then all honest parties output* $\mathsf{prop} = \{v\}$ *at time* 2Δ.

Proof. Let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Then in step 2, every party in $\mathcal{H}$ sends $(\mathsf{prepare}, v)$ to every party, which is delivered within Δ time. Thus, at time Δ, all the parties in $\mathcal{H}$ receive $(\mathsf{prepare}, v)$ from the parties in $\mathcal{H}$ and add v to vals. Further, no other value $v^\star$ will be added to vals, since only the parties in $Z^\star$ may send a $\mathsf{prepare}$ message for $v^\star$ and $\mathcal{Z}_s$ *does not* satisfy the $\mathbb{Q}^{(1)}(Z^\star, \mathcal{Z}_s)$ condition. Thus, all the parties in $\mathcal{H}$ send $(\mathsf{propose}, v)$ in step 5, which gets delivered to all the parties in $\mathcal{H}$ at time 2Δ. Consequently, all the parties in $\mathcal{H}$ output $\mathsf{prop} = \{v\}$.

Lemma A.9. *Protocol* Π_{Prop} *incurs a communication of* $\mathcal{O}(n^2)$ *bits.*

Proof. The proof follows from the fact that each party sends $(\mathsf{prepare}, b)$ to every other party at most once for any value of $b \in \{0, 1, \lambda\}$.

A.5.2 The Graded Agreement Protocol

We now present protocol Π_{GA} (Figure 17) based on protocol Π_{Prop}. The protocol cleverly "stitches" together two instances of Π_{Prop}, by defining the input for the second instance based on the output from the first instance. Each party, with input either 0 or 1, participates in the first instance of Π_{Prop} with their input. Since the parties participate with one of two inputs, this instance will eventually complete (in an *asynchronous* network), and the parties obtain an output, say prop_1. Only if prop_1 is a singleton set, say $\{b\}$, for some party, then that party participates in the second instance of Π_{Prop} with the input b. Otherwise, it participates in the second instance with a *default* input of λ. Since no two honest parties can output different singleton sets for prop_1, this ensures that each honest party participates with an input of either b or λ (in an *asynchronous* network). Thus, the second instance of Π_{Prop} also eventually completed with an output, say prop_2. This also ensures that prop_2 can contain only values b and λ. If prop_2 contains *only* b for some honest party, then that party outputs b with a grade of 2. If prop_2 contains b along with λ, then the party outputs b with a grade of 1. Else, if prop_2 contains only λ, then the party outputs $\perp$ with a grade of 0. If the network is *synchronous* and all honest parties start the protocol with the same input, then *both* prop_1 as well as prop_2 will be a *singleton* set containing that value and hence all honest parties will output that value with the highest grade.

Protocol $\Pi_{\mathsf{GA}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$

1. On having the input $v \in \{0, 1\}$, set $b_1 = v$. Participate in an instance of the protocol Π_{Prop} with input b_1 and wait for its completion. Let prop_1 be the output computed during the instance of prop.
2. If $\mathsf{prop}_1 = \{b\}$ for some $b \in \{0, 1\}$, then set $b_2 = b$. Else, set $b_2 = \lambda$. Then participate in an instance of the protocol Π_{Prop} with input b_2 and wait for its completion. Let prop_2 be the output computed from this instance of prop.
3. If $\mathsf{prop}_2 = \{b'\}$ and $b' \neq \lambda$, then output $(b', 2)$. If $\mathsf{prop}_2 = \{b', \lambda\}$ where $b' \in \{0, 1\}$, then output $(b', 1)$. Else, if $\mathsf{prop}_2 = \{\lambda\}$, then output $(\perp, 0)$.

Fig. 17: Asynchronous graded agreement with synchronous validity. The above code is executed by every party P_i with input $v \in \{0, 1\}$.

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

Lemma A.10. *Protocol Π_{GA} achieves the following in a synchronous network, where $T_{\mathsf{GA}} = 4\Delta$.*

- **(a)** $\mathcal{Z}_s$**-Liveness**: *If all honest parties participate in the protocol with the same input, then at time T_{GA}, all honest parties obtain an output.*
- **(b)** $\mathcal{Z}_s$**-Graded Validity**: *If every honest party's input is b, then all honest parties with an output, output $(b, 2)$.*

Proof. If all honest parties participate in the protocol Π_{GA} with the same input b, then from Lemma A.8, all honest parties output $\mathsf{prop}_1 = \{b\}$ at time 2Δ. Thus, all honest parties participate with input $b_2 = b$ in the second instance of Π_{GA} and, once again from Lemma A.8, output $\mathsf{prop}_2 = \{b\}$ at time 4Δ. Thus, all honest parties output $(b, 2)$.

Lemma A.11. *Protocol Π_{GA} achieves the following in an asynchronous network.*

- **(a)** $\mathcal{Z}_a$**-Liveness**: *If all honest parties participate in the protocol with a binary input, then each honest party eventually obtains an output.*
- **(b)** $\mathcal{Z}_a$**-Graded Validity**: *If every honest party's input is b, then all honest parties with an output, output $(b, 2)$.*
- **(c)** $\mathcal{Z}_a$**-Graded Consistency**: *If two honest parties output grades g, g', then $|g - g'| \leq 1$ holds; moreover, if two honest parties output (v, g) and (v', g') with $g, g' \geq 1$, then $v = v'$.*

Proof. Since each honest party participates with a binary input, from Lemma A.7, each party eventually outputs some value for prop_1 during the first instance of prop. Now there are two possible cases.

- **Case 1 - Some honest party outputs $\{b\}$ as its value for prop_1 where** $b \in \{0, 1\}$: From Lemma A.5, no honest party can output $\{b'\}$, where $b \neq b'$, as prop_1. Thus, each honest party participates with input either b or λ for the second instance of prop.

– **Case 2 - No honest party outputs $\{b\}$ as its value for prop_1 for any** $b \in \{0,1\}$: In this case, all honest parties participate with input λ in the second instance of prop.

In either case, the honest parties participate in the second instance of Π_{Prop} with no more than *two* different inputs. Thus, from Lemma A.7, all parties eventually compute some value for prop_2 during the second instance of prop and hence compute some output for protocol Π_{GA}. This proves the $\mathcal{Z}_a$-Liveness.

We next prove the $\mathcal{Z}_a$-Graded Consistency. We first show that the grades output by any two parties differ by at most 1. For this, suppose that some *honest* party P_i outputs $(b, 2)$. We show that no other honest party P_j can output $(\perp, 0)$. Since P_i output $(b, 2)$, from Lemma A.5, P_j cannot output $\mathsf{prop}_2 = \{\lambda\}$. Thus, P_j cannot output $(\perp, 0)$. Next, we show that any two *honest* parties which output *non-zero* grades must output the same value. Similar to what was argued for the proof of $\mathcal{Z}_a$-Liveness, there exists a bit b such that each honest party participates in Π_{Prop} with input b or λ during step 2. Thus, $\mathsf{prop}_2 \subseteq \{b, \lambda\}$ for every honest party. This means that any honest party which outputs a non-zero grade must output it along with the bit b.

We finally prove the $\mathcal{Z}_a$-Graded Validity. Suppose that each honest party participates with the same input bit b. From Lemma A.7, we know that all honest parties output some value for prop_1. From Lemma A.6, all honest parties must output $\mathsf{prop}_1 = \{b\}$. Hence, all honest parties participate in Π_{Prop} in step 2 with input b. By the same argument, all honest parties output $\mathsf{prop}_2 = \{b\}$. Hence, all honest parties output $(b, 2)$.

Lemma A.12. *Protocol Π_{GA} incurs a communication of $\mathcal{O}(n^2)$ bits.*

Proof. The proof follows from Lemma A.9, since Π_{Prop} is invoked twice in the protocol.

A.6 Properties of the Protocol Π_{ABA}

In this section, we prove the properties of the protocol Π_{ABA} (see Fig 5 for the protocol steps). We start with the properties in the *asynchronous* network first, which mostly follows from [23] and are recalled from [23]. We start with the validity property.

Lemma A.13. *In protocol Π_{ABA}, if the network is asynchronous and all honest parties have the same input bit b, then all honest parties eventually output b.*

Proof. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties. If every honest party has the same input bit b, then from the *$\mathcal{Z}_a$-Graded Validity* of Π_{GA} in the *asynchronous* network (Lemma A.11), all honest parties eventually output $(b, 2)$ at the end of the first as well as the second instance of the Π_{GA} protocol during the first iteration. Consequently, every *honest* party eventually sends a signed (ready, b) message to all the parties and only the parties in $Z^\star$ may send a signed $(\mathsf{ready}, \overline{b})$

message. It now follows easily from the steps of the output computation stage that *no* honest party ever sends a signed $(\mathsf{ready}, \bar{b})$ message and all hence honest parties eventually output b.

We next prove the *consistency* property.

Lemma A.14. *In protocol Π_{ABA}, if the network is asynchronous and if any honest party outputs b, then every other honest party eventually outputs b.*

Proof. Let P_i be the *first honest* party who sends a signed ready message for some bit $b \in \{0, 1\}$, during some iteration, say iteration r. We show that *no* honest party ever sends a signed $(\mathsf{ready}, \bar{b})$ message during iteration r or in the subsequent iterations. Since P_i has sent a signed ready message for b, it implies that P_i outputs $(b, 2)$ in the second instance of the Π_{GA} protocol during iteration r and sets $\mathsf{committed}$ to true. Then, from the $\mathcal{Z}_a$-*Graded Consistency* of Π_{GA} in the *asynchronous* network (Lemma A.11), every other honest party outputs either $(b, 2)$ or $(b, 1)$ in the second instance of the Π_{GA} protocol during iteration r. Consequently, no other honest party sends the signed $(\mathsf{ready}, \bar{b})$ message during iteration r. Also, from the protocol steps, all honest parties update their input to b for the next iteration. This further implies that all honest parties will continue to input b to each subsequent invocation of Π_{GA}, ignoring the output of Π_{CoinFlip}, for as long as they continue running. Consequently, no honest party ever sends a signed $(\mathsf{ready}, \bar{b})$ message.

Now let some *honest* party, say P_h, computes the output b during iteration k. This implies that P_h receives the signed (ready, b) message from a set of parties, say $\mathcal{T}$, such that $\mathcal{P} \setminus \mathcal{T} \in \mathcal{Z}_s$. The set $\mathcal{T}$ is bound to have at least one *honest* party, due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition, implying that at least one honest party has sent a signed (ready, b) message, either during the iteration k or some previous iteration. From the protocol steps, P_h sends $\mathcal{C}(b)$, the set of signed (ready, b) messages of the parties in $\mathcal{T}$ to all other parties, which get eventually delivered. Moreover, as shown above, no honest party will ever send a signed $(\mathsf{ready}, \bar{b})$ message. Consequently, every honest party eventually receives sufficiently many numbers of signed (ready, b) messages and outputs b.

We next prove that at the end of each iteration, the updated value of all honest parties will be the same with the probability at least $\frac{1}{2n}$.

Lemma A.15. *In protocol Π_{ABA}, if the network is asynchronous and if all honest parties participate during iteration k, then with probability at least $\frac{1}{2n}$, all honest parties have the same updated bit b at the end of iteration k.*

Proof. To prove the lemma statement, we consider an event Agree, which denotes that all honest parties have the same input for the second instance of Π_{GA} during iteration k. If the event Agree occurs, then from the $\mathcal{Z}_a$-*Graded Validity* of Π_{GA} in the *asynchronous* network (Lemma A.11), all honest parties will have the same updated bit at the end of iteration k. We show that the event Agree occurs during iteration k with a probability of at least $\frac{1}{2n}$. For this, we consider two different possible cases with respect to the output from the first instance of Π_{GA} during iteration k.

- **Case I: No honest party obtains an output** $(b, 2)$ **for any** $b \in \{0, 1\}$ **during the first instance of** Π_{GA}**.** In this case, all honest parties set the output from the instance of Π_{CoinFlip} during iteration k as the input for the second instance of Π_{GA}. From the $(\mathcal{Z}_a, p)$-*commonness* of Π_{CoinFlip} in *asynchronous* network [23], all honest parties will have the same output bit Coin_k from the instance of Π_{CoinFlip} with a probability of at least $p = \frac{1}{n} > \frac{1}{2n}$.
- **Case II: Some honest party obtains an output** $(b, 2)$ **during the first instance of** Π_{GA}**.** In this case, the $\mathcal{Z}_a$-*Graded Consistency* of Π_{GA} in the *asynchronous* network (Lemma A.11) ensure that all honest parties obtain the output $(b, 2)$ or $(b, 1)$ from the first instance of Π_{GA}. Moreover, from the protocol steps, the output of the instance of Π_{CoinFlip} during iteration k is *not* revealed, until the first honest party generates an output from the first instance of Π_{GA} during iteration k. Consequently, the output bit b from the first instance of Π_{GA} is *independent* of the output of Π_{CoinFlip}. From the $(\mathcal{Z}_a, p)$-*commonness* of Π_{CoinFlip} in *asynchronous* network [23], all honest parties will have the same output bit Coin_k from the instance of Π_{CoinFlip} with a probability of at least $p = \frac{1}{n}$. Then the probability that $\mathsf{Coin}_k = b$ holds is at least $\frac{1}{2} \cdot \frac{1}{n} = \frac{1}{2n}$ and with this probability, all honest parties will have the same input for the second instance of Π_{GA}.

We next derive the expected number of iterations required in the protocol Π_{ABA} for the honest parties to produce an output. This automatically gives the expected running time in an asynchronous network, since each iteration takes a constant time.

Lemma A.16. *If the network is asynchronous, then in protocol* Π_{ABA}*, it requires expected* $\mathcal{O}(n^2)$ *iterations for the honest parties to compute an output.*

Proof. To prove the lemma, we need to derive the expected number of iterations, until all the honest parties have the same input during the second instance of Π_{GA} of an iteration. This is because once all the honest parties have the same input during the second instance of Π_{GA} of an iteration, then all honest parties will set $\mathsf{committed}$ to true at the end of that iteration and start sending signed ready messages, followed by computing an output. Let τ be the random variable which counts the number of iterations until all honest parties have the same input during the second instance of Π_{GA} in an iteration. Then the probability that $\tau = k$ is given as:

$$\Pr(\tau = k) = \Pr(\tau \neq 1) \cdot \Pr(\tau \neq 2 \mid \tau \neq 1) \cdot \ldots \cdot Pr(\tau \neq (k-1) \mid \tau \neq 1 \cap \ldots \cap \tau \neq (k-2)) \cdot \Pr(\tau = k \mid \tau \neq 1 \cap \ldots \cap \tau \neq (k-1)).$$

From Lemma A.15, every multiplicand on the right-hand side in the above equation, except the last one, is upper bounded by $(1 - \frac{1}{2n})$ and the last multiplicand is upper bounded by $\frac{1}{2n}$. Hence, we get

$$\Pr(\tau = k) \leq (1 - \frac{1}{2n})^{k-1}(\frac{1}{2n}).$$

Now the expected value $E(\tau)$ of τ is computed as follows:

$$
\begin{aligned}
E(\tau) &= \sum_{k=0}^{\infty} \tau \cdot \Pr(\tau = k) \\
&\le \sum_{k=0}^{\infty} k(1 - \frac{1}{2n})^{k-1}(\frac{1}{2n}) \\
&= \frac{1}{2n} \sum_{k=0}^{\infty} k(1 - \frac{1}{2n})^{k-1} \\
&= \frac{1}{1 - (1 - \frac{1}{2n})} + \frac{1 - \frac{1}{2n}}{\Big(1 - (1 - \frac{1}{2n})\Big)^2} \\
&= 2n + 4n^2 - 2n = 4n^2
\end{aligned}
$$

The expression for $E(\tau)$ is a sum of AGP up to infinite terms, which is given by $\frac{a}{1-r} + \frac{dr}{(1-r)^2}$, where $a = 1$, $r = 1 - \frac{1}{2n}$ and $d = 1$. Hence, we have $E(\tau) \le 4n^2$.

We finally prove the properties of the protocol Π_{ABA} in a *synchronous* network.

Lemma A.17. *If the network is synchronous and if all honest parties have the same input $b \in \{0, 1\}$ during Π_{ABA}, then all honest parties output b, at time $T_{\mathsf{ABA}} = T_{\mathsf{CoinFlip}} + 2T_{\mathsf{GA}} + \Delta$.*

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. If all the parties in $\mathcal{H}$ participate with input b, then from the $\mathcal{Z}_s$*-liveness* and $\mathcal{Z}_s$*-Graded Validity* of Π_{GA} in the *synchronous* network (Lemma A.10), all the parties in $\mathcal{H}$ output $(b, 2)$ during the first instance of Π_{GA} at time T_{GA} in the first iteration. The $\mathcal{Z}_s$*-Guaranteed Liveness* of Π_{GA} in the *synchronous* network [23] ensures that all honest parties compute some output from the instance of Π_{CoinFlip} during the first iteration at the time $T_{\mathsf{GA}} + T_{\mathsf{CoinFlip}}$. Since the parties in $\mathcal{H}$ output $(b, 2)$ during the first instance of Π_{GA}, they participate with input b during the second instance of Π_{GA}. Consequently, from the $\mathcal{Z}_s$*-liveness* and $\mathcal{Z}_s$*-Graded Validity* of Π_{GA} in the *synchronous* network, all the parties in $\mathcal{H}$ compute the output $(b, 2)$ during the second instance of Π_{GA} in the first iteration at the time $2T_{\mathsf{GA}} + T_{\mathsf{CoinFlip}}$. Hence every party in $\mathcal{H}$ sends a signed ready message for b at the time $2T_{\mathsf{GA}} + T_{\mathsf{CoinFlip}}$, which gets delivered at the time T_{ABA}. Moreover, only the parties in $Z^\star$ may send a signed ready message for $\overline{b}$. Since $\mathcal{P} \setminus \mathcal{H} = Z^\star \in \mathcal{Z}_s$ and since $\mathcal{H} \notin \mathcal{Z}_s$ (due to the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition), it follows that all the parties in $\mathcal{H}$ will have sufficiently many signed ready messages for b at the time T_{ABA} to output b.

The proof of Theorem 3.4 now follows from Lemma A.13-A.17. The communication complexity follows from the communication complexity of Π_{CoinFlip} [23] and the communication complexity of Π_{GA} (Lemma A.12) and the fact that in a *synchronous* network, only a *constant* number of invocations of Π_{GA} and Π_{CoinFlip} are involved, while in an *asynchronous* network, there are $\text{poly}(n)$ invocations of Π_{GA} and Π_{CoinFlip} in *expectation*.

B Properties of Our Network Agnostic ICP

In this section, we prove the properties of our network-agnostic ICP (see Fig 6 for the formal details). Throughout this section, we assume that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the conditions $\mathcal{Z}_a \subset \mathcal{Z}_s$, $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$, $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ and $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$.

Lemma B.1. *If* S, I *and* R *are honest, then the following hold during protocol* Π_{Auth} *and* Π_{Reveal}*.*

- $\mathcal{Z}_s$**-Correctness***: In a synchronous network, each honest party sets* $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ *to* 1 *during* Π_{Auth} *at time* $T_{\mathsf{Auth}} = \Delta + 4T_{\mathsf{BC}}$*. Moreover* R *outputs* s *during* Π_{Reveal} *which takes* $T_{\mathsf{Reveal}} = \Delta$ *time.*
- $\mathcal{Z}_a$**-Correctness***: In an asynchronous network, each honest party eventually sets* $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ *to* 1 *during* Π_{Auth} *and* R *eventually outputs* s *during* Π_{Reveal}*.*

Proof. We first start with the *synchronous* network. Let $Z^\star \in \mathcal{Z}_s$ be set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. During Π_{Auth}, S chooses a random t-degree signing-polynomial $F(x)$ such that $s = F(0)$ holds, a random t-degree masking-polynomial $M(x)$, and computes verification points (α_i, v_i, m_i) such that $v_i = F(\alpha_i)$ and $m_i = M(\alpha_i)$ hold. S then sends the signing-polynomial $F(x)$ and masking-polynomial $M(x)$ to I, and the corresponding verification-point (α_i, v_i, m_i) to each verifier P_i. Consequently, each verifier in $\mathcal{H}$ receives its verification-point by time Δ, and indicates this by broadcasting $(\mathsf{Received}, i)$. Since $\mathcal{P} \setminus \mathcal{H} = Z^\star \in \mathcal{Z}_s$, from the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network (see Theorem 3.2), it follows that at time $\Delta + T_{\mathsf{BC}}$, S will find a set $\mathcal{SV}$, such that $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$, where each verifier in $\mathcal{SV}$ has indicated that it has received its verification-point. Consequently, S will broadcast $\mathcal{SV}$ at time $\Delta + T_{\mathsf{BC}}$. From the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network, I will receive $\mathcal{SV}$ at time $\Delta + 2T_{\mathsf{BC}}$. Moreover, due to the $\mathcal{Z}_s$*-Consistency* and $\mathcal{Z}_s$*-Validity* of Π_{BC} in the *synchronous* network, party I would have gotten $(\mathsf{Received}, i)$, corresponding to every verifier $P_i \in \mathcal{SV}$, by time $\Delta + 2T_{\mathsf{BC}}$. Furthermore, $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$ will hold. Hence, I will randomly select $d \in \mathbb{F}$, compute $B(x) = dF(x) + M(x)$, and broadcast $(d, B(x))$. From the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network, this will be delivered to every honest party, including S, by time $\Delta + 3T_{\mathsf{BC}}$. Moreover, S will find that $B(\alpha_j) = dv_j + m_j$ holds for all the verifiers $P_j \in \mathcal{SV}$. Consequently, S will broadcast an OK message, which is received by every $P_i \in \mathcal{H}$ at time $\Delta + 4T_{\mathsf{BC}}$, due to the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network. Thus, each $P_i \in \mathcal{H}$ sets $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ to 1, while I additionally sets $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to $F(x)$ at time $\Delta + 4T_{\mathsf{BC}}$.

During Π_{Reveal}, I will send $F(x)$ to R, and each verifier $P_i \in \mathcal{H} \cap \mathcal{SV}$ will send its verification point (α_i, v_i, m_i) to R. These points and the polynomial $F(x)$ are received by R within Δ time. Moreover, the condition $v_i = F(\alpha_i)$ will hold true for these points, and consequently, these points will be *accepted*. Since $\mathcal{SV} \setminus (\mathcal{H} \cap \mathcal{SV}) \subseteq Z^\star \in \mathcal{Z}_s$, it follows that at time Δ, receiver R will find a subset $\mathcal{SV}' \subseteq \mathcal{SV}$ where $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$, such that the points corresponding to all the parties in $\mathcal{SV}'$ are accepted. This implies that R will output $s = F(0)$ within time Δ.

The proof for the asynchronous case is similar as above, except that each "favourable" event occurs *eventually*, and follows from the fact that every set in $\mathcal{Z}_a$ is a subset of some set in $\mathcal{Z}_s$. Moreover, we rely on the properties of Π_{BC} in the *asynchronous* network.

We next prove the privacy property, for which we again need to consider an *honest* S, I and R.

Lemma B.2. *If* S, I *and* R *are honest, then the view of* Adv *remains independent of* s *during* Π_{Auth} *and* Π_{Reveal}*, irrespective of the network type.*

Proof. We prove privacy in a *synchronous* network. The privacy in an *asynchronous* network automatically follows, since $\mathcal{Z}_a$ is a subset of $\mathcal{Z}_s$. Let $t = \max\{|Z| : Z \in \mathcal{Z}_s\}$ and let $Z^\star \in \mathcal{Z}_s$ be the set of corrupt parties. For simplicity and without loss of generality, let $|Z^\star| = t$. During Π_{Auth}, the adversary Adv learns t verification-points $\{(\alpha_i, v_i, m_i)\}_{P_i \in Z^\star}$. However, since $F(x)$ is a random t-degree polynomial with $F(0) = s$, the points $\{(\alpha_i, v_i)\}_{P_i \in Z^\star}$ are distributed independently of s. That is, for every candidate $s \in \mathbb{F}$ from the point of view of Adv, there is a corresponding unique t-degree polynomial $F(x)$, such that $F(\alpha_i) = v_i$ holds corresponding to every $P_i \in Z^\star$.

During Π_{Auth}, the adversary Adv also learns d and the blinded-polynomial $B(x) = dF(x) + M(x)$. However, this does not add any new information about s to the view of the adversary. This is because $M(x)$ is a random t-degree polynomial and Adv learns t points on $M(x)$, corresponding to the parties in $Z^\star$. Hence, for every candidate $M(x)$ polynomial from the point of view of Adv where $M(\alpha_i) = m_i$ holds for every $P_i \in Z^\star$, there is a corresponding unique t-degree polynomial $F(x)$, such that $F(\alpha_i) = v_i$ holds corresponding to every $P_i \in Z^\star$, and where $dF(x) + M(x) = B(x)$. Finally, Adv does not learn anything new about s during Π_{Reveal}, since the verification-points and the signing-polynomial are sent only to R, who is *honest* as per the lemma conditions.

We next prove the unforgeability property, for which we have to consider a *corrupt* I.

Lemma B.3. *If* S, R *are honest,* I *is corrupt and if* R *outputs* $s' \in \mathbb{F}$ *during* Π_{Reveal}*, then* $s' = s$ *holds except with probability at most* $\epsilon_{\mathsf{ICP}} \stackrel{def}{=} \frac{nt}{|\mathbb{F}|-1}$*, where* $t = \max\{|Z| : Z \in \mathcal{Z}_s\}$*, irrespective of the network type.*

Proof. Let $\mathcal{H}$ be the set of *honest* parties in $\mathcal{P}$ and let $Z^\star = \mathcal{P} \setminus \mathcal{H}$ be the set of *corrupt* parties. Since R outputs s' during Π_{Reveal}, it implies that during Π_{Auth}, the variable $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ is set to 1 by R. This further implies that S has broadcasted an OK message during Π_{Auth}, which also implies that during Π_{Auth}, I had broadcasted a t-degree blinded-polynomial $B(x)$, and S broadcasted the set $\mathcal{SV}$. Furthermore, S has verified that $B(\alpha_i) = dv_i + m_i$ holds for every verifier $P_i \in \mathcal{SV}$. Now during Π_{Reveal}, if I sends $F(x)$ as $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to R, then $s' = s$ holds with probability 1. So, consider the case when I sends $F'(x)$ as $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to R, where $F'(x)$ is a t-degree polynomial such that $F'(x) \neq F(x)$

and where $F'(0) = s'$. In this case, we claim that except with probability at most $\frac{nt}{|\mathbb{F}|-1}$, the verification-point of no *honest* verifier from $\mathcal{SV}$ will get accepted by R during Π_{Reveal}, with respect to $F'(x)$. Now, assuming that the claim is true, the proof follows using the following arguments, depending upon the network type.

– *Synchronous Network*: In this case, all the verifiers in $\mathcal{H}$ will be present in $\mathcal{SV}$. This is because, each verifier $P_i \in \mathcal{H}$ would have received its verification-point from S during Π_{Auth}, within time Δ and indicates this by broadcasting $(\mathsf{Received}, i)$, which is received by S at time $\Delta + T_{\mathsf{BC}}$. Let $\mathcal{SV}'$ be the set of verifiers from which R receives verification points which it accepts. Since the verification point of *none* of the honest verifier will be accepted. Hence $(\mathcal{H} \cap \mathcal{SV}') = \emptyset$ and so $\mathcal{H} \subseteq \mathcal{SV} \setminus \mathcal{SV}'$ must hold. Since $\mathcal{H}$ satisfies the $\mathbb{Q}^1(\mathcal{P}, \mathcal{Z}_s)$ condition, $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$ will *never* hold true. Hence, R will *not* output $s' \neq s$.
– *Asynchronous Network*: In this case, we first note that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(1,1)}(\mathcal{SV}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. This is because $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$ and $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. From the steps of Π_{Reveal}, it follow that for R to output $F'(0)$, R should find a subset of verifiers $\mathcal{SV}' \subseteq \mathcal{SV}$, where $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$, such that the verification-points of all the verifiers in $\mathcal{SV}'$ are accepted by R. This further implies that $\mathcal{SV}' \cap \mathcal{H} \neq \emptyset$, as $\mathcal{Z}_a$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{SV}', \mathcal{Z}_a)$ condition. And hence $\mathcal{SV}'$ has at least one *honest* verifier, whose verification-point is accepted with respect to $F'(x)$. However, from the above claim, it is not possible and hence R will *not* output $s' \neq s$.

We now prove the claimed statement. So consider an *arbitrary* verifier $P_i \in \mathcal{H} \cap \mathcal{SV}$ from whom R receives the verification-point (α_i, v_i, m_i) during Π_{Reveal}. This point can be *accepted* with respect to $F'(x)$, only if either of the following holds.

- $v_i = F'(\alpha_i)$: This is possible with probability at most $\frac{t}{|\mathbb{F}|-1}$. This is because $F'(x)$ and $F(x)$, being distinct t-degree polynomials, can have at most t points in common. And the evaluation-point α_i corresponding to P_i, being randomly selected from $\mathbb{F} - \{0\}$, will *not* be known to I.
- $dv_i + m_i \neq B(\alpha_i)$: This is impossible, as otherwise S would have *not* broadcasted OK during Π_{Auth}, which is a contradiction.

As there could be up to $n-1$ *honest* verifiers in $\mathcal{SV}$, it follows from the union bound that except with probability at most $\frac{nt}{|\mathbb{F}|-1}$, the verification-point of no *honest* verifier from $\mathcal{SV}$ will get accepted by R during Π_{Reveal}, with respect to $F'(x)$.

We next prove the non-repudiation property, for which we have to consider a *corrupt* S.

Lemma B.4. *If* S *is corrupt,* I, R *are honest and if* I *sets* $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ *during* Π_{Auth}*, then the following hold, except with probability at most* $\frac{n}{|\mathbb{F}|-1}$.

– $\mathcal{Z}_s$**-Non-Repudiation***: In a synchronous network,* R *outputs* s *during* Π_{Reveal}*, which takes* $T_{\mathsf{Reveal}} = \Delta$ *time.*
– $\mathcal{Z}_a$**-Non-Repudiation***: In an asynchronous network,* R *eventually outputs* s *during during* Π_{Reveal}*.*

Proof. Let $\mathcal{H}$ be the set of honest parties in $\mathcal{P}$ and $Z^{\star}$ be the set of corrupt parties, where $\mathcal{H} = \mathcal{P} \setminus Z^{\star}$. Since I has set $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ during Π_{Auth}, it implies that I has set the variable $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ to 1. This further implies that I has broadcasted $(d, B(x))$, where $B(x) = dF(x) + M(x)$, and where $F(x)$ and $M(x)$ are the t-degree signing and masking-polynomials received by I from S. Moreover, I also received the set of supporting verifiers $\mathcal{SV}$ from the broadcast of S, and verified that $\mathcal{P} \setminus \mathcal{SV} \in \mathcal{Z}_s$ holds. Furthermore, S has broadcasted an OK message. Consequently, from the *consistency* properties of Π_{BC} (see Theorem 3.2), irrespective of the network type, all *honest* parties including R eventually set $\mathsf{authCompleted}_{(\mathsf{S},\mathsf{I},\mathsf{R})}$ to 1. Moreover, I sets $\mathsf{ICSig}(\mathsf{S}, \mathsf{I}, \mathsf{R}, s)$ to $F(x)$, where $s = F(0)$. During Π_{Reveal}, I sends $F(x)$ to R. Moreover, every verifier $P_i \in \mathcal{H} \cap \mathcal{SV}$ sends its verification-point (α_i, v_i, m_i) to R. In a *synchronous* network, these will be received by R within time Δ, while in an asynchronous network, these will be eventually received by R. We claim that except with probability at most $\frac{n}{|\mathbb{F}|-1}$, all these verification-points are accepted by R. Now, assuming that the claim is true, the proof follows from the fact that $\mathcal{H} \cap \mathcal{SV} = \mathcal{SV} \setminus Z^{\star}$, and $Z^{\star} \in \mathcal{Z}_s$ holds, *irrespective* of the network type (since $\mathcal{Z}_a \subset \mathcal{Z}_s$). Consequently, R accepts the verification-points from a subset of the verifiers $\mathcal{SV}' \subseteq \mathcal{SV}$ where $\mathcal{SV} \setminus \mathcal{SV}' \in \mathcal{Z}_s$. And hence it outputs s, either within time Δ in a synchronous network, or eventually, in an asynchronous network.

We now proceed to prove the claim. So consider an *arbitrary* verifier $P_i \in \mathcal{H} \cap \mathcal{SV}$ whose verification-point (α_i, v_i, m_i) is received by R during Π_{Reveal}. Now, there are two possible cases, depending upon the relationship that holds between $F(\alpha_i)$ and v_i during Π_{Auth}.

- $v_i = F(\alpha_i)$ *holds*: In this case, according to the protocol steps of Π_{Reveal}, the point (α_i, v_i, m_i) is *accepted* by R.
- $v_i \neq F(\alpha_i)$ *holds*: In this case, we claim that except with probability at most $\frac{1}{|\mathbb{F}|-1}$, the condition $dv_i + m_i \neq B(\alpha_i)$ will hold, implying that the point (α_i, v_i, m_i) is *accepted* by R. This is because the *only* way $dv_i + m_i = B(\alpha_i)$ holds is when S distributes (α_i, v_i, m_i) to P_i where $v_i \neq F(\alpha_i)$ and $m_i \neq M(\alpha_i)$ holds, and I selects $d = (M(\alpha_i) - m_i) \cdot (v_i - F(\alpha_i))^{-1}$. However, S will *not* be knowing the random d from $\mathbb{F} \setminus \{0\}$ which I is going to pick, while distributing $F(x), M(x)$ to I, and (α_i, v_i, m_i) to P_i. Hence, the probability that I indeed selects $d = (M(\alpha_i) - m_i) \cdot (v_i - F(\alpha_i))^{-1}$ is $\frac{1}{|\mathbb{F}|-1}$.

As there can be up to $n - 1$ *honest* verifiers in $\mathcal{SV}$, from the union bound, it follows that except with probability at most $\frac{n}{|\mathbb{F}|-1}$, the verification-point of *all* honest verifiers in $\mathcal{SV}$ are accepted by R.

We finally derive the communication complexity.

Lemma B.5. *Protocol Π_{Auth} incurs a communication of $\mathcal{O}(n^5 \cdot \log|\mathbb{F}| \cdot |\sigma|)$ bits. Protocol Π_{Reveal} incurs a communication of $\mathcal{O}(n \cdot \log|\mathbb{F}|)$ bits.*

Proof. During Π_{Auth}, signer S sends t-degree polynomials $F(x)$ and $M(x)$ to I, and verification-points to each verifier. This requires a communication of $\mathcal{O}(n \cdot \log|\mathbb{F}|)$ bits. Intermediary I needs to broadcast $B(x)$ and d using protocol Π_{BC}, while S needs to broadcast the set $\mathcal{SV}$ using Π_{BC}. Moreover, S may need to broadcast s using Π_{BC}. By substituting the communication cost of Π_{BC} (see Theorem 3.2), the overall communication cost of Π_{Auth} turns out to be $\mathcal{O}(n^5 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits. During Π_{Reveal}, I may send $F(x)$ to R, and each verifier may send its verification-point to R. This incurs a communication of $\mathcal{O}(n \cdot \log |\mathbb{F}|)$ bits.

The proof of Theorem 4.1 now follows easily from Lemma B.1-B.5.

C Properties of Network Agnostic VSS

In this section, we prove the properties of the protocol Π_{VSS} (see Fig 7). Throughout this section, we assume that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the conditions $\mathcal{Z}_a \subset \mathcal{Z}_s$, $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$, $\mathbb{Q}^{(3)}(\mathcal{P}, \mathcal{Z}_a)$ and $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$. We start with the properties in a *synchronous* network and first consider an *honest* D. We first show that an *honest* D will broadcast some candidate core-sets, which will be accepted by all honest parties. Moreover, adversary will not learn any additional information about s.

Lemma C.1. *If the network is synchronous and* D *is honest, participating in* Π_{VSS} *with input* s*, then all the following hold, where* $\mathcal{H}$ *is the set of honest parties.*

- *There exists some* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$*, such that* D *broadcasts a message* $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ *at time* $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$*, and every* $P_i \in \mathcal{H}$ *includes* (D, S_p) *to the set* $\mathcal{C}_i$ *at time* $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$*. Moreover, all the following hold for* $q = 1, \ldots, |\mathcal{Z}_s|$*.*
  - *If* $S_q = \mathcal{H}$*, then* $q \notin \mathcal{BS}$*.*
  - $\mathcal{W}_q$ *will be either* S_q *or* $(S_p \cap S_q)$*. Moreover,* $\mathcal{Z}_s$ *will satisfy the* $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ *condition.*
  - *Corresponding to every* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$*, every honest* $P_i \in \mathcal{W}_q$ *will have the IC-signature* $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$ *of every* $P_j \in \mathcal{W}_q$ *for every* $P_k \notin S_q$*, such that the underlying signatures will satisfy the linearity principle. Furthermore, if any corrupt* $P_j \in \mathcal{W}_q$ *has the IC-signature* $\mathsf{ICSig}(P_i, P_j, P_k, s'_q)$ *of any honest* $P_i \in \mathcal{W}_q$ *for any* $P_k \in \mathcal{P}$*, then* $s'_q = s_q$ *holds and the underlying signatures will satisfy the linearity principle.*
  - *Corresponding to every* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$*, every honest* $P_i \in S_q$ *will have the share* s_q*, except with a probability* $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$*, at time* $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$*, where* $s = s_1 + \ldots + s_{|\mathcal{Z}_s|}$*.*
- *The view of the adversary will be independent of* s*.*

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We note that $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$. Since D is honest, it picks the shares $s_1, \ldots, s_{|\mathcal{Z}_s|}$ such that $s = s_1 + \cdots + s_{|\mathcal{Z}_s|}$ and sends s_q to each party $P_i \in S_q$, corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. These shares are delivered within time Δ. Now consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. At time Δ, each party $P_i \in (S_q \cap \mathcal{H})$ starts giving $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$, where $s_{qi} = s_q$

holds. Moreover, the linearity principle is followed while generating these IC-signatures. Then from the $\mathcal{Z}_s$-*correctness* of Π_{Auth} in the *synchronous* network (Theorem 4.1), it follows that at time $\Delta + T_{\mathsf{Auth}}$, each party $P_i \in (S_q \cap \mathcal{H})$ will receive $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ from every $P_j \in (S_q \cap \mathcal{H})$, for every $P_k \in \mathcal{P}$, such that $s_{qj} = s_{qi} = s_q$ holds. Since S_q is arbitrary, it follows that at time $\Delta + T_{\mathsf{Auth}}$, every party $P_i \in \mathcal{H}$ broadcasts an $\mathsf{OK}(i, j)$ message, corresponding to every $P_j \in \mathcal{H}$. From the $\mathcal{Z}_s$-*validity* of Π_{BC} in the *synchronous* network (Theorem 3.2), it follows that these $\mathsf{OK}(i, j)$ messages are received by every party in $\mathcal{H}$ through *regular-mode* at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. Consequently, the set $\mathcal{H}$ constitutes a clique in the consistency graph of every party in $\mathcal{H}$ at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. Now since the set $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$, it follows that at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, there exists *some* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that D computes the core-sets $\{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and broadcast-set $\mathcal{BS}$, followed by broadcasting $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. Moreover, since D is *honest*, it computes the sets $\{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}$ honestly, satisfying the following conditions, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If S_q constitutes a clique in the graph $G^{(\mathsf{D})}$, then $\mathcal{W}_q = S_q$.
- Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\mathsf{D})}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}_q = (S_p \cap S_q)$.
- Else $\mathcal{W}_q = S_q$ and $q \in \mathcal{BS}$.

Note that for each $\mathcal{W}_q$, the condition $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ holds. This is obviously true if $\mathcal{W}_q = (S_p \cap S_q)$, since in this case D also checks that $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition holds. On the other hand, even if $\mathcal{W}_q = S_q$, the condition $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ holds, as $\mathbb{Q}^{(1)}(S_q, \mathcal{Z}_s)$ holds due to the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition. We also note that if $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}$ and consequently, D does not make public the share s_q. This is because as shown above, at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, the parties in S_q constitute a clique in the graph $G^{(\mathsf{D})}$.

Since D broadcasts $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, from the $\mathcal{Z}_s$-*validity* of Π_{BC} in the *synchronous* network, it follows that all the parties in $\mathcal{H}$ will receive $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D, at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. To show that every $P_i \in \mathcal{H}$ will include (D, S_p) to the set $\mathcal{C}_i$, we need to show that all the conditions which hold for D in its graph $G^{(\mathsf{D})}$ at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, are bound to hold for every $P_i \in \mathcal{H}$, at the time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. Namely, all the edges which are present in $G^{(\mathsf{D})}$ at time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, are bound to be present in the graph $G^{(i)}$ every $P_i \in \mathcal{H}$, at the time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. However, this simply follows from the $\mathcal{Z}_s$-*validity*, $\mathcal{Z}_s$-*consistency* and $\mathcal{Z}_s$-*fallback consistency* of Π_{BC} in the *synchronous* network (see Theorem 3.2) and the fact that edges are added to consistency graphs, based on the receipt of $\mathsf{OK}(\star, \star)$ messages, which are broadcasted through various Π_{BC} instances. Consequently, any edge (i, j) which is included in $G^{(\mathsf{D})}$ at the time $\Delta + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, is bound to be included in the graph $G^{(i)}$ of every $P_i \in \mathcal{H}$, latest by time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$.

We next note that corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest $P_i \in \mathcal{W}_q$ will have the share s_q, which is either made public by D as part of the CanCS message or received from D. Each honest $P_i \in W_q$ will thus set $[s]_q$ to s_q at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. We next show that each $P_i \in \mathcal{H}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$

corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, where $P_i \in \mathcal{W}_q$, for every $P_j \in \mathcal{W}_q$ and every $P_k \notin S_q$. This also is set at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. This is because there are two possible cases with respect to q. If $q \in \mathcal{BS}$, then from the protocol steps, $\mathcal{W}_q$ is publicly set to S_q and $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$ is set to the default value. On the other hand, if $q \notin \mathcal{BS}$, then also P_i will possess $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$. This is because as per the protocol steps, since $P_i, P_j \in \mathcal{W}_q$, it follows that P_i must have verified that the edge $(P_i, P_j) \in G^{(i)}$, which further implies that P_i has received $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ from P_j, where $s_{qj} = s_{qi}$ holds. And since D is *honest*, $s_{qi} = s_q$ holds, implying that $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ is the same as $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$. On the other hand, in the protocol, P_i gives $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})$ to every $P_j \in S_q$ for every $P_k \in \mathcal{P}$, where $s_{qi} = s_q$ holds. Hence if any corrupt $P_j \in \mathcal{W}_q$ has $\mathsf{ICSig}(P_i, P_j, P_k, s'_q)$ of any honest $P_i \in \mathcal{W}_q$ for any $P_k \in \mathcal{P}$, then $s'_q = s_q$ holds.

We now show that, corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest party $P_i \in S_q \setminus W_q$ sets $[s]_q$ to s_q, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$, at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$. For this, we first show that P_i sets $[s]_q$ to *some* value. Since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition, the set $\mathcal{W}_q$ contains *at least* one *honest* party, say P_j. Since P_j follows the protocol steps honestly, it reveals $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$ of every $P_k \in \mathcal{W}_q$ to P_i, at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. From the $\mathcal{Z}_s$*-correctness* of ICP in the *synchronous* network (see Theorem 4.1), it follows that P_i will accept these signatures after time T_{Reveal}. On the other hand, even if $P_k \in \mathcal{W}_q$ is *corrupt*, then also from the $\mathcal{Z}_s$*-non-repudiation* property of ICP in the *synchronous* network (see Theorem 4.1), it follows that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, except with a probability ϵ_{ICP}, after time T_{Reveal}. As there can be $\mathcal{O}(n)$ *corrupt* parties in $\mathcal{W}_q$, from the union bound, it follows that except with a probability $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$, party P_i will find a candidate party from $\mathcal{W}_q$, who reveals $[s]_q$, along with the IC-signature of all the parties in $\mathcal{W}_q$, after time T_{Reveal}. Now as there can be $\mathcal{O}(n)$ parties in $S_q \setminus \mathcal{W}_q$, it follows that except with probability $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, every honest party $P_i \in S_q \setminus \mathcal{W}_q$ will find a candidate party from $\mathcal{W}_q$, who reveals $[s]_q$ along with the IC-signature of all the parties in $\mathcal{W}_q$ at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$.

We next show that $P_i \in S_q \setminus W_q$ indeed sets $[s]_q$ to s_q. Suppose that P_i sets $[s]_q$ to some value s'. From the protocols steps, this implies that there exists some $P_j \in \mathcal{W}_q$, such that P_i has accepted $\mathsf{ICSig}(P_k, P_j, P_i, s')$ of every $P_k \in \mathcal{W}_q$, revealed by P_j. If P_j is *honest*, then indeed $s' = [s]_q$, as one of the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, s')$ is the same as $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, corresponding to the *honest* $P_k \in \mathcal{W}_q$, which is guaranteed to exist. So consider the case when P_j is *corrupt*. Moreover, let $P_k \in \mathcal{W}_q$ be an *honest* party (which is guaranteed to exist). In order that $s' \neq [s]_q$, it must be the case that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, s')$, revealed by P_j. However, from the *unforgeability* property of ICP (see Theorem 4.1), this can happen only with probability ϵ_{ICP}. Now as there can be up to $\mathcal{O}(n)$ corrupt parties in $\mathcal{W}_q$, from the union bound, it follows that the probability that P_i outputs $s' \neq [s]_q$ is at most $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$. Since there can be up to $\mathcal{O}(n)$ parties in $S_q \setminus \mathcal{W}_q$, it follows that except with probability at most $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, the output of every honest party in S_q is indeed s_q. Now, there can be $|\mathbb{S}_{\mathcal{Z}_s}|$ possibilities for S_q. From the union bound, it follows that corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest party $P_i \in S_q \setminus W_q$ sets $[s]_q$ to s_q, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

Finally, the privacy for s follows from the fact that throughout the protocol, the view of the adversary remains independent of the share s_q, corresponding to the group S_q, where $S_q = \mathcal{H}$. This is because as shown above, $q \notin \mathcal{BS}$ and consequently, D does not make public the share s_q. Moreover, during the pairwise consistency tests, the view of the adversary remains independent of s_q, when the parties in $\mathcal{H}$ exchange IC-signed s_q, which follows from the *privacy* property of ICP (see Theorem 4.1). Further, while computing the output, $\mathsf{ICSig}(P_k, P_j, P_i, s_q)$ is revealed by party $P_j \in W_q$ only to each party $P_i \in S_q \setminus W_q$. Hence, the adversary does not learn s_q.

An immediate corollary of Lemma C.1 is that if D is honest, then the parties output $[s]$ at time $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$, which follows from the definition of $[\cdot]$-sharing.

Corollary C.1. *If the network is synchronous and* D *is honest and participates in* Π_{VSS} *with input* s*, then the parties output* $[s]$ *at time* $\Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$*, except with a probability of* $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

We next consider a *corrupt* D in the *synchronous* network and show that if any honest party computes an output at time T, then there exists some $s^\star \in \mathbb{F}$ such that the (honest) parties output $[s^\star]$ by time $T + \Delta$, except with probability $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

Lemma C.2. *If the network is synchronous and* D *is corrupt and if any honest party computes an output at time* T*, then there exists some* $s^\star \in \mathbb{F}$*, such that the honest parties output* $[s^\star]$ *by time* $T + \Delta$*, except with a probability of* $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Let $P_\ell \in \mathcal{H}$ be the *first* honest party which computes an output at time T. The way in which the core sets are defined ensures that each set in $\{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}$ must contain at least one honest party. Since Π_{Reveal} takes at least T_{Reveal} time to complete, this means that there exists $S_p \in \mathbb{S}_{\mathcal{Z}_s}$ such that some honest party P_m receives a message $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D by the time $T - T_{\mathsf{Reveal}}$ and includes (D, S_p) to $\mathcal{C}_m$. This further implies that P_m has verified that the following hold, for $q = 1, \ldots, |\mathcal{Z}_s|$, by time $T - T_{\mathsf{Reveal}}$.

- If $q \notin \mathcal{BS}$, then $\mathcal{W}_q$ is either S_q or $(S_p \cap S_q)$. Moreover, $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition and the parties in $\mathcal{W}_q$ constitute a clique in the graph $G^{(m)}$.
- If $q \in \mathcal{BS}$, then D has made public s_q, as part of the CanCS message. Moreover, $\mathcal{W}_q$ is set to S_q.

From the $\mathcal{Z}_s$-*consistency* and $\mathcal{Z}_s$-*fallback consistency* of Π_{BC} in the *synchronous* network (see Theorem 3.2), it follows that every party in $\mathcal{H}$ will receive $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D, latest by time $T - T_{\mathsf{Reveal}} + \Delta$. We next show that each $P_i \in \mathcal{H}$ will include (D, S_p) to

$\mathcal{C}_i$, latest by time $T - T_{\mathsf{Reveal}} + \Delta$. For this, it is enough to show that all the edges which are present in $G^{(m)}$ at time $T - T_{\mathsf{Reveal}}$, are bound to be present in the graph $G^{(i)}$ every $P_i \in \mathcal{H}$, by the time $T - T_{\mathsf{Reveal}} + \Delta$. However, this simply follows from the $\mathcal{Z}_s$*-validity,* $\mathcal{Z}_s$*-consistency* and $\mathcal{Z}_s$*-fallback consistency* of Π_{BC} in the *synchronous* network (see Theorem 3.2) and the fact that edges are added to consistency graphs, based on the receipt of $\mathsf{OK}(\star,\star)$ messages, which are broadcasted through various Π_{BC} instances. Consequently, any edge (i, j) which is included in $G^{(m)}$ at the time T, is bound to be included in the graph $G^{(i)}$ of every $P_i \in \mathcal{H}$, latest by time $T - T_{\mathsf{Reveal}} + \Delta$.

We next show that by time $T - T_{\mathsf{Reveal}} + \Delta$, corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every $P_i \in \mathcal{W}_q$ will have a common share, say $s^\star_q$. If $q \in \mathcal{BS}$, this is trivially true, since in this case, D makes public the share s_q and hence $s^\star_q = s_q$. On the other hand, consider the case when $q \notin \mathcal{BS}$ and consider *arbitrary* $P_i, P_j \in (\mathcal{W}_q \cap \mathcal{H})$. Since P_i and P_j are part of a clique, it follows that P_i and P_j have broadcasted the messages $\mathsf{OK}(i, j)$ and $\mathsf{OK}(j, i)$ respectively. Moreover, these messages were broadcasted, latest by time $T - T_{\mathsf{Reveal}} - T_{\mathsf{BC}}$, since it takes at least T_{BC} time to compute any output in an instance of Π_{BC} in the *synchronous* network (see Theorem 3.2). Now since P_i and P_j have broadcasted $\mathsf{OK}(i, j)$ and $\mathsf{OK}(j, i)$ messages, it implies that they have verified that $s_{qi} = s_{qj}$ holds, where s_{qi} and s_{qj} are the shares, received by P_i and P_j respectively, from D. Let $s_{qi} = s_{qj} = s^\star_q$. We define

$$s^\star \stackrel{def}{=} \sum_{q=1,\ldots,|\mathcal{Z}_s|} s^\star_q.$$

Till now we have shown that there exists some $s^\star \in \mathbb{F}$, such that by time $T - T_{\mathsf{Reveal}} + \Delta$, all the parties in $\mathcal{H}$ will have the core-sets $\mathcal{W}_1, \ldots, \mathcal{W}_{|\mathcal{Z}_s|}$, where $S_q \setminus \mathcal{W}_q \in \mathcal{Z}_a$, for $q = 1, \ldots, |\mathcal{Z}_s|$ and where each $P_i \in (\mathcal{W}_q \cap \mathcal{H})$ will have a common share $[s^\star]_q$. We need to show that by the time $T - T_{\mathsf{Reveal}} + \Delta$, the parties in $\mathcal{H}$ will have the required IC-signatures, as part of $[s^\star]$, satisfying the linearity property. Namely, each $P_i \in \mathcal{H}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, [s^\star]_q)$, corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ where $P_i \in \mathcal{W}_q$, of every $P_j \in \mathcal{W}_q$ and for every $P_k \in \mathcal{P}$. There are two possible cases with respect to q. If $q \in \mathcal{BS}$, then from the protocol steps, $\mathcal{W}_q$ is publicly set to S_q and $\mathsf{ICSig}(P_j, P_i, P_k, [s^\star]_q)$ is set to the default value of $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$, where s_q is made public by D. As per our notations, $s_q = s^\star_q$ for every $q \in \mathcal{BS}$. On the other hand, if $q \notin \mathcal{BS}$, then also P_i will possess $\mathsf{ICSig}(P_j, P_i, P_k, [s^\star]_q)$. This is because as per the protocol steps, since $P_i, P_j \in \mathcal{W}_q$, it follows that P_i must have verified that the edge $(P_i, P_j) \in G^{(i)}$, which further implies that P_i has received $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ from P_j, where $s_{qj} = s_{qi}$ holds. Here s_{qi} is the share received by P_i from D and as per our notation, $s_{qi} = s^\star_q$. Hence $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ is the same as $\mathsf{ICSig}(P_j, P_i, P_k, [s^\star]_q)$. On the other hand, in the protocol, P_i gives $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})$ to every $P_j \in S_q$ for every $P_k \in \mathcal{P}$, where $s_{qi} = [s^\star]_q$ holds. Hence if any corrupt $P_j \in \mathcal{W}_q$ has $\mathsf{ICSig}(P_i, P_j, P_k, s'_q)$, of any honest $P_i \in \mathcal{W}_q$ for any $P_k \in \mathcal{P}$, then $s'_q = [s^\star]_q$ holds. It is easy to see that all the underlying IC-signatures will be linear since the parties follow the linearity principle while generating IC-signatures.

Finally, we now show that every honest party $P_i \in S_q \setminus W_q$ gets $[s^\star]_q$, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$, at time $T + \Delta$. For this, we first show that P_i computes some share on the behalf of S_q. Since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition, the set $\mathcal{W}_q$ contains *at least* one *honest* party, say P_j. Since P_j follows the protocol steps honestly, it reveals $\mathsf{ICSig}(P_k, P_j, P_i, [s^\star]_q)$ of every $P_k \in \mathcal{W}_q$ to P_i, at time $T - T_{\mathsf{Reveal}} + \Delta$. From the $\mathcal{Z}_s$*-correctness* of ICP in the *synchronous* network (see Theorem 4.1), it follows that P_i will accept the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, [s^\star]_q)$, revealed by P_j, after time T_{Reveal}. On the other hand, even if $P_k \in \mathcal{W}_q$ is *corrupt*, then also from the $\mathcal{Z}_s$*-non-repudiation* property of ICP in the *synchronous* network (see Theorem 4.1), it follows that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, [s^\star]_q)$, except with a probability ϵ_{ICP}, after time T_{Reveal}. As there can be $\mathcal{O}(n)$ *corrupt* parties in $\mathcal{W}_q$, from the union bound, it follows that except with a probability $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$, party P_i will find a candidate party from $\mathcal{W}_q$, who reveals $[s^\star]_q$, along with the IC-signature of all the parties in $\mathcal{W}_q$, after time T_{Reveal}. Now as there can be $\mathcal{O}(n)$ parties in $S_q \setminus \mathcal{W}_q$, it follows that except with probability $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, every honest party $P_i \in S_q \setminus \mathcal{W}_q$ will find a candidate party from $\mathcal{W}_q$, who reveals $[s^\star]_q$ along with the IC-signature of all the parties in $\mathcal{W}_q$ at time $T + \Delta$.

We next show that $P_i \in S_q \setminus W_q$ indeed sets $[s^\star]_q$ as the share corresponding to S_q. Suppose that P_i sets the share to some value s'. From the protocol steps, this implies that there exists some $P_j \in \mathcal{W}_q$, such that P_i has accepted the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, s')$ of every $P_k \in \mathcal{W}_q$, revealed by P_j. If P_j is *honest*, then indeed $s' = [s^\star]_q$, as one of the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, s')$ is the same as $\mathsf{ICSig}(P_k, P_j, P_i, [s^\star]_q)$, corresponding to the *honest* $P_k \in \mathcal{W}_q$, which is guaranteed to exist. So consider the case when P_j is *corrupt*. Moreover, let $P_k \in \mathcal{W}_q$ be an *honest* party (which is guaranteed to exist). In order that $s' \neq [s^\star]_q$, it must be the case that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, s')$, revealed by P_j. However, from the *unforgeability* property of ICP (see Theorem 4.1), this can happen only with probability ϵ_{ICP}. Now as there can be up to $\mathcal{O}(n)$ corrupt parties in $\mathcal{W}_q$, from the union bound, it follows that the probability that P_i outputs $s' \neq [s^\star]_q$ is at most $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$. Since there can be up to $\mathcal{O}(n)$ parties in $S_q \setminus \mathcal{W}_q$, it follows that except with probability at most $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, the output of every honest party in S_q is indeed $[s^\star]_q$. Now, there can be $|\mathbb{S}_{\mathcal{Z}_s}|$ possibilities for S_q. From the union bound, it follows that corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest party in S_q outputs $[s^\star]_q$, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

We next consider an *asynchronous* network and prove the analogue of Lemma C.1 by showing that if D is *honest*, then the parties eventually output $[s]$ except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$. Further, the adversary learns no additional information about s. The proof of the lemma follows closely the proof of Lemma C.1, except that we now rely on the properties of Π_{BC}, and ICP in the *asynchronous* network.

Lemma C.3. *If the network is asynchronous and* D *is honest, participating in* Π_{VSS} *with input* s*, then the honest parties eventually output* $[s]$ *except with a probability of* $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$*, with the view of the adversary remaining independent of* s*.*

Proof. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We note that $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$, since $Z^\star \in \mathcal{Z}_s$ as $\mathcal{Z}_a \subset \mathcal{Z}_s$. Since D is honest, it picks the shares $s_1, \ldots, s_{|\mathcal{Z}_s|}$ such that $s = s_1 + \cdots + s_{|\mathcal{Z}_s|}$ and sends s_q to each party $P_i \in S_q$, corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. These shares are eventually delivered. Now consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. After receiving the share s_{qi} from D, each party $P_i \in (S_q \cap \mathcal{H})$ starts giving $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$, where $s_{qi} = s_q$ holds, such that the linearity principle is followed while generating these IC-signatures. Then from the $\mathcal{Z}_a$*-correctness* of Π_{Auth} in the *asynchronous* network (Theorem 4.1), it follows that each party $P_i \in (S_q \cap \mathcal{H})$ will eventually receive $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj})$ from every $P_j \in (S_q \cap \mathcal{H})$, for every $P_k \in \mathcal{P}$, such that $s_{qj} = s_{qi} = s_q$ holds. Since S_q is arbitrary, it follows that eventually, every party $P_i \in \mathcal{H}$ broadcasts an $\mathsf{OK}(i, j)$ message, corresponding to every $P_j \in \mathcal{H}$. From the $\mathcal{Z}_a$*-weak validity* and $\mathcal{Z}_a$*-fallback validity* of Π_{BC} in the *asynchronous* network (Theorem 3.2), it follows that these $\mathsf{OK}(i, j)$ messages are eventually received by every party in $\mathcal{H}$. Consequently, the set $\mathcal{H}$ eventually constitutes a clique in the consistency graph of every party in $\mathcal{H}$. Now since the set $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$, it follows that eventually, there exists *some* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that D computes the core-sets $\{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and broadcast-set $\mathcal{BS}$, followed by broadcasting $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$. Moreover, since D is *honest*, it computes the sets $\{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}$ honestly, satisfying the following conditions, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If S_q constitutes a clique in the graph $G^{(\mathsf{D})}$, then $\mathcal{W}_q = S_q$.
- Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\mathsf{D})}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}_q = (S_p \cap S_q)$.
- Else $\mathcal{W}_q = S_q$ and $q \in \mathcal{BS}$.

Note that for each $\mathcal{W}_q$, the condition $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ holds. This is obviously true if $\mathcal{W}_q = (S_p \cap S_q)$, since in this case D also checks that $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ holds. On the other hand, even if $\mathcal{W}_q = S_q$, the condition $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ holds, since $\mathbb{Q}^{(1)}(S_q, \mathcal{Z}_s)$ holds. We also note that if $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}$ and consequently, D does not make public the share s_q. This is because the parties in $S_p \cap S_q$ will constitute a clique and the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition will be satisfied due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. Hence $\mathcal{W}_q$ will be set to $S_p \cap S_q$.

Since D eventually broadcasts $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$, from the $\mathcal{Z}_a$*-weak validity* and $\mathcal{Z}_a$*-fallback validity* of Π_{BC} in the *asynchronous* network, it follows that all the parties in $\mathcal{H}$ will eventually receive $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D. We next show that every $P_i \in \mathcal{H}$ will eventually include (D, S_p) to the set $\mathcal{C}_i$. For this, we need to show that all the conditions which hold for D in its graph $G^{(\mathsf{D})}$ when it broadcasts the CanCS message, are bound to eventually hold for every $P_i \in \mathcal{H}$. However, this simply follows from the $\mathcal{Z}_a$*-weak validity,* $\mathcal{Z}_a$*-fallback validity,* $\mathcal{Z}_a$*-weak consistency* and $\mathcal{Z}_a$*-fallback consistency* of Π_{BC} in the *asynchronous* network (see Theorem 3.2) and the fact that edges are added to consistency graphs, based on the receipt of $\mathsf{OK}(\star, \star)$ messages, which are broadcasted through various Π_{BC} instances. Consequently, any edge (i, j) which is included in $G^{(\mathsf{D})}$, is bound to be eventually included in the graph $G^{(i)}$ of every $P_i \in \mathcal{H}$.

Note that corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest $P_i \in \mathcal{W}_q$ will have the share s_q, which is either made public by D as part of the CanCS message or received from D. Next, we note that each $P_i \in \mathcal{H}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, s_q)$, corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ where $P_i \in \mathcal{W}_q$, from every $P_j \in \mathcal{W}_q$ and for every $P_k \in \mathcal{P}$. On the other hand, if any corrupt $P_j \in \mathcal{W}_q$ has $\mathsf{ICSig}(P_i, P_j, P_k, s'_q)$ of any honest $P_i \in \mathcal{W}_q$ for any $P_k \in \mathcal{P}$, then $s'_q = s_q$ holds. The proof for this is exactly the same as that of Lemma C.1.

We now show that every honest party $P_i \in S_q \setminus W_q$ eventually sets $[s]_q$ to s_q, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$. We first show that P_i sets $[s]_q$ to *some* value. Since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition, this means that $\mathcal{W}_q$ contains *at least* one *honest* party, say P_j. Since P_j follows the protocol steps honestly, it reveals $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$ of every $P_k \in \mathcal{W}_q$ to P_i. From the $\mathcal{Z}_a$-*correctness* property of ICP in the *asynchronous* network (see Theorem 4.1), it follows that P_i will eventually accept these signatures. On the other hand, even if $P_k \in \mathcal{W}_q$ is *corrupt*, then from the $\mathcal{Z}_a$-*non-repudiation* property of ICP in the *asynchronous* network (see Theorem 4.1), it follows that P_i eventually accepts $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, except with a probability ϵ_{ICP}. From the union bound, it follows that except with probability $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, each party $P_i \in S_q$ outputs some value for $[s]_q$. Further, this value must be s_q. The proof of this follows from what was shown in C.1. Now, there can be $|\mathbb{S}_{\mathcal{Z}_s}|$ possibilities for S_q. From the union bound, it follows that corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every honest party in S_q eventually outputs $[s]_q$, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

Finally, the privacy for s follows from the fact that throughout the protocol, the view of the adversary remains independent of the share s_q, corresponding to the group S_q, where $S_q = \mathcal{H}$. This is because as shown above, $q \notin \mathcal{BS}$ and consequently, D does not make public the share s_q. Moreover, during the pairwise consistency tests, the view of the adversary remains independent of s_q, when the parties in $\mathcal{H}$ exchange IC-signed s_q, which follows from the *privacy* property of ICP (see Theorem 4.1). Further, while computing the output, $\mathsf{ICSig}(P_k, P_j, P_i, s_q)$ is revealed by party $P_j \in W_q$ only to each party $P_i \in S_q \setminus W_q$. Hence, the adversary does not learn s_q.

Finally, we consider a *corrupt* D in the *asynchronous* network and prove the analogue of Lemma C.2, whose proof is very similar to that of Lemma C.2.

Lemma C.4. *If the network is asynchronous and* D *is corrupt and if any honest party computes an output, then there exists some* $s^\star \in \mathbb{F}$*, such that the honest parties eventually output* $[s^\star]$*, except with a probability of* $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

Proof. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Let $P_m \in \mathcal{H}$ be the *first* honest party, who computes an output in the protocol Π_{VSS}. This implies that there exists some $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that P_m receives a message $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D, and includes (D, S_p) to $\mathcal{C}_m$. This further implies that P_m has verified that the following hold, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If $q \notin \mathcal{BS}$, then $\mathcal{W}_q$ is either S_q or $(S_p \cap S_q)$. Moreover, $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition and the parties in $\mathcal{W}_q$ constitute a clique in the graph $G^{(m)}$.

– If $q \in \mathcal{BS}$, then D has made public s_q, as part of the CanCS message. Moreover, $\mathcal{W}_q$ is set to S_q.

From the $\mathcal{Z}_a$*-weak consistency* and $\mathcal{Z}_a$*-fallback consistency* of Π_{BC} in the *asynchronous* network (see Theorem 3.2), it follows that every party in $\mathcal{H}$ will eventually receive $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ from the broadcast of D. We next show that each $P_i \in \mathcal{H}$ will eventually include (D, S_p) to $\mathcal{C}_i$. For this, it is enough to show that all the edges which are present in $G^{(m)}$ when (D, S_p) is included in $\mathcal{C}_i$, are bound to be eventually present in the graph $G^{(i)}$ of every $P_i \in \mathcal{H}$. However, this simply follows from the $\mathcal{Z}_a$*-weak validity,* $\mathcal{Z}_a$*-fallback validity,* $\mathcal{Z}_a$*-weak consistency* and $\mathcal{Z}_a$*-fallback consistency* of Π_{BC} in the *asynchronous* network (see Theorem 3.2) and the fact that edges are added to consistency graphs, based on the receipt of $\mathsf{OK}(\star, \star)$ messages, which are broadcasted through various Π_{BC} instances.

Next, it can be shown that corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every $P_i \in \mathcal{W}_q$ will have a common share, say $s^\star_q$. The proof for this is the same as Lemma C.2. We define

$$s^\star \stackrel{def}{=} \sum_{q=1,\ldots,|\mathcal{Z}_s|} s^\star_q.$$

Now similar to the proof of Lemma C.2, it can be shown that each party P_i in $(\mathcal{H} \cap \mathcal{W}_q)$ will eventually have the required IC-signatures on $[s^\star]_q$ as part of $[s^\star]$ and will reveal these to parties in $S_q \setminus W_q$. Consequently, each party in $\mathcal{H}$ will eventually set $[s^\star]_q$ as the share corresponding to S_q and hence, $s^\star$ will eventually be secret-shared, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.

We next derive the communication complexity of the protocol Π_{VSS}.

Lemma C.5. *Protocol* Π_{VSS} *incurs a communication of* $\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ *bits.*

Proof. In the protocol, D needs to send the share s_q to *all* the parties in S_q. This incurs a total communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n \cdot \log |\mathbb{F}|)$ bits. There are $\mathcal{O}(|\mathcal{Z}_s| \cdot n^3)$ instances of Π_{Auth} invoked, to exchange IC-signed values, during the pairwise consistency tests. From Theorem 4.1, this incurs a total communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits. There are $\mathcal{O}(n^2)$ OK messages which need to be broadcasted, which from Theorem 3.2, incurs a total communication of $\mathcal{O}(n^6 \cdot \log n \cdot |\sigma|)$ bits, since each OK message encodes the identity of two parties, requiring $\log n$ bits. Finally, D needs to broadcast a candidate $(\mathsf{CanCS}, \mathsf{D}, S_p, \{\mathcal{W}_q\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}, \{s_q\}_{q \in \mathcal{BS}})$ message, where $S_p, \mathcal{BS}$ and each $\mathcal{W}_q$ can be represented by $\mathcal{O}(n)$ bits. And corresponding to the indices in $\mathcal{BS}$, the dealer D may end up broadcasting $\mathcal{O}(|\mathcal{Z}_s|)$ shares. From Theorem 3.2, this incurs a total communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot (n^5 \cdot |\sigma| + n^4 \cdot \log |\mathbb{F}| \cdot |\sigma|))$ bits. While computing the output, $\mathcal{O}(n^3 \cdot |\mathcal{Z}_s|)$ instances of Π_{Reveal} are involved, which incur a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^4 \cdot \log |\mathbb{F}|)$ bits.

Theorem 5.1 now follows from Lemma C.1-C.5.

Theorem 5.1. *Protocol Π_{VSS} achieves the following, except with a probability of $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$, where D has input $s \in \mathbb{F}$ for Π_{VSS} and where $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + T_{\mathsf{Reveal}}$.*

- *If D is honest, then the following hold.*
  - $\mathcal{Z}_s$**-correctness***: In a synchronous network, the honest parties output $[s]$ at time T_{VSS}.*
  - $\mathcal{Z}_a$**-correctness***: In an asynchronous network, the honest parties eventually output $[s]$.*
  - **Privacy***: Adversary's view remains independent of s in any network.*
- *If D is corrupt, then the following hold.*
  - $\mathcal{Z}_s$**-commitment***: In a synchronous network, either no honest party obtains any output or there exists some $s^\star \in \mathbb{F}$, such that the parties output $[s^\star]$. Moreover, if any honest party computes its output corresponding to $[s^\star]$ at time T, then all honest parties compute their output corresponding to $[s^\star]$ by time $T + \Delta$.*
  - $\mathcal{Z}_a$**-commitment***: In an asynchronous network, either no honest party obtains any output or there exists some $s^\star \in \mathbb{F}$, such that the honest parties eventually output $[s^\star]$.*
- **Communication Complexity***: $\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits are communicated by the honest parties.*

D Network Agnostic Reconstruction Protocols and Secure Multicast

This section presents our network-agnostic reconstruction protocols and secure multicast protocol, along with their properties. We start with the protocol Π_{RecShare} for reconstructing a designated share, presented in Fig 18.

Protocol $\Pi_{\mathsf{RecShare}}([s], S_q, \mathcal{R})$

- **Sending IC-signed Share to the Parties**: If $P_i \in \mathcal{W}_q$, then reveal $\mathsf{ICSig}(P_j, P_i, P_k, [s]_q)$ of every $P_j \in \mathcal{W}_q$ to every $P_k \in \mathcal{R} \setminus \mathcal{W}_q$. Here $\mathcal{W}_q$ denotes the publicly known core-set corresponding to $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, as part of $[s]$.
- **Computing Output**: If $P_i \in (\mathcal{R} \cap \mathcal{W}_q)$, then output $[s]_q$. Else if $P_i \in \mathcal{R} \setminus \mathcal{W}_q$, then check if there exists any $P_j \in \mathcal{W}_q$ and a value s_{qj}, such that P_i has accepted $\mathsf{ICSig}(P_k, P_j, P_i, s_{qj})$ of every $P_k \in \mathcal{W}_q$. Upon finding such a P_j, output $[s]_q = s_{qj}$.[a]

[a] If there are multiple such parties P_j, then consider the one with the smallest index.

Fig. 18: Network agnostic reconstruction protocol to reconstruct a designated share $[s]_q$. The above code is executed by each $P_i \in \mathcal{P}$.

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

Lemma 6.1. *Let s be a value which is linearly secret-shared with IC sig-*

natures, let $S_q \in \mathbb{S}_{\mathcal{Z}_s}$ *be a designated set and let* $\mathcal{R} \subseteq \mathcal{P}$ *be a designated set of receivers. Then protocol* Π_{RecShare} *achieves the following.*

- $\mathcal{Z}_s$**-correctness**: *In a synchronous network, all honest parties in* $\mathcal{R}$ *output* $[s]_q$ *at time* $T_{\mathsf{RecShare}} = T_{\mathsf{Reveal}}$*, except with a probability of* $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$*.*
- $\mathcal{Z}_a$**-correctness**: *In an asynchronous network, all honest parties in* $\mathcal{R}$ *eventually output* $[s]_q$*, except with a probability of* $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$*.*
- **Privacy**: *If* $\mathcal{R}$ *consists of only honest parties, then the view of the adversary remains independent of* $[s]_q$*.*
- **Communication Complexity**: $\mathcal{O}(|\mathcal{R}| \cdot n^3 \cdot \log |\mathbb{F}|)$ *bits are communicated.*

Proof. We first note that all honest parties in $(\mathcal{R} \cap \mathcal{W}_q)$ output $[s]_q$ correctly. So consider an *arbitrary* honest $P_i \in \mathcal{R} \setminus \mathcal{W}_q$. We first show that P_i indeed computes an output in the protocol, irrespective of the network type.

Since $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_q, \mathcal{Z}_s)$ condition, it contains *at least* one *honest* party, say P_j. Since P_j follows the protocol steps honestly, it reveals $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$ of every $P_k \in \mathcal{W}_q$ to P_i. From the *correctness* properties of ICP (see Theorem 4.1), it follows that P_i will accept the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, revealed by P_j, after time T_{Reveal} in a *synchronous* network, or eventually in an *asynchronous* network. On the other hand, even if $P_k \in \mathcal{W}_q$ is *corrupt*, then also from the *non-repudiation* properties of ICP, it follows that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, except with a probability ϵ_{ICP}, after time T_{Reveal} in a *synchronous* network, or eventually in an *asynchronous* network. As there can be $\mathcal{O}(n)$ *corrupt* parties in $\mathcal{W}_q$, from the union bound, it follows that except with a probability $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$, party P_i will find a candidate party from $\mathcal{W}_q$, who reveals $[s]_q$, along with the IC-signature of all the parties in $\mathcal{W}_q$, after time T_{Reveal} in a *synchronous* network, or eventually in an *asynchronous* network. This is because the *honest* party in $\mathcal{W}_q$ always constitutes a candidate party. Now as there can be $\mathcal{O}(n)$ parties in $\mathcal{R} \setminus \mathcal{W}_q$, it follows that except with probability $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, every honest party in $\mathcal{R} \setminus \mathcal{W}_q$ will find a candidate party from $\mathcal{W}_q$, who reveals $[s]_q$, along with the IC-signature of all the parties in $\mathcal{W}_q$, after time T_{Reveal} in a *synchronous* network, or eventually in an *asynchronous* network. Hence all honest parties in $\mathcal{R}$ compute an output, after time T_{Reveal} in a *synchronous* network, or eventually in an *asynchronous* network, except with a probability $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$.

We next show that the output computed by all the honest parties is indeed correct. While this is trivially true for the parties in $(\mathcal{R} \cap \mathcal{W}_q)$, consider an arbitrary *honest* party $P_i \in \mathcal{R} \setminus \mathcal{W}_q$. The above argument shows that P_i computes an output in the protocol, irrespective of the network type. So let P_i output s'. We wish to show that $s' = [s]_q$. From the protocol steps, since P_i outputs s', it implies that there exists some $P_j \in \mathcal{W}_q$, such that P_i has accepted $\mathsf{ICSig}(P_k, P_j, P_i, s')$ of every $P_k \in \mathcal{W}_q$, revealed by P_j. If P_j is *honest*, then indeed $s' = [s]_q$, as one of the IC-signatures $\mathsf{ICSig}(P_k, P_j, P_i, s')$ is the same as $\mathsf{ICSig}(P_k, P_j, P_i, [s]_q)$, corresponding to the *honest* $P_k \in \mathcal{W}_q$, which is guaranteed to exist. So consider the case when P_j is *corrupt*. Moreover, let $P_k \in \mathcal{W}_q$ be an *honest* party (which is guaranteed to exist). In order that $s' \neq [s]_q$, it must be the case that P_i accepts $\mathsf{ICSig}(P_k, P_j, P_i, s')$, revealed by P_j. However, from the *unforgeability* property of

ICP (see Theorem 4.1), this can happen only with probability ϵ_{ICP}. Now as there can be up to $\mathcal{O}(n)$ corrupt parties in $\mathcal{W}_q$, from the union bound, it follows that the probability that P_i outputs $s' \neq [s]_q$ is at most $\mathcal{O}(n \cdot \epsilon_{\mathsf{ICP}})$. And since there can be up to $\mathcal{O}(n)$ parties in $\mathcal{R} \setminus \mathcal{W}_q$, it follows that except with probability at most $\mathcal{O}(n^2 \cdot \epsilon_{\mathsf{ICP}})$, the output of every honest party in $\mathcal{R}$ is indeed $[s]_q$.

Communication complexity follows from the communication complexity of Π_{Reveal} (Theorem 4.1) and the fact that $\mathcal{O}(|\mathcal{R}| \cdot n^2)$ instances of Π_{Reveal} are involved. And privacy follows from the privacy of ICP.

Protocol Π_{Rec} for reconstructing s by a designated set of receivers is presented in Fig 19.

Protocol $\Pi_{\mathsf{Rec}}([s], \mathcal{R})$

- **Reconstructing Individual Shares**: Corresponding to each $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, participate in an instance $\Pi_{\mathsf{RecShare}}([s], S_q, \mathcal{R})$ of Π_{RecShare}, to let the parties in $\mathcal{R}$ reconstruct $[s]_q$.
- **Computing Output**: If $P_i \in \mathcal{R}$, then output $s = \sum_{S_q \in \mathbb{S}} [s]_q$.

Fig. 19: Network agnostic reconstruction protocol to reconstruct a secret-shared value with IC signatures. The above code is executed by each party $P_i \in \mathcal{P}$.

The properties of the protocol Π_{Rec} are stated in Lemma 6.2.

Lemma 6.2. *Let s be a value which is linearly secret-shared with IC signatures and let $\mathcal{R} \subseteq \mathcal{P}$ be a set of designated receivers. Then protocol Π_{Rec} achieves the following.*

- **$\mathcal{Z}_s$-correctness**: *In a synchronous network, all honest parties in $\mathcal{R}$ output s at time $T_{\mathsf{Rec}} = T_{\mathsf{RecShare}}$, except with probability $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- **$\mathcal{Z}_a$-correctness**: *In an asynchronous network, all honest parties in $\mathcal{R}$ eventually output s, except with probability $\mathcal{O}(|\mathbb{S}_{\mathcal{Z}_s}| \cdot n^2 \cdot \epsilon_{\mathsf{ICP}})$.*
- **Privacy**: *If $\mathcal{R}$ consists of only honest parties, then the view of the adversary remains independent of s.*
- **Communication Complexity**: *$\mathcal{O}(|\mathcal{Z}_s| \cdot |\mathcal{R}| \cdot n^3 \cdot \log |\mathbb{F}|)$ bits are communicated.*

Proof. The proof follows from Lemma 6.1, and the fact that $|\mathbb{S}_{\mathcal{Z}_s}|$ instances of Π_{RecShare} are invoked.

D.1 Network Agnostic Secure Multicast

Protocol Π_{SVM} is presented in Fig 20.

Protocol $\Pi_{\mathsf{SVM}}(\mathsf{Sen}, v, \mathcal{R})$

- **Sending the Value to the Parties**: Sen on having the input v, invokes an instance of Π_{VSS} with input v and the parties in $\mathcal{P}$ participates in this instance.

– **Verifying if Sender has Committed Any Value**: Each $P_i \in \mathcal{P}$ waits till its local time becomes T_{VSS}, initializes a Boolean variable $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 0 and then do the following.
  – Upon computing an output in the Π_{VSS} instance, set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1.
  – Upon setting $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1, participate in an instance $\Pi_{\mathsf{Rec}}([v], \mathcal{R})$ of Π_{Rec} to let the parties in $\mathcal{R}$ reconstruct v
– **Computing Output**: Each $P_i \in \mathcal{R}$ upon computing an output v during the instance $\Pi_{\mathsf{Rec}}([v], \mathcal{R})$, outputs v.

Fig. 20: The network agnostic SVM protocol

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

Lemma 6.3. *Protocol Π_{SVM} achieves the following, where* Sen *participates with input v and where each honest party initializes $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 0.*
– **Synchronous Network**: *If* Sen *is honest, then all honest parties set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 at time T_{VSS} and except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ output v, after time $T_{\mathsf{SVM}} = T_{\mathsf{VSS}} + T_{\mathsf{Rec}}$. Moreover, if $\mathcal{R}$ consists of only honest parties, then the view of* Adv *remains independent of v. If* Sen *is corrupt and some honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1, then there exists some $v^\star$ such that, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ output $v^\star$. Moreover, if any honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 at time T, then all honest parties in $\mathcal{R}$ output $v^\star$ by time $T + 2\Delta$.*
– **Asynchronous Network**: *If* Sen *is honest, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ eventually output v. Moreover, if $\mathcal{R}$ consists of only honest parties, then the view of the adversary remains independent of v. If* Sen *is corrupt and some honest party sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1, then there exists some $v^\star$ such that, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties in $\mathcal{R}$ eventually output $v^\star$.*
– **Communication Complexity**: *$\mathcal{O}(|\mathcal{Z}_s| \cdot n^8 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits are communicated.*

Proof. Let us first consider an *honest* Sen. If the network is *synchronous*, then from the *$\mathcal{Z}_s$-correctness* of Π_{VSS} in the *synchronous* network (Theorem 5.1), at time T_{VSS}, all *honest* parties will output $[v]$. Consequently, each honest party will set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1 and start participating in the instance of Π_{Rec}. Hence, from the *$\mathcal{Z}_s$-correctness* of Π_{Rec} in the *synchronous* network (Lemma 6.2), coupled with the modifications presented in Section 6.1, it follows that all honest parties in $\mathcal{R}$ output v, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, after time $T_{\mathsf{SVM}} = T_{\mathsf{VSS}} + T_{\mathsf{Rec}}$. The privacy of v follows from the *privacy* of Π_{VSS} (Theorem 5.1) and *privacy* of Π_{Rec} (Lemma 6.2). The proof for the case of honest Sen in an *asynchronous* network is the same as above, except that we now rely on the *$\mathcal{Z}_a$-correctness* of Π_{VSS} in the *asynchronous* network (Theorem 5.1) and the *$\mathcal{Z}_a$-correctness* of Π_{Rec} in the *asynchronous* network (Lemma 6.2).

Next consider a *corrupt* Sen. Let us first consider a *synchronous* network. Let P_i be the *first* honest party who sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1. This implies that there exists some $v^\star$, such that P_i outputs $[v^\star]$. Let T be the time when P_i outputs $[v^\star]$ during the instance of Π_{VSS} (and hence sets $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1). From the

$\mathcal{Z}_s$-*commitment* of Π_{VSS} in the *synchronous* network, it follows that *all* honest parties will output $[v^\star]$ (and hence set $\mathsf{flag}^{(\mathsf{Sen},\mathcal{R})}$ to 1), latest by time $T + \Delta$. Hence all honest parties will start participating in the instance of Π_{Rec}, latest by time $T + \Delta$. Hence, from the $\mathcal{Z}_s$-*correctness* of Π_{Rec} in the *synchronous* network (Lemma 6.2), coupled with the modifications presented in Section 6.1, it follows that all honest parties in $\mathcal{R}$ output $v^\star$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, by time $T + 2\Delta$.

The proof for the case of a *corrupt* Sen in an *asynchronous* network is the same as above, except that we now rely on the $\mathcal{Z}_a$-*commitment* of Π_{VSS} in the *asynchronous* network (Theorem 5.1) and the $\mathcal{Z}_a$-*correctness* of Π_{Rec} in the *asynchronous* network (Lemma 6.2).

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

E Properties of the Protocol Π_{MDVSS}

In this section, we prove the properties of the protocol Π_{MDVSS} (see Fig 8 for the formal description).

We first show that if the network is *synchronous*, then all honest parties will compute a *common* candidate set of committed dealers $\mathcal{CD}$ set by time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$, such that *all* honest dealers are guaranteed to be present in $\mathcal{CD}$.

Lemma E.1. *If the network is synchronous and $P_\ell \in \mathcal{P}$ is an honest dealer participating with input $s^{(\ell)}$, then all the following hold in Π_{MDVSS}, where $\mathcal{H}$ is the set of honest parties.*

- *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all the parties in $\mathcal{H}$ will have a common $\mathcal{CD}$ set by time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$, where $\mathcal{H} \subseteq \mathcal{CD}$.* [18]
- *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, corresponding to every dealer $P_\ell \in \mathcal{CD}$ and every $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$, every party in $(\mathcal{H} \cap S_q)$ will have a common share, say ${s^\star}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$, for an honest P_ℓ.*

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. From the properties of Π_{SVM} in the *synchronous* network (Lemma 6.3), it follows that every $P_i \in \mathcal{H}$ will set $\mathsf{flag}^{(P_\ell,S_q)}$ to 1 at time T_{VSS}, corresponding to every $P_\ell \in \mathcal{H}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. Moreover, at time T_{SVM}, corresponding to every $P_\ell \in \mathcal{H}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, each $P_i \in (\mathcal{H} \cap S_q)$ computes the output $s_{qi}^{(\ell)}$ during the instance $\Pi_{\mathsf{SVM}}^{(P_\ell,S_q)}$. Furthermore, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the value $s_{qi}^{(\ell)}$ will be the same as $s_q^{(\ell)}$. Hence, corresponding to every $P_\ell \in \mathcal{H}$, every $P_i \in \mathcal{H}$ will start participating with input 1 during the instance $\Pi_{\mathsf{BA}}^{(\ell)}$, at time T_{SVM}. Hence from the $\mathcal{Z}_s$-*validity* of Π_{BA} in the *synchronous* network (Theorem 3.1), at time $T_{\mathsf{SVM}} + T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ will obtain the output 1 from the $\Pi_{\mathsf{BA}}^{(\ell)}$ instances, corresponding to each $P_\ell \in \mathcal{H}$. Since $\mathcal{P} \setminus \mathcal{H} \in \mathcal{Z}_s$, it follows that at time $T_{\mathsf{SVM}} + T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ start

[18] This automatically implies that $\mathcal{P} \setminus \mathcal{CD} \in \mathcal{Z}_s$.

participating with input 0 in any remaining instance $\Pi_{\mathsf{BA}}^{(\star)}$ of Π_{BA}, for which no input is provided yet. Hence from the *$\mathcal{Z}_s$-security* of Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ will compute some output in all the n instances of $\Pi_{\mathsf{BA}}^{(\star)}$ by time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$. Moreover, the outputs will be common for the parties in $\mathcal{H}$. Consequently, all the parties in $\mathcal{H}$ will have a common $\mathcal{CD}$ set at the time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$. Moreover, $\mathcal{H} \subseteq \mathcal{CD}$, since $\mathcal{CD}$ includes all the dealers P_ℓ such that $\Pi_{\mathsf{BA}}^{(\ell)}$ outputs 1. And as shown above the $\Pi_{\mathsf{BA}}^{(\ell)}$ instances corresponding to $P_\ell \in \mathcal{H}$ outputs 1.

Next, consider an *arbitrary* $P_\ell \in \mathcal{CD}$. This implies that at time $T_{\mathsf{SVM}} + T_{\mathsf{BA}}$, at least one party from $\mathcal{H}$, say P_k, has participated with input 1 during the instance $\Pi_{\mathsf{BA}}^{(\ell)}$. If not, then from the *$\mathcal{Z}_s$-validity* of Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ would have obtained the output 0 from the instance $\Pi_{\mathsf{BA}}^{(\ell)}$ at the time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$ and hence $P_\ell \notin \mathcal{CD}$, which is a *contradiction*. This implies that by the time $T_{\mathsf{SVM}} + T_{\mathsf{BA}}$, party P_k has set $\mathsf{flag}^{(P_\ell, S_q)}$ to 1 during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$, for $q = 1, \ldots, |\mathcal{Z}_s|$. So consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. From the properties of Π_{SVM} in the *synchronous* network (Lemma 6.3), it follows that there exists some value ${s^\star}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$ for an *honest* P_ℓ, such that except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all the parties in $\mathcal{H}$ output ${s^\star}_q^{(\ell)}$ during the instance $\Pi_{\mathsf{SVM}}^{(P_\ell, S_q)}$ by time $T_{\mathsf{SVM}} + T_{\mathsf{BA}} + 2\Delta < T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$.

We next show that if the network is *synchronous* and if an *honest* dealer from $\mathcal{CD}$ broadcasts any set of candidate core-sets, then all honest parties will "accept" the core-sets. Moreover, the dealer will *never* make public the share corresponding to the group from $\mathbb{S}_{\mathcal{Z}_s}$ consisting of only honest parties while making public these core-sets. Furthermore, each honest dealer in $\mathcal{CD}$ will start making public *at least* one candidate set of core-sets, namely the one computed with respect to the group from $\mathbb{S}_{\mathcal{Z}_s}$, consisting of only honest parties. A consequence of all these properties is that if the dealer is honest, the adversary will not learn any information about the dealer's input.

Lemma E.2. *If the network is synchronous and $P_\ell \in \mathcal{CD}$ is an honest dealer participating with input $s^{(\ell)}$, then all the following hold in Π_{MDVSS} except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, where $\mathcal{H}$ is the set of honest parties.*

- *If $S_p = \mathcal{H}$, then P_ℓ will broadcast $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$.*
- *If P_ℓ broadcasts any $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ at time T, then every honest $P_i \in \mathcal{P}$ will include (P_ℓ, S_p) to $\mathcal{C}_i$ at time $T + T_{\mathsf{BC}}$. Moreover, the following will hold.*
  - *If $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}_p^{(\ell)}$.*
  - *For $q = 1, \ldots, |\mathcal{Z}_s|$, each $\mathcal{W}_{p,q}^{(\ell)}$ will be either S_q or $(S_p \cap S_q)$. Moreover, $\mathcal{Z}_s$ will satisfy the $\mathbb{Q}^{(1)}(\mathcal{W}_{p,q}^{(\ell)}, \mathcal{Z}_s)$ condition.*
  - *If $q \notin \mathcal{BS}_p^{(\ell)}$, then every honest $P_i \in S_q$ will have the share $s_q^{(\ell)}$. Moreover, every honest $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, s_q^{(\ell)})$ of every $P_j \in$*

$\mathcal{W}_{p,q}^{(\ell)}$ *for every* $P_k \in \mathcal{P}$*. Furthermore, if any corrupt* $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ *have* $\mathsf{ICSig}(P_i, P_j, P_k, s_q'^{(\ell)})$ *of any honest* $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ *for any* $P_k \in \mathcal{P}$*, then* $s_q'^{(\ell)} = s_q^{(\ell)}$ *holds. Also, all the underlying IC-signatures will satisfy the linearity property.*

– *The view of the adversary will be independent of* $s^{(\ell)}$*.*

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Since the dealer P_ℓ is *honest*, from Lemma E.1 it follows that corresponding to every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, every party in $(\mathcal{H} \cap S_q)$ will have the share $s_{qi}^{(\ell)}$, by time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, where $s_{qi}^{(\ell)} = s_q^{(\ell)}$. Now consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. At time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$, each party $P_i \in (S_q \cap \mathcal{H})$ starts giving $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi}^{(\ell)})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$, where $s_{qi}^{(\ell)} = s_q^{(\ell)}$ holds. Then from the $\mathcal{Z}_s$*-correctness* of Π_{Auth} in the *synchronous* network (Theorem 4.1), it follows that at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}}$, each party $P_i \in (S_q \cap \mathcal{H})$ will receive $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj}^{(\ell)})$ from every $P_j \in (S_q \cap \mathcal{H})$, for every $P_k \in \mathcal{P}$, such that $s_{qj}^{(\ell)} = s_{qi}^{(\ell)} = s_q^{(\ell)}$ holds. Since S_q is arbitrary, it follows that at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}}$, every party $P_i \in \mathcal{H}$ broadcasts an $\mathsf{OK}^{(\ell)}(i, j)$ message, corresponding to every $P_j \in \mathcal{H}$. From the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network (Theorem 3.2), it follows that these $\mathsf{OK}^{(\ell)}(i, j)$ messages are received by every party in $\mathcal{H}$ through *regular-mode* at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. Consequently, the set $\mathcal{H}$ constitutes a clique in the consistency graph $G^{(\ell,i)}$ of every party $P_i \in \mathcal{H}$ at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. Note that the set $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$. Let S_p be the set from $\mathbb{S}_{\mathcal{Z}_s}$, such that $S_p = \mathcal{H}$. From the protocol steps, it then follows that at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$, the dealer P_ℓ will compute core-sets $\mathcal{W}_{p,q}^{(\ell)}$ for $q = 1, \ldots, |\mathcal{Z}_s|$ and broadcast-set $\mathcal{BS}_p^{(\ell)}$ with respect to S_p as follows.

– If S_q constitutes a clique in the graph $G^{(\ell,\ell)}$, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as S_q.
– Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,\ell)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as $(S_p \cap S_q)$.
– Else $\mathcal{W}_{p,q}^{(\ell)}$ is set to S_q and q is included to $\mathcal{BS}_p^{(\ell)}$.

After computing the core-sets and broadcast-set, P_ℓ will broadcast $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. This proves the first part of the lemma.

We next proceed to prove the second part of the lemma. So consider an *arbitrary* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that P_ℓ compute core-sets $\mathcal{W}_{p,q}^{(\ell)}$ for $q = 1, \ldots, |\mathcal{Z}_s|$ and broadcast-set $\mathcal{BS}_p^{(\ell)}$ with respect to S_p and broadcasts $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ at time T. This means at time T, the parties in S_p constitute a clique in the graph $G^{(\ell,\ell)}$. We also note that $T \geq T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + T_{\mathsf{BC}}$. This is because any instance of Π_{BC} takes at least T_{BC} time in a *synchronous* network to generate an output. And the parties in $\mathcal{H}$ start participating in any Π_{BC} instance invoked for broadcasting any $\mathsf{OK}^{(\ell)}(\star, \star)$ message, only *after* time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}}$. Consequently, any

$\mathsf{OK}^{(\ell)}(\star,\star)$ message received by P_ℓ, must be *after* time $T_{\mathsf{SVM}}+2T_{\mathsf{BA}}+T_{\mathsf{Auth}}+T_{\mathsf{BC}}$. We also note that any edge (P_j,P_k) which is present in the graph $G^{(\ell,\ell)}$ of P_ℓ at time T, is bound to be present in the graph $G^{(\ell,i)}$ of every $P_i \in \mathcal{H}$, latest by time $T+\Delta$. This is because the edge (P_j,P_k) is added to $G^{(\ell,\ell)}$ upon the receipt of $\mathsf{OK}^{(\ell)}(j,k)$ and $\mathsf{OK}^{(\ell)}(k,j)$ messages from the broadcast of P_j and P_k respectively. And from the $\mathcal{Z}_s$*-validity,* $\mathcal{Z}_s$*-consistency* and $\mathcal{Z}_s$*-fallback consistency* of Π_{BC} in the *synchronous* network, these $\mathsf{OK}^{(\ell)}(\star,\star)$ messages will be received by every party $P_i \in \mathcal{H}$, latest by time $T+\Delta$. Since P_ℓ is assumed to be *honest*, it follows that the sets $\{\mathcal{W}^{(\ell)}_{p,q}\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}^{(\ell)}_p$ satisfy the following properties.

- If S_q constitutes a clique in the graph $G^{(\ell,\ell)}$, then $\mathcal{W}^{(\ell)}_{p,q}$ is set as S_q.
- Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,\ell)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}^{(\ell)}_{p,q}$ is set as $(S_p \cap S_q)$.
- Else $\mathcal{W}^{(\ell)}_{p,q}$ is set to S_q and q is included to $\mathcal{BS}^{(\ell)}_p$.

We also note that if $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}^{(\ell)}_p$ and consequently, P_ℓ will *not* make the share $s^{(\ell)}_q$ public. This is because $T \geq T_{\mathsf{SVM}}+2T_{\mathsf{BA}}+T_{\mathsf{Auth}}+T_{\mathsf{BC}}$. And as shown in the proof of the first part, the set $\mathcal{H}$ will constitute a clique in the graph $G^{(\ell,\ell)}$ at time $T_{\mathsf{SVM}}+2T_{\mathsf{BA}}+T_{\mathsf{Auth}}+T_{\mathsf{BC}}$. Since P_ℓ is *honest*, from the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network, it follows that all the parties in $\mathcal{H}$ will receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}^{(\ell)}_{p,q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_p, \{s^{(\ell)}_q\}_{q\in\mathcal{BS}^{(\ell)}_p})$ through the regular-mode at time $T+T_{\mathsf{BC}}$. Moreover, each party $P_i \in \mathcal{H}$ will include (P_ℓ, S_p) to the set $\mathcal{C}_i$ at time $T+T_{\mathsf{BC}}$. This is because since P_ℓ has computed the sets $\{\mathcal{W}^{(\ell)}_{p,q}\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}^{(\ell)}_p$ *honestly*, these sets will pass all the verifications for each $P_i \in \mathcal{H}$ at time $T+\Delta$.

Next consider an *arbitrary* $q \notin \mathcal{BS}^{(\ell)}_p$. This implies that P_ℓ has set $\mathcal{W}^{(\ell)}_{p,q}$ as $(S_p \cap S_q)$ because the parties in $(S_p \cap S_q)$ constitutes a clique in the graph $G^{(\ell,\ell)}$. Now consider an *arbitrary* $P_i \in (\mathcal{H} \cap S_q)$. This implies that P_i has computed $s^{(\ell)}_{qi}$ during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s^{(\ell)}_q, S_q)$ at time T_{SVM}, which will be the same as $s^{(\ell)}_q$, since P_ℓ is *honest*. Next consider *arbitrary* $P_i, P_j \in \mathcal{W}^{(\ell)}_{p,q}$, such that $P_j \neq P_i$. This implies that the edge (i,j) is present in the graph $G^{(\ell,\ell)}$, which further implies that P_i has broadcasted the message $\mathsf{OK}^{(\ell)}(i,j)$. This further implies that P_i must have received $\mathsf{ICSig}(P_j, P_i, P_k, s^{(\ell)}_{qj})$ from P_j, for every $P_k \in \mathcal{P}$, such that $s^{(\ell)}_{qj} = s^{(\ell)}_{qi}$ holds. Since $s^{(\ell)}_{qi} = s^{(\ell)}_q$, it follows that $\mathsf{ICSig}(P_j, P_i, P_k, s^{(\ell)}_{qj})$ is the same as $\mathsf{ICSig}(P_j, P_i, P_k, s^{(\ell)}_q)$. On the other hand, consider an *arbitrary* $P_j \in \mathcal{W}^{(\ell)}_{p,q}$, such that P_j is *corrupt* and where P_j has received $\mathsf{ICSig}(P_i, P_j, P_k, s'^{(\ell)}_q)$ from P_i, for any $P_k \in \mathcal{P}$. Then from the protocol steps, it follows that $s'^{(\ell)}_q = s^{(\ell)}_{qi}$, since P_i gives the IC-signature on the share $s^{(\ell)}_{qi}$, received from P_ℓ. And since $s^{(\ell)}_{qi} = s^{(\ell)}_q$, it follows that $\mathsf{ICSig}(P_i, P_j, P_k, s'^{(\ell)}_q)$ is the same as $\mathsf{ICSig}(P_i, P_j, P_k, s^{(\ell)}_q)$. The linearity of the underlying IC-signatures follow from the fact the parties follow the linearity principle while generating IC-signatures.

Finally, the privacy of $s^{(\ell)}$ follows from the fact that throughout the protocol, adversary *does not* learn anything about the share $s^{(\ell)}_q$, provided $S_q = \mathcal{H}$.

Namely, during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$ where $(S_q \cap Z^\star) = \emptyset$, the view of the adversary remains independent of $s_q^{(\ell)}$, which follows from the privacy of Π_{SVM} (Lemma 6.3). Moreover, as shown above, P_ℓ never makes public the share $s_q^{(\ell)}$, as $q \notin \mathcal{BS}_p^{(\ell)}$. Furthermore, since the set $\mathcal{H}$ will consists of *only* honest parties, from the *privacy* of ICP (see Theorem 4.1), it follows that the adversary does not learn any additional information about $s_q^{(\ell)}$, when the parties in $\mathcal{H}$ exchange IC-signed $s_q^{(\ell)}$ during the pairwise consistency tests.

We next show that if the network is *synchronous*, then any candidate set of core-sets "accepted" on the behalf of a *corrupt* dealer by any *honest* party at the time T, is bound to be accepted by all honest parties, latest by time $T + \Delta$.

Lemma E.3. *If the network is synchronous and if in* Π_{MDVSS} *any honest party* P_i *receives* $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ *from the broadcast of any corrupt dealer* $P_\ell \in \mathcal{CD}$ *and includes* (P_ℓ, S_p) *to* $\mathcal{C}_i$ *at time* T*, then all honest parties* P_j *will receive* $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ *from the broadcast of* P_ℓ *and include* (P_ℓ, S_p) *to* $\mathcal{C}_j$ *by time* $T + \Delta$*. Moreover, for* $q = 1, \ldots, |\mathcal{Z}_s|$*, the following holds, except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*.*

- $\mathcal{W}_{p,q}^{(\ell)}$ *is either* S_q *or* $(S_p \cap S_q)$*. Moreover,* $\mathcal{Z}_s$ *satisfies the* $\mathbb{Q}^{(1)}(\mathcal{W}_{p,q}^{(\ell)}, \mathcal{Z}_s)$ *condition.*
- *If* $q \notin \mathcal{BS}_p^{(\ell)}$*, then every honest* $P_i \in S_q$ *will have a common share, say* ${s^\star}_q^{(\ell)}$*. Moreover, every honest* $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ *will have* $\mathsf{ICSig}(P_j, P_i, P_k, {s^\star}_q^{(\ell)})$ *of every* $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ *and for every* $P_k \in \mathcal{P}$*. Furthermore, if any corrupt* $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ *has* $\mathsf{ICSig}(P_i, P_j, P_k, {s'}_q^{(\ell)})$ *of any honest* $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ *for any* $P_k \in \mathcal{P}$*, then* ${s'}_q^{(\ell)} = {s^\star}_q^{(\ell)}$ *holds. Also, all the underlying IC-signatures will satisfy the linearity principle.*

Proof. The proof follows very closely the proof of Lemma E.2. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Now consider an *arbitrary corrupt* dealer $P_\ell \in \mathcal{CD}$ and an *arbitrary* $P_i \in \mathcal{H}$, such that P_i receives $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ and includes (P_ℓ, S_p) to $\mathcal{C}_i$ at time T. Now consider another *arbitrary* $P_j \in \mathcal{H}$, such that $P_j \neq P_i$. From the $\mathcal{Z}_s$-*consistency* and $\mathcal{Z}_s$-*fallback consistency* of Π_{BC} in the *synchronous* network, it follows that P_j is bound to receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ, latest by time $T + \Delta$. We wish to show that P_j will include (P_ℓ, S_p) to $\mathcal{C}_j$, by time $T + \Delta$. For this, we note that since P_i has included (P_ℓ, S_p) to $\mathcal{C}_i$ at time T, *all* the following conditions hold for P_i at time T, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If $q \in \mathcal{BS}_p^{(\ell)}$, then the set $\mathcal{W}_{p,q}^{(\ell)} = S_q$.
- If $(q \notin \mathcal{BS}_p^{(\ell)})$, then $\mathcal{W}_{p,q}^{(\ell)}$ is either S_q or $(S_p \cap S_q)$, such that:
  - If $\mathcal{W}_{p,q}^{(\ell)} = S_q$, then S_q constitutes a clique in $G^{(\ell,i)}$.

– Else if $\mathcal{W}^{(\ell)}_{p,q} = (S_p \cap S_q)$, then $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,i)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition.

We claim that all the above conditions will hold *even* for P_j by time $T + \Delta$. This is because *all* the edges which are present in the consistency graph $G^{(\ell,i)}$ at time T are bound to be present in the consistency graph $G^{(\ell,j)}$ by time $T + \Delta$. This follows from the $\mathcal{Z}_s$*-validity,* $\mathcal{Z}_s$*-consistency* and $\mathcal{Z}_s$*-fallback consistency* of Π_{BC} in the *synchronous* network (see Theorem 3.2) and the fact that the edges in the graph $G^{(\ell,i)}$ are based on $\mathsf{OK}^{(\ell)}(\star,\star)$ messages, which are received through various Π_{BC} instances.

Next consider an *arbitrary* $q \notin \mathcal{BS}^{(\ell)}_p$. This implies that $\mathcal{W}^{(\ell)}_{p,q}$ is set as $(S_p \cap S_q)$ and all the parties in $(S_p \cap S_q)$ constitute a clique in the consistency graph of every party in $\mathcal{H}$. From the properties of Π_{SVM} in the *synchronous* network, all honest parties in S_q compute a common output, say ${s^{\star}}^{(\ell)}_q$, during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s^{(\ell)}_q, S_q)$. Next consider an *arbitrary* $P_i \in (\mathcal{H} \cap \mathcal{W}^{(\ell)}_{p,q})$ and any *arbitrary* $P_j \in \mathcal{W}^{(\ell)}_{p,q}$. This implies that P_i has broadcasted the message $\mathsf{OK}^{(\ell)}(i,j)$, after receiving $\mathsf{ICSig}(P_j, P_i, P_k, s^{(\ell)}_{qj})$ from P_j, for every $P_k \in \mathcal{P}$, and verifying that the share $s^{(\ell)}_{qj}$ is the *same* as the one, computed during the instance $\Pi^{(P_\ell, s^{(\ell)}_q, S_q)}_{\mathsf{SVM}}$. Since the share computed by P_i during $\Pi_{\mathsf{SVM}}(P_\ell, s^{(\ell)}_q, S_q)$ is ${s^{\star}}^{(\ell)}_q$, it follows that $\mathsf{ICSig}(P_j, P_i, P_k, s^{(\ell)}_{qj})$ is the same as $\mathsf{ICSig}(P_j, P_i, P_k, {s^{\star}}^{(\ell)}_q)$. On the other hand, since P_i gives its IC-signature on ${s^{\star}}^{(\ell)}_q$ to *every* $P_j \in S_q$, it follows that if any *corrupt* $P_j \in \mathcal{W}^{(\ell)}_{p,q}$ has $\mathsf{ICSig}(P_i, P_j, P_k, {s'}^{(\ell)}_q)$ from P_i for any $P_k \in \mathcal{P}$, then ${s'}^{(\ell)}_q = {s^{\star}}^{(\ell)}_q$ holds. The linearity of the underlying IC-signatures follow from the fact that the parties follow the linearity principle while generating the IC-signatures.

Now based on the previous two lemmas, we show that in a *synchronous* network, all honest parties will output a "legitimate" set of parties CORE after time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 6T_{\mathsf{BA}}$, such that *at least* one honest party is present in CORE. And corresponding to every party in CORE, there exists some value, which is linearly secret-shared with IC-signatures. Moreover, the values corresponding to the honest parties remain private.

Lemma E.4. *If the network is synchronous, then in* Π_{MDVSS}*, except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, at the time* $T_{\mathsf{MDVSS}} = T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 6T_{\mathsf{BA}}$*, all honest parties output a common set* CORE*, such that at least one honest party will be present in* CORE*. Moreover, corresponding to every* $P_\ell \in \mathsf{CORE}$*, there exists some* ${s^{\star}}^{(\ell)}$*, where* ${s^{\star}}^{(\ell)} = s^{(\ell)}$ *for an honest* P_ℓ*, which is the input of* P_ℓ *for* Π_{MDVSS}*, such that the values* $\{{s^{\star}}^{(\ell)}\}_{P_\ell \in \mathsf{CORE}}$ *are linearly secret-shared with IC-signatures. Furthermore, if* P_ℓ *is honest, then adversary's view is independent of* $s^{(\ell)}$*.*

Proof. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of honest parties. From Lemma E.1, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all the parties in $\mathcal{H}$ will have a common set of committed dealers $\mathcal{CD}$ by the time

$T_{\mathsf{SVM}} + 2T_{\mathsf{BA}}$, where $\mathcal{H} \subseteq \mathcal{CD}$. Moreover, corresponding to every dealer $P_\ell \in \mathcal{CD}$ and every $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$, every party in $(\mathcal{H} \cap S_q)$ will have a common share, say ${s^\star}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$, for an P_ℓ. We begin by showing that once the set of committed dealers $\mathcal{CD}$ is decided, then all the $|\mathcal{Z}_s|$ instances $\Pi_{\mathsf{BA}}^{(1,\star)}$ of Π_{BA} and then all the $|\mathcal{CD}|$ instances $\Pi_{\mathsf{BA}}^{(2,\star)}$ of Π_{BA} will produce some output, for all the parties in $\mathcal{H}$, by time T_{MDVSS}.

Consider the set $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that $S_p = \mathcal{H}$. Then corresponding to each $P_\ell \in (\mathcal{H} \cap \mathcal{CD})$, every $P_i \in \mathcal{H}$ will receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ and includes (P_ℓ, S_p) to $\mathcal{C}_i$ at time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$ (see Lemma E.2). Since $\mathcal{P} \setminus \mathcal{H} \in \mathcal{Z}_s$ and $\mathcal{H} \subseteq \mathcal{CD}$, it follows that at the time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$, every party $P_i \in \mathcal{H}$ will have a set $\mathcal{A}_{p,i}$ (namely $\mathcal{A}_{p,i} = \mathcal{H}$), where $\mathcal{CD} \setminus \mathcal{A}_{p,i} \in \mathcal{Z}_s$ and where $(P_\ell, S_p) \in \mathcal{C}_i$ for every $P_\ell \in \mathcal{A}_{p,i}$. Consequently, each $P_i \in \mathcal{H}$ starts participating in the instance $\Pi_{\mathsf{BA}}^{(1,p)}$ with input 1, at the time $T_{\mathsf{SVM}} + 2T_{\mathsf{BA}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}}$. From the $\mathcal{Z}_s$*-security* of Π_{BA} in the *synchronous* network (see Theorem 3.1), it follows that at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$, every $P_i \in \mathcal{H}$ obtains the output 1 from the instance $\Pi_{\mathsf{BA}}^{(1,p)}$. Consequently, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$, every party in $\mathcal{H}$ will start participating in the remaining $\Pi_{\mathsf{BA}}^{(1,\star)}$ instances for which no input has been provided yet (if there are any), with input 0. And from the $\mathcal{Z}_s$*-security* of Π_{BA} in the *synchronous* network, these $\Pi_{\mathsf{BA}}^{(1,\star)}$ instances will produce common outputs, for every party in $\mathcal{H}$, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$. As a result, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ will compute a q_{core}. Moreover, q_{core} will be *common* for all the parties in $\mathcal{H}$, since it corresponds to the least-indexed $\Pi_{\mathsf{BA}}^{(1,\star)}$ instance among $\Pi_{\mathsf{BA}}^{(1,1)}, \ldots, \Pi_{\mathsf{BA}}^{(1,|\mathcal{Z}_s|)}$, which produces output 1. And from the $\mathcal{Z}_s$*-security* of Π_{BA} in the *synchronous* network, each $\Pi_{\mathsf{BA}}^{(1,q)}$ instance produces a common output for every party in $\mathcal{H}$. We also note that q_{core} will be indeed set to some value from the set $\{1, \ldots, |\mathcal{Z}_s|\}$. This is because as shown above, the index p where $S_p = \mathcal{H}$ always constitute a candidate q_{core}.

We next claim that corresponding to $S_{q_{\mathsf{core}}}$, there exists a subset of parties $\mathcal{B}_{q_{\mathsf{core}}}$, where $\mathcal{CD} \setminus \mathcal{B}_{q_{\mathsf{core}}} \in \mathcal{Z}_s$, such that corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$, the ordered pair $(P_j, S_{q_{\mathsf{core}}})$ is present in the set $\mathcal{C}_i$ of every $P_i \in \mathcal{H}$, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$. Assuming that the claim is true, it implies that all the parties in $\mathcal{H}$ will participate with input 1 in the instances $\Pi_{\mathsf{BA}}^{(2,j)}$, corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$. And hence from the $\mathcal{Z}_s$*-security* of the Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties will obtain the output 1 in the $\Pi_{\mathsf{BA}}^{(2,j)}$ instances, corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$, at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}}$. As a result, at time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ will start participating in the remaining $\Pi_{\mathsf{BA}}^{(2,\star)}$ instances for which no input has been provided yet (if there are any), with input 0. Consequently, from the $\mathcal{Z}_s$*-security* of the Π_{BA} in the *synchronous* network (Theorem 3.1), at the time T_{MDVSS}, all the parties in $\mathcal{H}$ will have some output from all the $|\mathcal{CD}|$ instances of $\Pi_{\mathsf{BA}}^{(2,\star)}$. Moreover, the outputs will be *common* for all the parties in $\mathcal{H}$. Furthermore, the parties in $\mathcal{H}$ will have a subset CORE, which corresponds to all the $\Pi_{\mathsf{BA}}^{(2,j)}$ instances, which have produced output 1. Note that $(\mathcal{H} \cap \mathsf{CORE}) \neq \emptyset$. This is because $\mathcal{B}_{q_{\mathsf{core}}} \subseteq \mathsf{CORE}$ and $\mathcal{H} \subseteq \mathcal{CD}$. Consequently, $(\mathcal{H} \cap \mathcal{B}_{q_{\mathsf{core}}}) \neq \emptyset$, as otherwise $\mathcal{Z}_s$ *does not* satisfy the $\mathbb{Q}^{(2)}(\mathcal{P}, \mathcal{Z}_s)$ condition, which is a contradiction.

We next proceed to prove our claim. Since the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$ has produced output 1, it follows that at least one party from $\mathcal{H}$, say P_k, have participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$. This is because if *all* the parties in $\mathcal{H}$ participates with input 0 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, then from the $\mathcal{Z}_s$-*validity* of Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ would have obtained the output 0 from the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, which is a contradiction. We also note that P_k would have started participating with input 1 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, latest by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$. This is because as argued above, by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ would have started participating in *all* the $|\mathcal{Z}_s|$ instances of $\Pi_{\mathsf{BA}}^{(1,\star)}$, with *some* input. Now since P_k has participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, it follows that at the time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$, there exists a subset of parties $\mathcal{A}_{q_{\mathsf{core}},k}$, where $\mathcal{CD} \setminus \mathcal{A}_{q_{\mathsf{core}},k} \in \mathcal{Z}_s$, such that $(P_\ell, S_{q_{\mathsf{core}}})$ is present in the set $\mathcal{C}_k$, corresponding to every $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$. We show that the set $\mathcal{A}_{q_{\mathsf{core}},k}$ constitutes the candidate $\mathcal{B}_{q_{\mathsf{core}}}$. For this, note that for any $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$, party P_k includes $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_i$, only after receiving a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}})$ from the broadcast of P_ℓ and verifying it. Moreover, P_k must have received the message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}})$ from each $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$, latest by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}}$. It then follows from Lemma E.4 that by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 3T_{\mathsf{BA}} + \Delta < T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$, *every* party in $\mathcal{H}$ would have received $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}})$ from each $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$. And hence each party $P_i \in \mathcal{H}$ would include $(P_\ell, S_{q_{\mathsf{core}}})$ to the set $\mathcal{C}_i$, corresponding to every $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$, by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 4T_{\mathsf{BA}}$. This proves our claim.

We next claim that at the time T_{MDVSS}, corresponding to every $P_\ell \in \mathsf{CORE}$, every $P_i \in \mathcal{H}$ would have received a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}})$ from the broadcast of P_ℓ. The proof for this is very similar to the proof of the previous claim and relies on the properties of Π_{BA}. So consider an *arbitrary* $P_\ell \in \mathsf{CORE}$. This implies that the instance $\Pi_{\mathsf{BA}}^{(2,\ell)}$ has produced output 1 for *all* the parties in $\mathcal{H}$, which further implies that at least one party from $\mathcal{H}$, say P_m, has participated with input 1 during the instance $\Pi_{\mathsf{BA}}^{(2,\ell)}$. If not, then from the $\mathcal{Z}_s$-*validity* of Π_{BA} in the *synchronous* network (Theorem 3.1), the instance $\Pi_{\mathsf{BA}}^{(2,\ell)}$ would have produced output 0 for all the parties in $\mathcal{H}$, which is a contradiction. We also note that P_m must have started participating in the instance $\Pi_{\mathsf{BA}}^{(2,\ell)}$, latest by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}}$. This is because as shown above, by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ would have started participating in *all* the $|\mathcal{CD}|$ instances of $\Pi_{\mathsf{BA}}^{(2,\star)}$, with some input. Now since P_m participates with input 1 in the instance $\Pi_{\mathsf{BA}}^{(2,\ell)}$, it follows that by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}}$, party P_m must have received a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from the broadcast of P_ℓ and included $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_m$. It then follows from Lemma E.4 that by time $T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 5T_{\mathsf{BA}} + \Delta < T_{\mathsf{SVM}} + T_{\mathsf{Auth}} + 2T_{\mathsf{BC}} + 6T_{\mathsf{BA}}$, *every* party P_i in $\mathcal{H}$ would have received $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from P_ℓ and would include $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_i$.

Till now we have shown that all the all the $|\mathcal{Z}_s|$ instances $\Pi_{\mathsf{BA}}^{(1,\star)}$ of Π_{BA} and then all the $|\mathcal{CD}|$ instances $\Pi_{\mathsf{BA}}^{(2,\star)}$ of Π_{BA} will produce some output, for all the parties in $\mathcal{H}$, by time T_{MDVSS}. Moreover, at the time T_{MDVSS}, all the parties in $\mathcal{H}$ will have a common $q_{\mathsf{core}} \in \{1, \ldots, |\mathcal{Z}_s|\}$ and a common set $\mathsf{CORE} \subseteq \mathcal{P}$, where CORE has at least one *honest* party. Furthermore, corresponding to every $P_\ell \in \mathsf{CORE}$, each $P_i \in \mathcal{H}$ would have received a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from the broadcast of P_ℓ. Furthermore, from Lemma E.4, corresponding to each $P_\ell \in \mathsf{CORE}$, the set $\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}$ will be either the set S_q or $(S_{q_{\mathsf{core}}} \cap S_q)$, for $q = 1, \ldots, |\mathcal{Z}_s|$. If $\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q} = S_q$ for *every* $P_\ell \in \mathsf{CORE}$, then all the parties in $\mathcal{H}$ will set $\mathcal{W}_q$ to S_q. On the other hand, if $\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q} = (S_{q_{\mathsf{core}}} \cap S_q)$ for *any* $P_\ell \in \mathsf{CORE}$, then all the parties in $\mathcal{H}$ will set $\mathcal{W}_q$ to $(S_{q_{\mathsf{core}}} \cap S_q)$. Irrespective of the case, all the parties in $\mathcal{H}$ would set $\mathcal{W}_q$ to a common subset. We also note that irrespective of the case, $S_q \setminus \mathcal{W}_q \in \mathcal{Z}_a$ holds. This is because from Lemma E.4 and Lemma E.3, the condition $S_q \setminus \mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q} \in \mathcal{Z}_a$ holds, corresponding to *every* $P_\ell \in \mathsf{CORE}$.

Finally consider an *arbitrary* $P_\ell \in \mathsf{CORE}$ and an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. We claim that at the time T_{MDVSS}, all the parties in $(\mathcal{H} \cap S_q)$ will have a common share, say ${s^\star}^{(\ell)}_q$, where ${s^\star}^{(\ell)}_q = s^{(\ell)}_q$ for an *honest* P_ℓ. For this, we consider two possible cases. If $q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}$, then each $P_i \in (\mathcal{H} \cap S_q)$ would have received $s^{(\ell)}_q$ from the broadcast of P_ℓ, as part of the $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ message. Consequently, in this case ${s^\star}^{(\ell)}_q$ is the same as $s^{(\ell)}_q$, received from the broadcast of P_ℓ. On the other hand, if $q \notin \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}$, then from Lemma E.3, each $P_i \in (\mathcal{H} \cap S_q)$ would have a common share, say ${s^{(\star)}}^{(\ell)}_q$; moreover, from Lemma E.2, if P_ℓ is *honest*, then ${s^{(\star)}}^{(\ell)}_q = s^{(\ell)}_q$ holds. We define

$$ {s^\star}^{(\ell)} \stackrel{def}{=} \sum_{q=1,\ldots,|\mathcal{Z}_s|} {s^{(\star)}}^{(\ell)}_q, $$

where ${s^\star}^{(\ell)} = s^{(\ell)}$ for an *honest* P_ℓ. Hence at time T_{MDVSS}, each party in $(\mathcal{H} \cap S_q)$ has $[{s^\star}^{(\ell)}]_q$. We also note that if $q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}$, then *every* $P_i \in \mathcal{W}_q$ sets $\mathsf{ICSig}(P_j, P_i, P_k, [{s^\star}^{(\ell)}]_q)$ to the default value, corresponding to every $P_j \in \mathcal{W}_q$ and every $P_k \in \mathcal{P}$. On the other hand, if $q \notin \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}$, then every $P_i \in (\mathcal{H} \cap S_q)$ will have $\mathsf{ICSig}(P_j, P_i, P_k, {s^{\star}}_q^{(\ell)})$ of every $P_j \in \mathcal{W}_q$ and for every $P_k \in \mathcal{P}$. Furthermore, if any corrupt $P_j \in \mathcal{W}_q$ has $\mathsf{ICSig}(P_i, P_j, P_k, {s'}_q^{(\ell)})$ of any $P_i \in (\mathcal{H} \cap S_q)$ for any $P_k \in \mathcal{P}$, then ${s'}_q^{(\ell)} = {s^{\star}}_q^{(\ell)}$ holds. Moreover, from Lemma E.3, if P_ℓ is *honest*, then ${s^{\star}}_q^{(\ell)}$ in the IC-signatures mentioned above will be the same as $s_q^{(\ell)}$. It then follows that ${s^{\star}}^{(\ell)}$ will be linearly secret-shared; the linearity of the underlying IC-signatures follows since the (honest) parties follow the linearity principle, while generating the IC-signatures.

The privacy of $s^{(\ell)}$ for an *honest* P_ℓ follows from Lemma E.3.

We next consider an *asynchronous* network. We first prove an analogue of Lemma E.1 in the asynchronous network.

Lemma E.5. *If the network is asynchronous and $P_\ell \in \mathcal{P}$ is an honest dealer participating with input $s^{(\ell)}$, then all the following hold in Π_{MDVSS}, where $\mathcal{H}$ is the set of honest parties.*

- *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, all the parties in $\mathcal{H}$ will eventually have a common $\mathcal{CD}$ set, where $\mathcal{P} \setminus \mathcal{CD} \in \mathcal{Z}_s$.*
- *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, corresponding to every dealer $P_\ell \in \mathcal{CD}$ and every $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$, every party in $(\mathcal{H} \cap S_q)$ will eventually have a common share, say ${s^{\star}}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$, for an honest P_ℓ.*

Proof. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Note that $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$, since $\mathcal{Z}_a \subset \mathcal{Z}_s$. From the properties of Π_{SVM} in the *asynchronous* network (Lemma 6.3), it follows that every $P_i \in \mathcal{H}$ will eventually set $\mathsf{flag}^{(P_\ell, S_q)}$ to 1 during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$, corresponding to every $P_\ell \in \mathcal{H}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. Moreover, corresponding to every $P_\ell \in \mathcal{H}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, each $P_i \in (\mathcal{H} \cap S_q)$ eventually computes an output $s_{qi}^{(\ell)}$ during the instance $\Pi_{\mathsf{SVM}}((P_\ell, s_q^{(\ell)}, S_q)$. Furthermore, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the value $s_{qi}^{(\ell)}$ will be the same as $s_q^{(\ell)}$.

We first claim that there always exists a subset of parties $\mathcal{D}$, where $\mathcal{P} \setminus \mathcal{D} \in \mathcal{Z}_s$, such that the Π_{BA} instance $\Pi_{\mathsf{BA}}^{(\ell)}$ eventually produces output 1 for all the parties in $\mathcal{H}$, corresponding to every $P_\ell \in \mathcal{D}$. Assuming that the claim is true, it implies that all the parties in $\mathcal{H}$ will eventually participate with some input in the Π_{BA} instances $\Pi_{\mathsf{BA}}^{(1)}, \ldots, \Pi_{\mathsf{BA}}^{(n)}$. This is because from the protocol steps, once the $\Pi_{\mathsf{BA}}^{(\star)}$ instances corresponding to the parties in $\mathcal{D}$ produce output 1, all the parties in $\mathcal{H}$ will start participating with input 0 in the remaining instances $\Pi_{\mathsf{BA}}^{(\star)}$ of Π_{BA} (if any), for which no input has been provided yet. And hence from the $\mathcal{Z}_a$*-security* of Π_{BA} in the *asynchronous* network, it follows that almost-surely, all these Π_{BA} instances will eventually produce some output for all the parties in $\mathcal{H}$. Moreover, the outputs will be the same for all the parties in $\mathcal{H}$. Consequently, all the parties in $\mathcal{H}$ will eventually obtain a common $\mathcal{CD}$ set. Moreover, $\mathcal{P} \setminus \mathcal{CD} \in \mathcal{Z}_s$, since $\mathcal{CD}$ consists of all those parties P_ℓ, such that the instance $\Pi_{\mathsf{BA}}^{(\ell)}$ produces output 1. And according to our claim, $\mathcal{D} \subseteq \mathcal{CD}$ holds. We now proceed to prove our claim.

There are two possible cases. Consider the case when *some* $P_i \in \mathcal{H}$ starts participating with input 0 in any $\Pi_{\mathsf{BA}}^{(\star)}$ instance. This implies that for P_i, there exists a subset of parties $\mathcal{CD}_i$ where $\mathcal{P} \setminus \mathcal{CD}_i \in \mathcal{Z}_s$, such that corresponding to every $P_\ell \in \mathcal{CD}_i$, the instance $\Pi_{\mathsf{BA}}^{(\ell)}$ has produced output 1 for P_i. In this case, the set $\mathcal{CD}_i$ is the candidate $\mathcal{D}$ set, whose existence we want to prove. Next, consider the case when *no* party in $\mathcal{H}$ has started participating with input 0 in *any* of the $\Pi_{\mathsf{BA}}^{(\star)}$ instances. In this case, the set $\mathcal{H}$ constitutes the candidate $\mathcal{D}$ set. This is because as shown above, every $P_i \in \mathcal{H}$ will eventually set $\mathsf{flag}^{(P_\ell, S_q)}$ to 1, corresponding to every $P_\ell \in \mathcal{H}$ and every $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. And hence every $P_i \in \mathcal{H}$ will eventually start participating with input 1 in the $\Pi_{\mathsf{BA}}^{(\ell)}$ instances, corresponding to $P_\ell \in \mathcal{H}$. Consequently, the $\mathcal{Z}_s$-*validity* of Π_{BA} in the *asynchronous* network (Theorem 3.1) will guarantee that the $\Pi_{\mathsf{BA}}^{(\ell)}$ instances, corresponding to $P_\ell \in \mathcal{H}$ eventually produce output 1 for all the parties in $\mathcal{H}$.

Next, consider an *arbitrary* $P_\ell \in \mathcal{CD}$. This implies that at least one party from $\mathcal{H}$, say P_k, has participated with input 1 during the instance $\Pi_{\mathsf{BA}}^{(\ell)}$. If not, then from the $\mathcal{Z}_s$-*validity* of Π_{BA} in the *asynchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ would have obtained the output 0 from the instance $\Pi_{\mathsf{BA}}^{(\ell)}$ and hence $P_\ell \notin \mathcal{CD}$, which is a *contradiction*. This implies that party P_k has set $\mathsf{flag}^{(P_\ell, S_q)}$ to 1 during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$, for $q = 1, \ldots, |\mathcal{Z}_s|$. So consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. From the properties of Π_{SVM} in the *asynchronous* network (Lemma 6.3), it follows that there exists some value ${s^\star}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$ for an *honest* P_ℓ, such that except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all the parties in $\mathcal{H}$ eventually output ${s^\star}_q^{(\ell)}$ during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$.

We next prove the analogue of Lemma E.2 in the *asynchronous* network.

Lemma E.6. *If the network is asynchronous and $P_\ell \in \mathcal{CD}$ is an honest dealer participating with input $s^{(\ell)}$, then all the following hold in Π_{MDVSS} except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, where $\mathcal{H}$ is the set of honest parties.*

- *If $S_p = \mathcal{H}$, then P_ℓ will eventually broadcast $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$.*
- *If P_ℓ broadcasts any $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ then every honest $P_i \in \mathcal{P}$ will eventually include (P_ℓ, S_p) to $\mathcal{C}_i$. Moreover, the following will hold.*
  - *If $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}_p^{(\ell)}$.*
  - *For $q = 1, \ldots, |\mathcal{Z}_s|$, each $\mathcal{W}_{p,q}^{(\ell)}$ will be either S_q or $(S_p \cap S_q)$ such that $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_{p,q}^{(\ell)}, \mathcal{Z}_s)$ condition.*
  - *If $q \notin \mathcal{BS}_p^{(\ell)}$, then every honest $P_i \in S_q$ will have the share $s_q^{(\ell)}$. Moreover, every honest $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, s_q^{(\ell)})$ of every $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ for every $P_k \in \mathcal{P}$. Furthermore, if any corrupt $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ have $\mathsf{ICSig}(P_i, P_j, P_k, {s'}_q^{(\ell)})$ of any honest $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ for any $P_k \in \mathcal{P}$, then ${s'}_q^{(\ell)} = s_q^{(\ell)}$ holds. Also, all the underlying IC-signatures will satisfy the linearity property.*
- *The view of the adversary will be independent of $s^{(\ell)}$.*

Proof. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We first note that $\mathcal{H} \in \mathbb{S}_{\mathcal{Z}_s}$, since $Z^\star \in \mathcal{Z}_s$ as $\mathcal{Z}_a \subset \mathcal{Z}_s$. The proof for the first part of the lemma is similar to the proof of the first part of Lemma E.2, except that all the "favourable" conditions hold for an *honest* P_ℓ *eventually.* In more detail, consider an *arbitrary* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$. Then each $P_i \in (S_q \cap \mathcal{H})$ eventually computes the share $s_{qi}^{(\ell)}$ during the instance $\Pi_{\mathsf{SVM}}(P_\ell, s_q^{(\ell)}, S_q)$, where $s_{qi}^{(\ell)} = s_q^{(\ell)}$ holds, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Consequently, P_i starts giving $\mathsf{ICSig}(P_i, P_j, P_k, s_{qi}^{(\ell)})$ to every $P_j \in S_q$, for every $P_k \in \mathcal{P}$. Then from the $\mathcal{Z}_a$*-correctness* of Π_{Auth} in the *asynchronous* network (Theorem 4.1), it follows that each party $P_i \in (S_q \cap \mathcal{H})$ will eventually receive $\mathsf{ICSig}(P_j, P_i, P_k, s_{qj}^{(\ell)})$ from every $P_j \in (S_q \cap \mathcal{H})$, for every $P_k \in \mathcal{P}$, such that $s_{qj}^{(\ell)} = s_{qi}^{(\ell)} = s_q^{(\ell)}$ holds. Since S_q is arbitrary, it follows that eventually, every party $P_i \in \mathcal{H}$ broadcasts an $\mathsf{OK}^{(\ell)}(i, j)$ message, corresponding to every $P_j \in \mathcal{H}$. From the $\mathcal{Z}_a$*-weak validity* and $\mathcal{Z}_a$*-fallback validity* of Π_{BC} in the *asynchronous* network (Theorem 3.2), it follows that these $\mathsf{OK}^{(\ell)}(i, j)$ messages are eventually received by every party in $\mathcal{H}$. Consequently, the set $\mathcal{H}$ eventually becomes a clique in the consistency graph $G^{(\ell,i)}$ of every party $P_i \in \mathcal{H}$. Let S_p be the set from $\mathbb{S}_{\mathcal{Z}_s}$, such that $S_p = \mathcal{H}$. From the protocol steps, it then follows that the dealer P_ℓ will eventually compute core-sets $\mathcal{W}_{p,q}^{(\ell)}$ for $q = 1, \ldots, |\mathcal{Z}_s|$ and broadcast-set $\mathcal{BS}_p^{(\ell)}$ with respect to S_p as follows,

- If S_q constitutes a clique in the graph $G^{(\ell,\ell)}$, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as S_q.
- Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,\ell)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as $(S_p \cap S_q)$.
- Else $\mathcal{W}_{p,q}^{(\ell)}$ is set to S_q and q is included to $\mathcal{BS}_p^{(\ell)}$.

After computing the core-sets and broadcast-set, P_ℓ will eventually broadcast $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$.

We next proceed to prove the second part of the lemma, whose proof is again similar to the proof of the second part of the Lemma E.2, except that all the "favourable" conditions which hold for P_ℓ, are guaranteed to hold eventually for all the parties in $\mathcal{H}$. In more detail, consider an *arbitrary* $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, such that P_ℓ compute core-sets $\mathcal{W}_{p,q}^{(\ell)}$ for $q = 1, \ldots, |\mathcal{Z}_s|$ and broadcast-set $\mathcal{BS}_p^{(\ell)}$ with respect to S_p and broadcasts $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$. This means the parties in S_p constitute a clique in the graph $G^{(\ell,\ell)}$. We note that all the edges which are present in the graph $G^{(\ell,\ell)}$ when S_p constitute a clique in $G^{(\ell,\ell)}$ are bound to be eventually included in the graph $G^{(\ell,i)}$ of every party $P_i \in \mathcal{H}$. This is because the edges are included by P_ℓ based on various $\mathsf{OK}^{(\ell)}(\star, \star)$ messages, which are received by P_ℓ through various Π_{BC} instances. Consequently, due to the various properties of Π_{BC} in the *asynchronous* network, these $\mathsf{OK}^{(\ell)}(\star, \star)$ messages are bound to be eventually delivered to every party in $\mathcal{H}$. As a result, all the properties which hold for P_ℓ in the graph $G^{(\ell,\ell)}$ when S_p constitute a clique in $G^{(\ell,\ell)}$ are bound to hold eventually for every

$P_i \in \mathcal{H}$ in the graph $G^{(\ell,\ell)}$. Since P_ℓ is assumed to be *honest*, it computes the sets $\{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}_p^{(\ell)}$, satisfying the following properties.

- If S_q constitutes a clique in the graph $G^{(\ell,\ell)}$, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as S_q.
- Else if $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,\ell)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, then $\mathcal{W}_{p,q}^{(\ell)}$ is set as $(S_p \cap S_q)$.
- Else $\mathcal{W}_{p,q}^{(\ell)}$ is set to S_q and q is included to $\mathcal{BS}_p^{(\ell)}$.

We also note that if $S_q = \mathcal{H}$, then $q \notin \mathcal{BS}_p^{(\ell)}$ and consequently, P_ℓ will *not* make the share $s_q^{(\ell)}$ public. This is because P_ℓ will set $\mathcal{W}_{p,q}^{(\ell)}$ to $(S_p \cap S_q)$. In more detail, the parties in $(S_p \cap S_q)$ will constitute a clique in $G^{(\ell,\ell)}$, since S_p constitutes a clique in $G^{(\ell,\ell)}$, when P_ℓ starts computing the core-sets $\{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}$. Moreover, $\mathcal{Z}_s$ will satisfy the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition, due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition.

Since P_ℓ is *honest*, from the *$\mathcal{Z}_a$-weak validity* and *$\mathcal{Z}_a$-fallback validity* of Π_{BC} in the *asynchronous* network (see Theorem 3.2), it follows that all the parties in $\mathcal{H}$ will eventually receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ. Moreover, each party $P_i \in \mathcal{H}$ will eventually include (P_ℓ, S_p) to the set $\mathcal{C}_i$. This is because since P_ℓ has computed the sets $\{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}$ and $\mathcal{BS}_p^{(\ell)}$ *honestly*, these sets will eventually pass all the verifications for each $P_i \in \mathcal{H}$. The last statement is true because as shown above, all the properties which hold for P_ℓ in the graph $G^{(\ell,\ell)}$ when S_p constitute a clique in $G^{(\ell,\ell)}$ are bound to hold eventually for every $P_i \in \mathcal{H}$ in the graph $G^{(\ell,\ell)}$.

The proof for the rest of the properties stated in the lemma is similar to that of Lemma E.2, except that we now rely on the security properties of ICP in the *asynchronous* network; to avoid repetition we do not produce the details here. The linearity of the underlying IC-signatures is ensured since the parties follow the linearity principle while generating the IC-signatures.

We next prove an analogue of Lemma E.3 in the *asynchronous* network.

Lemma E.7. *If the network is asynchronous and if in Π_{MDVSS} any honest party P_i receives $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of any corrupt dealer $P_\ell \in \mathcal{CD}$ and includes (P_ℓ, S_p) to $\mathcal{C}_i$, then all honest parties P_j will eventually receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ and include (P_ℓ, S_p) to $\mathcal{C}_j$. Moreover, for $q = 1, \ldots, |\mathcal{Z}_s|$, the following holds, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$.*

- *$\mathcal{W}_{p,q}^{(\ell)}$ is either S_q or $(S_p \cap S_q)$. Moreover, $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(\mathcal{W}_{p,q}^{(\ell)}, \mathcal{Z}_s)$ condition.*
- *If $q \notin \mathcal{BS}_p^{(\ell)}$, then every honest $P_i \in S_q$ will have a common share, say ${s^\star}_q^{(\ell)}$. Moreover, every honest $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ will have $\mathsf{ICSig}(P_j, P_i, P_k, {s^\star}_q^{(\ell)})$ of every $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ for every $P_k \in \mathcal{P}$. Furthermore, if any corrupt $P_j \in \mathcal{W}_{p,q}^{(\ell)}$ has $\mathsf{ICSig}(P_i, P_j, P_k, {s'}_q^{(\ell)})$ of any honest $P_i \in \mathcal{W}_{p,q}^{(\ell)}$ for any $P_k \in \mathcal{P}$, then*

$s'^{(\ell)}_q = s^{\star(\ell)}_q$ holds. Also, all the underlying IC-signatures will satisfy the linearity property.

Proof. The proof is very similar to the proof of Lemma E.3, except that we now rely on the properties of Π_{BC} in the *asynchronous* network. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Now consider an *arbitrary corrupt* dealer $P_\ell \in \mathcal{CD}$ and an *arbitrary* $P_i \in \mathcal{H}$, such that P_i receives $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}^{(\ell)}_{p,q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_p, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_p})$ from the broadcast of P_ℓ and includes (P_ℓ, S_p) to $\mathcal{C}_i$. Now consider another *arbitrary* $P_j \in \mathcal{H}$, such that $P_j \neq P_i$. From the $\mathcal{Z}_a$*-weak consistency* and $\mathcal{Z}_a$*-fallback consistency* of Π_{BC} in the *asynchronous* network, it follows that P_j is bound to eventually receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}^{(\ell)}_{p,q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_p, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_p})$ from the broadcast of P_ℓ. We wish to show that P_j will eventually include (P_ℓ, S_p) to $\mathcal{C}_j$. For this, we note that since P_i has included (P_ℓ, S_p) to $\mathcal{C}_i$, *all* the following conditions hold for P_i, for $q = 1, \ldots, |\mathcal{Z}_s|$.

- If $q \in \mathcal{BS}^{(\ell)}_p$, then the set $\mathcal{W}^{(\ell)}_{p,q} = S_q$.
- If $(q \notin \mathcal{BS}^{(\ell)}_p)$, then $\mathcal{W}^{(\ell)}_{p,q}$ is either S_q or $(S_p \cap S_q)$, such that:
  - If $\mathcal{W}^{(\ell)}_{p,q} = S_q$, then S_q constitutes a clique in $G^{(\ell,i)}$.
  - Else if $\mathcal{W}^{(\ell)}_{p,q} = (S_p \cap S_q)$, then $(S_p \cap S_q)$ constitutes a clique in $G^{(\ell,i)}$ and $\mathcal{Z}_s$ satisfies the $\mathbb{Q}^{(1)}(S_p \cap S_q, \mathcal{Z}_s)$ condition.

We claim that all the above conditions will hold eventually *even* for P_j. This is because *all* the edges which are present in the consistency graph $G^{(\ell,i)}$ when P_i includes (P_ℓ, S_p) to $\mathcal{C}_i$ are bound to be eventually present in the consistency graph $G^{(\ell,j)}$. This follows from the $\mathcal{Z}_a$*-weak validity,* $\mathcal{Z}_a$*-fallback validity,* $\mathcal{Z}_a$*-weak consistency* and $\mathcal{Z}_a$*-fallback consistency* of Π_{BC} in the *asynchronous* network (see Theorem 3.2) and the fact that the edges in the graph $G^{(\ell,i)}$ are based on $\mathsf{OK}^{(\ell)}(\star,\star)$ messages, which are received through various Π_{BC} instances.

The proof regarding the IC-signatures is exactly the same as Lemma E.3 and to avoid repetition, we do not produce the formal details here.

Finally, based on the previous two lemmas, we prove an analogue of Lemma E.4 and show that in an *asynchronous* network, all honest parties will eventually output a "legitimate" set of parties CORE.

Lemma E.8. *If the network is asynchronous, then in Π_{MDVSS}, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely all honest parties eventually output a common set CORE, such that at least one honest party will be present in CORE. Moreover, corresponding to every $P_\ell \in \mathsf{CORE}$, there exists some $s^{\star(\ell)}$, where $s^{\star(\ell)} = s^{(\ell)}$ for an honest P_ℓ, which is the input of P_ℓ for Π_{MDVSS}, such that the values $\{s^{\star(\ell)}\}_{P_\ell \in \mathsf{CORE}}$ are linearly secret-shared with IC-signatures. Furthermore, if P_ℓ is honest, then the adversary's view is independent of $s^{(\ell)}$.*

Proof. The proof structure is very similar to that of Lemma E.4, except that we now rely on the properties of Π_{BA} and Π_{BC} in the *asynchronous* network and Lemma E.5-E.7. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of honest parties. From Lemma E.5, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely all the parties in $\mathcal{H}$ will eventually have a common set of committed dealers $\mathcal{CD}$, where $\mathcal{P} \backslash \mathcal{CD} \in \mathcal{Z}_s$. Moreover, corresponding to every dealer $P_\ell \in \mathcal{CD}$ and every $S_q \in \mathbb{S}_{|\mathcal{Z}_s|}$, every party in $(\mathcal{H} \cap S_q)$ will eventually have a common share, say ${s^\star}_q^{(\ell)}$, which is the same as $s_q^{(\ell)}$, for an P_ℓ. We begin by showing that once the set of committed dealers $\mathcal{CD}$ is decided, then almost-seurely, all the $|\mathcal{Z}_s|$ instances $\Pi_{\mathsf{BA}}^{(1,\star)}$ of Π_{BA} and then all the $|\mathcal{CD}|$ instances $\Pi_{\mathsf{BA}}^{(2,\star)}$ of Π_{BA} will eventually produce some output, for all the parties in $\mathcal{H}$.

We first claim that irrespective of way messages are scheduled and the order in which the parties in $\mathcal{H}$ participate in various $\Pi_{\mathsf{BA}}^{(1,\star)}$ instances, there will be some instance $\Pi_{\mathsf{BA}}^{(1,p)}$ corresponding to some $S_p \in \mathbb{S}_{\mathcal{Z}_s}$, which will eventually produce output 1 for all the parties in $\mathcal{H}$. For this, consider the set $S_p \in \mathbb{S}_{|\mathcal{Z}_s|}$, such that $S_p = \mathcal{H}$ (such an S_p is bound to exist since $Z^\star \in \mathcal{Z}_s$ also holds, as $\mathcal{Z}_a \subset \mathcal{Z}_s$). If there exists *some* $P_i \in \mathcal{H}$ which starts participating with input 0 in the instance $\Pi_{\mathsf{BA}}^{(1,p)}$, then the claim is true, because P_i participates with input 0 during $\Pi_{\mathsf{BA}}^{(1,p)}$, only after receiving the output 1 from some other instance of $\Pi_{\mathsf{BA}}^{(1,\star)}$, say $\Pi_{\mathsf{BA}}^{(1,q)}$. And hence from the $\mathcal{Z}_a$*-consistency* of Π_{BA} in the *asynchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ will eventually obtain the output 1 from the instance $\Pi_{\mathsf{BA}}^{(1,q)}$, thus proving our claim. On the other hand, consider the case when *no* party has yet started participating with any input in the instance $\Pi_{\mathsf{BA}}^{(1,p)}$. Then corresponding to each $P_\ell \in (\mathcal{H} \cap \mathcal{CD})$, every $P_i \in \mathcal{H}$ will eventually receive $(\mathsf{CanCS}, P_\ell, S_p, \{\mathcal{W}_{p,q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_p^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_p^{(\ell)}})$ from the broadcast of P_ℓ and includes (P_ℓ, S_p) to $\mathcal{C}_i$ (see Lemma E.6). Since $\mathcal{CD} \setminus \mathcal{H} \subseteq Z^\star$ and $Z^\star \in \mathcal{Z}_s$ (due to condition $\mathcal{Z}_s \subset \mathcal{Z}_a$), it follows that every party $P_i \in \mathcal{H}$ will eventually have a set $\mathcal{A}_{p,i}$ (namely $\mathcal{A}_{p,i} = \mathcal{H}$), where $\mathcal{CD} \setminus \mathcal{A}_{p,i} \in \mathcal{Z}_s$ and where $(P_\ell, S_p) \in \mathcal{C}_i$ for every $P_\ell \in \mathcal{A}_{p,i}$. Consequently, each $P_i \in \mathcal{H}$ will eventually start participating in the instance $\Pi_{\mathsf{BA}}^{(1,p)}$ with input 1, if they have not done so. And from the $\mathcal{Z}_s$*-validity* of Π_{BA} in the *asynchronous* network (see Theorem 3.1), it follows that all the parties in $\mathcal{H}$ eventually obtain the output 1 from the instance $\Pi_{\mathsf{BA}}^{(1,p)}$, thus proving our claim in this case as well.

Now from the above claim, it follows that every party in $\mathcal{H}$ will eventually start participating in the remaining $\Pi_{\mathsf{BA}}^{(1,\star)}$ instances for which no input has been provided yet (if there are any), with input 0. And from the $\mathcal{Z}_a$*-security* of Π_{BA} in the *asynchronous* network, almost-surely, these $\Pi_{\mathsf{BA}}^{(1,\star)}$ instances will eventually produce common outputs, for every party in $\mathcal{H}$. As a result, all the parties in $\mathcal{H}$ will eventually compute a $q_{\mathsf{core}} \in \{1, \ldots, |\mathcal{Z}_s|\}$. Moreover, q_{core} will be *common* for all the parties in $\mathcal{H}$, since it corresponds to the least-indexed $\Pi_{\mathsf{BA}}^{(1,\star)}$ instance among $\Pi_{\mathsf{BA}}^{(1,1)}, \ldots, \Pi_{\mathsf{BA}}^{(1,|\mathcal{Z}_s|)}$, which produces output 1. And from the $\mathcal{Z}_a$*-security* of Π_{BA} in the *asynchronous* network, each $\Pi_{\mathsf{BA}}^{(1,\star)}$ instance produces a common output for every party in $\mathcal{H}$. We also note that q_{core} will be indeed set to some value from the set $\{1, \ldots, |\mathcal{Z}_s|\}$. This is because as shown above, the index p where $S_p = \mathcal{H}$ always constitute a candidate q_{core}.

We next claim that corresponding to $S_{q_{\mathsf{core}}} \in \mathbb{S}_{\mathcal{Z}_s}$, there exists a subset of parties $\mathcal{B}_{q_{\mathsf{core}}}$, where $\mathcal{CD} \setminus \mathcal{B}_{q_{\mathsf{core}}} \in \mathcal{Z}_s$, such that corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$, the ordered pair $(P_j, S_{q_{\mathsf{core}}})$ is eventually included in the set $\mathcal{C}_i$ of every $P_i \in \mathcal{H}$. Assuming that the claim is true, we next show that there always exists a set of parties $\mathcal{B}$ where $\mathcal{CD} \setminus \mathcal{B} \in \mathcal{Z}_s$, such that the $\Pi_{\mathsf{BA}}^{(2,j)}$ instance eventually produce output 1 for all the parties in $\mathcal{H}$, corresponding to every $P_j \in \mathcal{B}$. For this, we consider two possible cases. If any $P_i \in \mathcal{H}$ has started participating with input 0 in any instance $\Pi_{\mathsf{BA}}^{(2,\star)}$, then it implies that for P_i, there exists a subset $\mathcal{B}_i$ where $\mathcal{CD} \setminus \mathcal{B}_i \in \mathcal{Z}_s$ and where corresponding to each $P_j \in \mathcal{B}_i$, the instance $\Pi_{\mathsf{BA}}^{(2,j)}$ has produced output 1. Hence from the $\mathcal{Z}_a$-*consistency* of Π_{BA} in the *asynchronous* network (see Theorem 3.1), all these $\Pi_{\mathsf{BA}}^{(2,j)}$ instances will eventually produce output 1 for all the parties in $\mathcal{H}$. On the other hand, consider the case when *no* party in $\mathcal{H}$ has started participating with input 0 in any instance $\Pi_{\mathsf{BA}}^{(2,\star)}$. Then as per the claim, all the parties in $\mathcal{H}$ will eventually start participating with input 1 in the instances $\Pi_{\mathsf{BA}}^{(2,j)}$, corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$. Hence from the $\mathcal{Z}_a$-*security* of the Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties will eventually obtain the output 1 in the $\Pi_{\mathsf{BA}}^{(2,j)}$ instances, corresponding to every $P_j \in \mathcal{B}_{q_{\mathsf{core}}}$. Thus irrespective of the case, the set $\mathcal{B}$ is guaranteed. As a result, all the parties in $\mathcal{H}$ will eventually start participating in the remaining $\Pi_{\mathsf{BA}}^{(2,\star)}$ instances for which no input has been provided yet (if there are any), with input 0. Consequently, from the $\mathcal{Z}_a$-*security* of the Π_{BA} in the *asynchronous* network (Theorem 3.1), almost-surely, all the parties in $\mathcal{H}$ will eventually have some output from all the $|\mathcal{CD}|$ instances of $\Pi_{\mathsf{BA}}^{(2,\star)}$. Moreover, the outputs will be *common* for all the parties in $\mathcal{H}$. Furthermore, the parties in $\mathcal{H}$ will have a subset CORE, which corresponds to all the $\Pi_{\mathsf{BA}}^{(2,j)}$ instances, which produces output 1. Note that $\mathcal{CD} \setminus \mathsf{CORE} \in \mathcal{Z}_s$ holds, since we have shown that the $\Pi_{\mathsf{BA}}^{(2,j)}$ instances, corresponding to the parties $P_j \in \mathcal{B}$ will produce output 1, implying $\mathcal{B} \subseteq \mathsf{CORE}$. And $\mathcal{CD} \setminus \mathcal{B} \in \mathcal{Z}_s$ holds. Now since $\mathcal{P} \setminus \mathcal{CD} \in \mathcal{Z}_s$ and $\mathcal{CD} \setminus \mathsf{CORE} \in \mathcal{Z}_s$, it follows that $(\mathcal{H} \cap \mathsf{CORE}) \neq \emptyset$, since the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition is satisfied.

We next proceed to prove our claim about the existence of $\mathcal{B}_{q_{\mathsf{core}}}$. Since the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$ has produced output 1, it follows that at least one party from $\mathcal{H}$, say P_k, must have participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$. This is because if *all* the parties in $\mathcal{H}$ participates with input 0 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, then from the $\mathcal{Z}_a$-*validity* of Π_{BA} in the *asynchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ would have obtained the output 0 from the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, which is a contradiction. Now since P_k has participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(1,q_{\mathsf{core}})}$, it follows there exists a subset of parties $\mathcal{A}_{q_{\mathsf{core}},k}$, where $\mathcal{CD} \setminus \mathcal{A}_{q_{\mathsf{core}},k} \in \mathcal{Z}_s$, such that $(P_\ell, S_{q_{\mathsf{core}}})$ is present in the set $\mathcal{C}_k$, corresponding to every $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$. We show that the set $\mathcal{A}_{q_{\mathsf{core}},k}$ constitutes the candidate $\mathcal{B}_{q_{\mathsf{core}}}$. For this, note that for any $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$, party P_k includes $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_i$, only after receiving a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}_{q_{\mathsf{core}},q}^{(\ell)}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}, \{s_q^{(\ell)}\}_{q \in \mathcal{BS}_{q_{\mathsf{core}}}^{(\ell)}})$ from the broadcast of P_ℓ and verifying it. It then follows from Lemma E.8 that eventually, *every* party in $\mathcal{H}$ would have received

$(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from each $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$. Hence each party $P_i \in \mathcal{H}$ would eventually include $(P_\ell, S_{q_{\mathsf{core}}})$ to the set $\mathcal{C}_i$, corresponding to every $P_\ell \in \mathcal{A}_{q_{\mathsf{core}},k}$.

We next claim that corresponding to every $P_\ell \in \mathsf{CORE}$, every $P_i \in \mathcal{H}$ eventually receives a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from the broadcast of P_ℓ. The proof for this is very similar to the proof of the previous claim and relies on the properties of Π_{BA} in the *asynchronous* network. So consider an *arbitrary* $P_\ell \in \mathsf{CORE}$. This implies that the instance $\Pi^{(2,\ell)}_{\mathsf{BA}}$ have produced output 1, which further implies that at least one party from $\mathcal{H}$, say P_m, have participated with input 1 during the instance $\Pi^{(2,\ell)}_{\mathsf{BA}}$. If not, then from the $\mathcal{Z}_a$*-validity* of Π_{BA} in the *asynchronous* network (Theorem 3.1), the instance $\Pi^{(2,\ell)}_{\mathsf{BA}}$ would have produced output 0, which is a contradiction. Now since P_m participates with input 1 in the instance $\Pi^{(2,\ell)}_{\mathsf{BA}}$, it follows that P_m must have received a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from the broadcast of P_ℓ and included $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_m$. It then follows from Lemma E.8 that eventually, *every* party P_i in $\mathcal{H}$ would receive $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from P_ℓ and includes $(P_\ell, S_{q_{\mathsf{core}}})$ to $\mathcal{C}_i$.

Till now we have shown that almost-surely, all the $|\mathcal{Z}_s|$ instances $\Pi^{(1,\star)}_{\mathsf{BA}}$ of Π_{BA} and then all the $|\mathcal{CD}|$ instances $\Pi^{(2,\star)}_{\mathsf{BA}}$ of Π_{BA} will eventually produce some output, for all the parties in $\mathcal{H}$. Moreover, all the parties in $\mathcal{H}$ will eventually have a common $q_{\mathsf{core}} \in \{1, \ldots, |\mathcal{Z}_s|\}$ and a common set $\mathsf{CORE} \subseteq \mathcal{P}$, where CORE has at least one *honest* party. Furthermore, corresponding to every $P_\ell \in \mathsf{CORE}$, each $P_i \in \mathcal{H}$ would eventually received a message $(\mathsf{CanCS}, P_\ell, S_{q_{\mathsf{core}}}, \{\mathcal{W}^{(\ell)}_{q_{\mathsf{core}},q}\}_{q=1,\ldots,|\mathcal{Z}_s|}, \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}, \{s^{(\ell)}_q\}_{q \in \mathcal{BS}^{(\ell)}_{q_{\mathsf{core}}}})$ from the broadcast of P_ℓ. The rest of the proof will be now same as Lemma E.4, except that we now rely on Lemma E.6 and Lemma E.7. To avoid repetition, we do not produce the formal details here.

We finally derive the communication complexity of the protocol.

Lemma E.9. *Protocol Π_{MDVSS} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n^9 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits. In addition, $\mathcal{O}(|\mathcal{Z}_s| + n)$ instances of Π_{BA} are invoked.*

Proof. The number of Π_{BA} instances follows easily from the protocol inspection. The communication complexity of the protocol is dominated by the instances of Π_{SVM} and Π_{Auth} invoked in the protocol. There are $\mathcal{O}(|\mathcal{Z}_s| \cdot n)$ instances of Π_{SVM} and $\mathcal{O}(|\mathcal{Z}_s| \cdot n^4)$ instances of Π_{Auth} invoked. The communication complexity now follows from the communication complexity of Π_{SVM} (Lemma 6.3) and Π_{Auth} (Theorem 4.1).

Theorem 7.1 now follows from Lemma E.4, Lemma E.8 and Lemma E.9.

F Properties of the Triple-Generation Protocol

In this section, we prove the properties of the triple-generation protocol and related sub-protocols.

F.1 Properties of the Protocol Π_{LSh}

We first prove the properties of the protocol Π_{LSh} (see Fig 10 for the formal description).

Lemma 8.1. *Let r be a random value which is linearly secret-shared with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Then protocol Π_{LSh} achieves the following where D participates with the input s.*

- *If D is honest, then the following hold, where $T_{\mathsf{LSh}} = T_{\mathsf{Rec}} + T_{\mathsf{BC}}$.*
  - $\mathcal{Z}_s$**-Correctness***: If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties output $[s]$ at the time T_{LSh}, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
  - $\mathcal{Z}_a$**-Correctness***: If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties eventually output $[s]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
  - **Privacy***: Irrespective of the network type, the view of the adversary remains independent of s.*
- *If D is corrupt then either no honest party computes any output or there exists some value, say $s^\star$, such that the following hold.*
  - $\mathcal{Z}_s$**-Commitment***: If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties output $[s^\star]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, if any honest party computes its output at the time T, then all honest parties will have their respective output by the time $T + \Delta$.*
  - $\mathcal{Z}_a$**-Commitment***: If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties eventually output $[s^\star]$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.*
- **Communication Complexity***: $\mathcal{O}(|\mathcal{Z}_s| \cdot n^3 \cdot \log |\mathbb{F}| + n^4 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits are communicated by the honest parties.*

Proof. Let us first consider an *honest* dealer. Moreover, we consider a *synchronous* network. Form the $\mathcal{Z}_s$*-correctness* of Π_{Rec} in the *synchronous* network (Lemma 6.2), after time T_{Rec}, the dealer D will reconstruct r, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Moreover, from the *privacy* property of Π_{Rec}, r will be random from the point of the view of the adversary. Since D is *honest*, from the $\mathcal{Z}_s$*-validity* of Π_{BC} in the *synchronous* network (Theorem 3.2), all honest parties will receive $\mathbf{s}$ from the broadcast of D at the time $T_{\mathsf{Rec}} + T_{\mathsf{BC}}$, where $\mathbf{s} = s + r$. The parties then take the default linear secret-sharing $[\mathbf{s}]$ of $\mathbf{s}$ with the IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Since r is also linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, from the linearity property of the secret sharing, it follows that $[\mathbf{s} - r]$ will be the same as a linear secret-sharing of s with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. This proves the $\mathcal{Z}_s$-*correctness*. The *privacy* of s follows since r remains random for the adversary and hence $\mathbf{s}$ *does not* reveal any information about s to the adversary. The $\mathcal{Z}_a$-*correctness* and *privacy* for an *honest* dealer in an *asynchronous* network follows using similar arguments as above, except that we now rely on the $\mathcal{Z}_a$-*correctness* of Π_{Rec} in the *asynchronous* network (Lemma 6.2) and the $\mathcal{Z}_a$-*validity* of Π_{BC} in the *asynchronous* network (Theorem 3.2).

We next consider a *corrupt* dealer and a *synchronous* network. Let P_h be the *first* honest party which computes some output in the protocol, at the time T. This implies that P_h has received some value $\mathbf{s}$ from the broadcast of D. From the $\mathcal{Z}_s$-*consistency* and $\mathcal{Z}_s$-*fallback consistency* of Π_{BC} in the *synchronous* network (Theorem 3.2), it follows that all the honest parties will receive $\mathbf{s}$ from the broadcast of D, latest by the time $T + \Delta$. We define

$$s^{\star} \stackrel{def}{=} \mathbf{s} - r.$$

Since the (honest) parties take the default linear secret-sharing $[\mathbf{s}]$ of $\mathbf{s}$ with the IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets and since r is also linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, from the linearity property of the secret sharing, it follows that $[\mathbf{s} - r]$ will be the same as a linear secret-sharing of $s^{\star}$ with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. This completes the proof of the $\mathcal{Z}_s$-*commitment* in the *synchronous* network. The proof of the $\mathcal{Z}_a$-*commitment* in the *asynchronous* network is similar as above, except that we now rely on the $\mathcal{Z}_a$-*weak consistency* and $\mathcal{Z}_a$-*fallback consistency* of Π_{BC} in the *asynchronous* network (Theorem 3.2) and $\mathcal{Z}_a$-*correctness* of Π_{Rec} in the *asynchronous* network (Lemma 6.2).

In the protocol, one instance of Π_{Rec} with $|\mathcal{R}| = 1$ and one instance of Π_{BC} with $\ell = \log |\mathbb{F}|$ bits are invoked. The communication complexity now follows from the communication complexity of Π_{Rec} (Lemma 6.2) and communication complexity of Π_{BC} (Theorem 3.2).

F.2 Properties of the Protocol $\Pi_{\mathsf{BasicMult}}$

In this section, we prove the properties of the protocol $\Pi_{\mathsf{BasicMult}}$ (see Fig 11 for the formal details). While proving these properties, we will assume that *no honest* party is present in the set $\mathcal{GD}$; looking ahead, this will be ensured in the protocol $\Pi_{\mathsf{RandMultCI}}$ (presented in Section 8.3), where the set $\mathcal{GD}$ is maintained.

We begin by showing that no summand-sharing party during the first two phases are from the discarded set of parties.

Lemma 8.2. *During any instance* $\Pi_{\mathsf{BasicMult}}(\mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, [a], [b], \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$ *of* $\Pi_{\mathsf{BasicMult}}$*, if* $P_j \in \mathsf{Selected}_{\mathsf{iter}}$ *then* $P_j \notin \mathcal{GD}$*, irrespective of the network type.*

Proof. Let P_j be an *arbitrary* party belonging to the set $\mathsf{Selected}_{\mathsf{iter}}$. This implies that P_j is included to $\mathsf{Selected}_{\mathsf{iter}}$, either during Phase I or Phase II. If $P_j \in \mathcal{GD}$, then no honest party will participate with input 1 in the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$ during Phase I for any value of hop and instance $\Pi_{\mathsf{BA}}^{(\mathsf{phII},j)}$ during Phase II. Consequently, from the $\mathcal{Z}_s$*-validity* and $\mathcal{Z}_a$*-validity* of Π_{BA} (Theorem 3.1), party P_j will not be added to $\mathsf{Selected}_{\mathsf{iter}}$, which is a contradiction.

We next show that the first phase will get over for the honest parties after a fixed time in a *synchronous* network and eventually in an asynchronous network. Towards this we show that if the honest parties start any hop during the first phase, then they will complete it after a fixed time in a *synchronous* network and eventually in an asynchronous network.

Lemma 8.3. *Suppose that no honest party is present in $\mathcal{GD}$. If the honest parties start participating during hop number* hop *of Phase I of* $\Pi_{\mathsf{BasicMult}}$ *with iteration number* iter*, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the hop takes $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$ time to complete in a synchronous network, or almost-surely completes eventually in an asynchronous network.*

Proof. Let $Z^\star$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We note that since $\mathcal{Z}_a \subset \mathcal{Z}_s$, *irrespective* of the network type, there exists some set in $\mathbb{S}_{\mathcal{Z}_s}$, say S_h, such that $S_h \in \mathbb{S}_{\mathcal{Z}_s}$. Since the honest parties participate in hop number hop, it implies that there exists *no* $S_q \in \mathbb{S}_{\mathcal{Z}_s}$, such that $\mathsf{SIS}_{\mathsf{iter}}^{(S_q)} = \emptyset$. Particularly, this implies that $\mathsf{SIS}_{\mathsf{iter}}^{(S_h)} \neq \emptyset$. Hence there exists some $P_j \in \mathcal{H}$, such that $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$. This is because if $\mathcal{H} \subseteq \mathsf{Selected}_{\mathsf{iter}}$, then clearly $\mathsf{SIS}_{\mathsf{iter}}^{(S_h)} = \emptyset$ and hence the parties in $\mathcal{H}$ will *not* participate in hop number hop.

Let us first consider a *synchronous* network. During the hop number hop, *every* $P_j \in \mathcal{H}$ such that $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$ will invoke an instance of Π_{LSh} with the input $c_{\mathsf{iter}}^{(j)}$. Then from the $\mathcal{Z}_s$*-correctness* of Π_{LSh} in the *synchronous* network (Lemma 8.1), after time T_{LSh}, the parties in $\mathcal{H}$ output $[c_{\mathsf{iter}}^{(j)}]$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Since $(\mathcal{H} \cap \mathcal{GD}) = \emptyset$, all the parties in $\mathcal{H}$ will participate in the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$ with input 1. Hence from the $\mathcal{Z}_s$*-validity* of Π_{BA} in the *synchronous* network (Theorem 3.1), all the parties in $\mathcal{H}$ output 1 during the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$, after time $T_{\mathsf{LSh}} + T_{\mathsf{BA}}$. Consequently, the parties in $\mathcal{H}$ start participating with input 0 in the remaining Π_{BA} instances $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},k)}$ (if any), for which no input is provided yet. Hence from the $\mathcal{Z}_s$*-security* of Π_{BA} in the *synchronous* network, at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the parties in $\mathcal{H}$ compute some output during the Π_{BA} instances invoked during hop number hop and hence complete the hop number hop.

Next, consider an *asynchronous* network. From the $\mathcal{Z}_a$*-correctness* of Π_{LSh} in the *asynchronous* network (Lemma 8.1), corresponding to *every* $P_j \in \mathcal{H}$ such that $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$, the parties in $\mathcal{H}$ eventually output $[c_{\mathsf{iter}}^{(j)}]$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, during the instance $\Pi_{\mathsf{LSh}}^{(\mathsf{phI},\mathsf{hop},j)}$. Now there are now two cases.

– **Case 1 - There exists some $P_j \in \mathcal{H}$ where $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$ and some party $P_i \in \mathcal{H}$, such that P_i has started participating with input 0 in the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$**: This implies that P_i has computed the output 1 in some Π_{BA} instance during the Phase I, say $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},m)}$. And hence P_i starts participating in all the *remaining* Π_{BA} instances of hop number hop of Phase I (if any) with the input 0. From the $\mathcal{Z}_a$*-consistency* of Π_{BA} in the *asynchronous* network, all the parties in $\mathcal{H}$ will also eventually compute the output 1 during the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},m)}$ and will start participating in all the *remaining* Π_{BA} instances of hop number hop of Phase I (if any) with the input 0. Consequently, from the $\mathcal{Z}_a$*-security* of Π_{BA} in the *asynchronous* network (Theorem 3.1), almost-surely, the parties in $\mathcal{H}$ eventually compute some output during all the Π_{BA} instances of hop number hop and hence complete the hop number hop.

– **Case 2 - No honest party has yet started participating with input 0 in any of the BA instances of Phase I corresponding to the honest parties**: In this case, the honest parties will eventually start participating with input 1 in the Π_{BA} instances $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$, corresponding to the parties $P_j \in \mathcal{H}$ where $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$. Hence from the $\mathcal{Z}_a$*-validity* of Π_{BA} in the *asynchronous* network (Theorem 3.1), it follows that eventually, there will be some $P_j \in \mathcal{H}$ where $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$, such that all the parties in $\mathcal{H}$ compute the output 1 during the instance $\Pi_{\mathsf{BA}}^{(\mathsf{phI},\mathsf{hop},j)}$. The rest of the proof is similar to what is argued in the previous case.

We next show that in protocol $\Pi_{\mathsf{BasicMult}}$, the honest parties compute some output, after a fixed time in a *synchronous* network and eventually in an *asynchronous* network.

Lemma 8.4. *If no honest party is present in $\mathcal{GD}$, then in protocol $\Pi_{\mathsf{BasicMult}}$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all honest parties compute some output by the time $T_{\mathsf{BasicMult}} = (2n+1) \cdot T_{\mathsf{BA}} + (n+1) \cdot T_{\mathsf{LSh}} + T_{\mathsf{Rec}}$ in a synchronous network, or almost-surely, eventually in an asynchronous network.*

Proof. In the protocol $\Pi_{\mathsf{BasicMult}}$, to compute an output, the (honest) parties need to complete the three phases. We first show that *irrespective* of the network type, the honest parties will complete Phase II, provided they complete Phase I. This is because the honest parties will participate in *all* the Π_{BA} instances $\Pi_{\mathsf{BA}}^{(\mathsf{phII},j)}$ of phase II with *some* input, after waiting *exactly* for time T_{LSh}. Consequently, once phase I is completed, from the $\mathcal{Z}_s$*-security* of Π_{BA} in the *synchronous* network (Theorem 3.1), it takes $T_{\mathsf{LSh}} + T_{\mathsf{BA}}$ time for the honest parties to compute outputs in the Π_{BA} instances during phase II in a *synchronous* network. On the other hand, the $\mathcal{Z}_a$*-security* of Π_{BA} in the *asynchronous* network (Theorem 3.1) guarantees that almost-surely, all honest parties eventually compute some output during the Π_{BA} instances during phase II in an asynchronous network. Now once Phase I and Phase II are completed, it takes T_{Rec} time for the parties to complete Phase III in a *synchronous* network (follows from Lemma 6.2), while in an *asynchronous* network, it gets completed eventually (see Lemma 6.2).

We now show that Phase I always gets completed for the honest parties. Lemma 8.3 guarantees that if the honest parties start any hop during Phase I, then it gets completed for all the honest parties after time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$ in a *synchronous* network or eventually in an *asynchronous* network. From the protocol steps, it follows that there can be *at most* n hops during Phase I. This is because once the set of *honest* parties are included in the set of summand-sharing parties $\mathsf{Selected}_{\mathsf{iter}}$ during Phase I, then the parties will exit Phase I.

We next show that the adversary does not learn anything additional about a and b during the protocol.

Lemma 8.5. *If no honest party is present in $\mathcal{GD}$, then the view of the adversary remains independent of a and b throughout the protocol, irrespective of the network type.*

Proof. Let $Z^\star$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. The view of the adversary remains independent of a and b during Phase I and Phase II. This is because the *privacy* of Π_{LSh} (Lemma 8.1) ensures that the view of the adversary remains independent of the summand-sums shared by the *honest* summand-sharing parties during Phase I and Phase II. Let $S_h \in \mathbb{S}_{\mathcal{Z}_s}$ be the group consisting of *only* honest parties; i.e. $(S_h \cap Z^\star) = \emptyset$. To prove that the view of the adversary remains independent of a and b during Phase III, we show that *irrespective* of the network type, any summand of the form (h, q) or (p, h) will *not* be present in $\mathsf{SIS}_{\mathsf{iter}}$ during this phase corresponding to any $p, q \in \{1, \ldots, |\mathcal{Z}_s|\}$, implying that the shares $[a]_h$ and $[b]_h$ *does not* get publicly reconstructed. Note that $(h, q) \notin \mathsf{SIS}^{(j)}_{\mathsf{iter}}$ for any $P_j \in Z^\star$, as otherwise it would imply that a *corrupt* $P_j \in (S_h \cap Z^\star)$, which is a contradiction. Similarly, $(p, h) \notin \mathsf{SIS}^{(j)}_{\mathsf{iter}}$ for any $P_j \in Z^\star$

Let us first consider a *synchronous* network and consider an *arbitrary* ordered pair (h, q). Consider the case when (h, q) has *not been* removed from $\mathsf{SIS}_{\mathsf{iter}}$ during Phase I in any of the hops. Then, during Phase II, the pair (h, q) will get *statically* re-assigned to the $\mathsf{SIS}^{(j)}_{\mathsf{iter}}$ set of some $P_j \in \mathcal{H}$, such that $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$. From the protocol steps, P_j will include the summand $[a]_h \cdot [b]_q$ while computing $c^{(j)}_{\mathsf{iter}}$ and share $c^{(j)}_{\mathsf{iter}}$ through an instance of Π_{LSh}. From the $\mathcal{Z}_s$*-correctness* of Π_{LSh} in the *synchronous* network, the parties in $\mathcal{H}$ will output $[c^{(j)}_{\mathsf{iter}}]$ during the instance of Π_{LSh} invoked by P_j. Since $P_j \notin (\mathsf{Selected}_{\mathsf{iter}} \cup \mathcal{GD})$, it follows that all the parties in $\mathcal{H}$ will participate in the instance $\Pi^{(\mathsf{phII}, j)}_{\mathsf{BA}}$ with input 1 and compute the output 1. Consequently, (h, q) will be removed from the updated $\mathsf{SIS}_{\mathsf{iter}}$ during Phase II, if not removed during Phase I. By the same logic, any ordered pair of the form (p, h) will also be removed from $\mathsf{SIS}_{\mathsf{iter}}$, by the end of Phase II.

Next, consider an *asynchronous* network and an *arbitrary* (h, q). In this case, we show that (h, q) will eventually be removed from $\mathsf{SIS}_{\mathsf{iter}}$ during Phase I itself. Let the parties in $\mathcal{H}$ complete Phase I. This implies that there exists some $S_\ell \in \mathbb{S}_{\mathcal{Z}_s}$ such that $\mathsf{SIS}^{(S_\ell)}_{\mathsf{iter}} = \emptyset$ when Phase I gets over. We show that (h, q) was present in $\mathsf{SIS}^{(S_\ell)}_{\mathsf{iter}}$ at the beginning of Phase I, when the parties initialize $\mathsf{SIS}^{(S_\ell)}_{\mathsf{iter}}$.

That is, at the time of initialization, there was some $P_j \in (S_\ell \cap \mathcal{H} \cap S_q)$ such that $(h, q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter}}$. For this, it is enough to show that $(S_\ell \cap \mathcal{H} \cap S_q) \neq \emptyset$, which follows from the fact that $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the condition $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ and $(S_\ell \cap \mathcal{H} \cap S_q) = \mathcal{P} \setminus (Z_\ell \cup Z^\star \cup Z_q)$, where $Z^\star \in \mathcal{Z}_a$ (because we are considering an *asynchronous* network) and $Z_\ell, Z_q \in \mathcal{Z}_s$ (follows from the construction of $\mathbb{S}_{\mathcal{Z}_s}$). Let $P_j \in (S_\ell \cap \mathcal{H} \cap S_q)$. Hence at the time of the initialization, $(h, q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter}}$, which further implies that $(h, q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter}} \subseteq \mathsf{SIS}^{(S_\ell)}_{\mathsf{iter}}$ at the time of initialization. And hence (h, q) must have been removed from $\mathsf{SIS}_{\mathsf{iter}}$ by the end of Phase I, since P_j would have been selected as the summand-sharing party in one of the hops during Phase I. The same logic also applies to any arbitrary (p, h), implying that (p, h) would have been removed from $\mathsf{SIS}_{\mathsf{iter}}$ by the end of Phase I itself.

We next show that if all the parties behave honestly in the protocol, the parties output a linear-secret sharing of $a \cdot b$ with IC-signatures.

Lemma 8.6. *If no honest party is present in $\mathcal{GD}$ and if all parties in $\mathcal{P} \setminus \mathcal{GD}$ behave honestly, then in protocol $\Pi_{\mathsf{BasicMult}}$, the honest parties output a linear secret-sharing of $a \cdot b$ with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, irrespective of the network type.*

Proof. Note that $(\mathsf{Selected}_{\mathsf{iter}} \cap \mathcal{GD}) = \emptyset$, which follows from Lemma 8.2. Now Consider an *arbitrary* $(p, q) \in \mathsf{SIS}_{\mathsf{iter}}$. We claim that *irrespective* of the network type, if all the parties behave honestly, then the summand $[a]_p \cdot [b]_q$ is considered on behalf of *exactly* one party while secret-sharing the summand-sums. That is there is *exactly* one P_j across the three phases, such that the following hold:

$$c^{(j)}_{\mathsf{iter}} = \ldots + [a]_p[b]_q + \ldots.$$

Assuming that the claim is true, the proof then follows from the linearity property of secret-sharing and the fact that $c_{\mathsf{iter}} = c^{(1)}_{\mathsf{iter}} + \ldots + c^{(n)}_{\mathsf{iter}}$ holds. That is, corresponding to each $P_j \in \mathsf{Selected}_{\mathsf{iter}}$, the value $c^{(j)}_{\mathsf{iter}}$ is secret-shared through an instance of Π_{LSh}, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, which guarantees that $c^{(j)}_{\mathsf{iter}}$ is linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, by the end of Phase II, the parties will be publicly knowing the set $\mathsf{Selected}_{\mathsf{iter}}$, which follows from the security properties of Π_{BA}. Furthermore, after Phase II, the parties will be publicly reconstructing the shares $[a]_p$ and $[b]_q$ corresponding to every (p, q), which are still present in $\mathsf{SIS}_{\mathsf{iter}}$. Since corresponding to each $P_j \in \mathcal{P} \setminus \mathsf{Selected}_{\mathsf{iter}}$, the parties take the default linear secret-sharing of $c^{(j)}_{\mathsf{iter}}$ with IC-signatures and $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, it follows that c will be linearly secret-shared with IC-signatures and $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, $c = a \cdot b$ holds, since each summand $[a]_p \cdot [b]_q$ is considered exactly once, as per our claim. We now proceed to prove our claim.

We first show that there exists *at least one* P_j in one of the three phases, such that the following holds:

$$c^{(j)}_{\mathsf{iter}} = \ldots + [a]_p[b]_q + \ldots.$$

For this, consider the following cases.

- **Case 1 — During Phase I, there is some $P_j \in \mathsf{Selected}_{\mathsf{iter}}$, such that (p, q) was present in the set $\mathsf{SIS}^{(j)}_{\mathsf{iter}}$ when P_j was added to $\mathsf{Selected}_{\mathsf{iter}}$**: In this case, there is nothing to show.
- **Case 2 — At the end of Phase I, (p, q) is still *present* in $\mathsf{SIS}_{\mathsf{iter}}$**: This implies that at the end of Phase I, $(\mathsf{Selected}_{\mathsf{iter}} \cap S_p \cap S_q) = \emptyset$. Since $\mathcal{Z}_s$ and $\mathcal{Z}_a$ satisfy the condition $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$, it follows that $(S_p \cap S_q) \neq \emptyset$. Let $P_j \stackrel{def}{=} \min(S_p, S_q)$. Note that $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$ at the end of Phase I, otherwise (p, q) would have been *removed* from $\mathsf{SIS}_{\mathsf{iter}}$, which is a contradiction. Now if $P_j \notin \mathcal{GD}$, then clearly the summand $[a]_p \cdot [b]_q$ will be present in $c^{(j)}_{\mathsf{iter}}$, shared by P_j during the Phase II. Else, during Phase III, the parties will publicly reconstruct $[a]_p$ and $[b]_q$ and consequently $[a]_p \cdot [b]_q$ will be present in the default secret-sharing of $c^{(j)}_{\mathsf{iter}}$, taken on the behalf of P_j.

To complete the proof of our claim, we next show that (p, q) *cannot* be present in the summand-sum of *more than one* party across the three phases. On *contrary*, let P_j and P_k be two *distinct* parties, such that the following holds across the three phases:

$$c^{(j)}_{\mathsf{iter}} = \ldots + [a]_p[b]_q + \ldots \quad \wedge \quad c^{(k)}_{\mathsf{iter}} = \ldots + [a]_p[b]_q + \ldots$$

Now there are three following cases.

- **Case 1 - $P_j, P_k \in \mathsf{Selected}_{\mathsf{iter}}$ at the end of Phase I**: From the security properties of Π_{BA}, the parties will agree on which party to add to $\mathsf{Selected}_{\mathsf{iter}}$ during every hop during Phase I. Moreover, from the protocol steps, exactly one party is selected as a summand-sharing party and added to $\mathsf{Selected}_{\mathsf{iter}}$ in each hop. Suppose that P_j was added to $\mathsf{Selected}_{\mathsf{iter}}$ during $\mathsf{hop}^{(j)}$, and that P_k was added to $\mathsf{Selected}_{\mathsf{iter}}$ during $\mathsf{hop}^{(k)}$. Moreover, without loss of generality, let $\mathsf{hop}^{(j)} < \mathsf{hop}^{(k)}$. From the protocol steps, it follows that $P_j, P_k \notin \mathcal{GD}$, as otherwise no honest party would have voted for P_j and P_k as a candidate summand-sharing party and consequently, $P_j, P_k \notin \mathsf{Selected}_{\mathsf{iter}}$. Now as per the lemma conditions, all the parties (including P_k) behave honestly. Hence, the ordered pair (p, q) would be removed from $\mathsf{SIS}^{(k)}_{\mathsf{iter}}$ at the end of $\mathsf{hop}^{(j)}$. Consequently, P_k will *not include* the summand $[a]_p \cdot [b]_q$ while computing $c^{(k)}_{\mathsf{iter}}$ during hop number $\mathsf{hop}^{(k)}$, which is a contradiction.
- **Case 2 — $P_j \in \mathsf{Selected}_{\mathsf{iter}}$ at the end of Phase I and $P_k \notin \mathsf{Selected}_{\mathsf{iter}}$ at the end of Phase I**: In this case, (p, q) will be *removed* from $\mathsf{SIS}_{\mathsf{iter}}$ at the end of Phase I. Hence, it *cannot* get re-assigned to any other party P_k after Phase I and hence *cannot* belong to $\mathsf{SIS}^{(k)}_{\mathsf{iter}}$. Consequently, the summand $[a]_p \cdot [b]_q$ will *not* be considered while computing $c^{(k)}_{\mathsf{iter}}$, which is a contradiction,
- **Case 3 - $P_j \notin \mathsf{Selected}_{\mathsf{iter}}$ and $P_k \notin \mathsf{Selected}_{\mathsf{iter}}$ at the end of Phase I**: In this case, the summand $[a]_p \cdot [b]_p$ is *deterministically* and *statically* re-assigned to the least-indexed party from the set $(S_p \cap S_q)$, as per the min function. Hence (p, q) will be re-assigned to either P_j or P_k, but *not both.*

Consequently, the summand $[a]_p \cdot [b]_q$ will be considered while computing either $c^{(j)}_{\mathsf{iter}}$ or $c^{(k)}_{\mathsf{iter}}$, but not both, which is a contradiction.

Lemma 8.7. *Protocol $\Pi_{\mathsf{BasicMult}}$ incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot \log |\mathbb{F}| + n^6 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^2)$ calls to Π_{BA}.*

Proof. During Phase I, there can be $\mathcal{O}(n)$ hops, where during each hop, a party from $\mathcal{P} \setminus \mathcal{GD}$ secret-shares a field element through an instance of Π_{LSh}. Moreover, n instances of Π_{BA} are invoked to agree upon the summand-sharing party of the hop. Hence, $\mathcal{O}(n^2)$ instances of Π_{LSh} and $\mathcal{O}(n^2)$ instances of Π_{BA} are required during Phase I. During Phase II, $\mathcal{O}(n)$ instances of Π_{LSh} and $\mathcal{O}(n)$ instances of Π_{BA} are required. Finally during Phase III, up to $\mathcal{O}(|\mathcal{Z}_s|)$ shares need to be publicly reconstructed. The communication complexity now follows from the communication complexity of Π_{LSh} (Lemma 8.1) and communication complexity of Π_{RecShare} with $|\mathcal{R}| = n$ (Lemma 6.1).

As a corollary of Lemma 8.7, we derive the following corollary, which determines the *maximum* number of instances of Π_{LSh} which are invoked during an instance of $\Pi_{\mathsf{BasicMult}}$. Looking ahead, this will be useful to later calculate the maximum number of instances of Π_{LSh} which need to be invoked as part of our final multiplication protocol. This will be further useful to determine the number of linearly secret-shared values with IC-signatures and core-sets $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$, which need to be generated through the protocol Π_{Rand} beforehand.

Corollary 8.1. *During any instance of $\Pi_{\mathsf{BasicMult}}$, there can be at most $n^2 + n$ instances of Π_{LSh} invoked.*

Proof. The proof follows from the fact that during Phase I, there can be up to n^2 instances of Π_{LSh}, if $\mathcal{GD} = \emptyset$ and during Phase II, there can be up to $n - 1$ instances of Π_{LSh}, if only one party is added to $\mathsf{Selected}_{\mathsf{iter}}$ during Phase I.

F.3 Properties of the Protocol $\Pi_{\mathsf{RandMultCI}}$

In this section, we prove the properties of the protocol $\Pi_{\mathsf{RandMultCI}}$ (see Fig 12 for the formal description). We begin by showing that irrespective of the network type, the (honest) parties compute linearly secret-shared $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} with IC-signatures, which are random from the point of view of the adversary.

Lemma 8.8. *In protocol $\Pi_{\mathsf{RandMultCI}}$, the following hold.*
- **Synchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, honest parties will have linearly secret-shared $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, by the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$. Moreover, adversary's view is independent of $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter}.*
- **Asynchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, honest parties will eventually have linearly secret-shared $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, adversary's view is independent of $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter}.*

Proof. We first consider a *synchronous* network. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Corresponding to each $P_j \in \mathcal{H}$, the honest parties compute the output $[a^{(j)}_{\mathsf{iter}}], [b^{(j)}_{\mathsf{iter}}], [b'^{(j)}_{\mathsf{iter}}]$ and $[r^{(j)}_{\mathsf{iter}}]$ during the instances of Π_{LSh} invoked by P_j at the time T_{LSh}, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. This follows from the $\mathcal{Z}_s$*-correctness* of Π_{LSh} in the *synchronous* network (Lemma 8.1). Consequently, at the time T_{LSh}, all the parties in $\mathcal{H}$ will be present in the set $\mathcal{C}_i$ of every $P_i \in \mathcal{H}$. Hence corresponding to each $P_j \in \mathcal{H}$, each $P_i \in \mathcal{H}$ starts participating with input 1 in the instance $\Pi^{(j)}_{\mathsf{BA}}$ at the time T_{LSh}. Hence from the $\mathcal{Z}_s$*-validity* and $\mathcal{Z}_s$*-guaranteed liveness* properties of Π_{BA} in the *synchronous* network (Theorem 3.1), it follows that at the time $T_{\mathsf{LSh}}+T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ compute the output 1 during the instance $\Pi^{(j)}_{\mathsf{BA}}$, corresponding to every $P_j \in \mathcal{H}$. Consequently, at the time $T_{\mathsf{LSh}} + T_{\mathsf{BA}}$, all the parties in $\mathcal{H}$ 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 (Theorem 3.1), these Π_{BA} instances will produce common outputs for every honest party by the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$. Hence, at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the honest parties will have a common $\mathcal{CS}$, where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$ and where $\mathcal{H} \subseteq \mathcal{CS}$. We next wish to show that corresponding to *every* $P_j \in \mathcal{CS}$, there exists some quadruplet of values, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

Consider an *arbitrary* party $P_j \in \mathcal{CS}$. If $P_j \in \mathcal{H}$, then whatever we wish to show is correct, as shown above. Next, consider a *corrupt* $P_j \in \mathcal{CS}$. Since $P_j \in \mathcal{CS}$, it follows that the instance $\Pi^{(j)}_{\mathsf{BA}}$ 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 instances of Π_{LSh} invoked by P_j, by the time $T_{\mathsf{LSh}} + T_{\mathsf{BA}}$ (implying that $P_j \in \mathcal{C}_i$) and participated with input 1 in the instance $\Pi^{(j)}_{\mathsf{BA}}$. Otherwise, *all honest* parties would participate with input 0 in the instance $\Pi^{(j)}_{\mathsf{BA}}$ at the time $T_{\mathsf{LSh}}+T_{\mathsf{BA}}$ and then from the $\mathcal{Z}_s$*-validity* of Π_{BA} in the *synchronous* network, every honest party would have computed the output 0 in the instance $\Pi^{(j)}_{\mathsf{BA}}$ and hence P_j will not be present in $\mathcal{CS}$, which is a contradiction. Now if P_i has computed some output during the instances of Π_{LSh} invoked by P_j at the time $T_{\mathsf{LSh}} + T_{\mathsf{BA}}$, then from the $\mathcal{Z}_s$*-commitment* of Π_{LSh} in the *synchronous* network (Lemma 8.1), it follows that except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, there exist values $(a^{(j)}_{\mathsf{iter}}, b^{(j)}_{\mathsf{iter}}, b'^{(j)}_{\mathsf{iter}}, r^{(j)}_{\mathsf{iter}})$, which will be linearly secret-shared with IC-signature, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, by the time $T_{\mathsf{LSh}}+T_{\mathsf{BA}}+\Delta$. Since $\Delta < T_{\mathsf{BA}}$, it follows that by the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the honest parties will have $[a^{(j)}_{\mathsf{iter}}], [b^{(j)}_{\mathsf{iter}}], [b'^{(j)}_{\mathsf{iter}}]$ and $[r^{(j)}_{\mathsf{iter}}]$. From the linearity property of secret-sharing, it then follows that by the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the values $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} will be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

From the *privacy* of Π_{LSh} (Lemma 8.1), the view of the adversary will be independent of the values $(a^{(j)}_{\mathsf{iter}}, b^{(j)}_{\mathsf{iter}}, b'^{(j)}_{\mathsf{iter}}, r^{(j)}_{\mathsf{iter}})$, corresponding to the parties $P_j \in \mathcal{H}$. As $(\mathcal{H} \cap \mathcal{CS}) \neq \emptyset$, it follows that $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} will be indeed random from the point of view of the adversary. This completes the proof for the case of *synchronous* network.

We next consider an *asynchronous* network. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Notice that $\mathcal{P} \setminus \mathcal{H} \in \mathcal{Z}_s$, since $\mathcal{Z}_a \subset \mathcal{Z}_s$. Now irrespective of the way messages are scheduled, there will be eventually a subset of parties $\mathcal{P} \setminus Z$ 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 $\mathcal{P} \setminus Z$. This is because, corresponding to every $P_j \in \mathcal{H}$, all the parties in $\mathcal{H}$ *eventually* compute some output during the instances of Π_{LSh} invoked by P_j except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, which follows from the $\mathcal{Z}_a$*-correctness* of Π_{LSh} in the *asynchronous* network (Lemma 8.1). So even if the *corrupt* parties P_j do not invoke their respective instances of Π_{LSh}, there will be a set of Π_{BA} instances corresponding to the parties in $\mathcal{P} \setminus Z$ 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 (Theorem 3.1), 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 instances of Π_{LSh} invoked by P_j. If P_j is *honest*, then the $\mathcal{Z}_a$*-correctness* of Π_{LSh} in the *asynchronous* network guarantees that except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the values $(a_{\mathsf{iter}}^{(j)}, b_{\mathsf{iter}}^{(j)}, b_{\mathsf{iter}}^{\prime(j)}, r_{\mathsf{iter}}^{(j)})$ will be *eventually* linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. On the other hand, even if P_j is *corrupt*, the $\mathcal{Z}_a$*-commitment* of Π_{LSh} in the *asynchronous* network (Lemma 8.1) guarantees that there exist values $(a_{\mathsf{iter}}^{(j)}, b_{\mathsf{iter}}^{(j)}, b_{\mathsf{iter}}^{\prime(j)}, r_{\mathsf{iter}}^{(j)})$ which are *eventually* linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. From the linearity property of secret-sharing, it then follows that eventually, the values $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} will be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. The privacy of $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}$ and r_{iter} is similar as for the synchronous communication network and the fact that $(\mathcal{H} \cap \mathcal{CS}) \neq \emptyset$ still holds in the *asynchronous* network.

We next claim that all honest parties will eventually agree on whether the instances of $\Pi_{\mathsf{BasicMult}}$ in $\Pi_{\mathsf{RandMultCI}}$ have succeeded or failed.

Lemma 8.9. *Consider an arbitrary* iter*, such that all honest parties participate in the instance* $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$*, where no honest party is present in* $\mathcal{GD}$*. Then except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, all honest parties reconstruct a (common) value* d_{iter} *and set* $\mathsf{flag}_{\mathsf{iter}}$ *to a common*

Boolean value, at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ *in a synchronous network, or eventually in an asynchronous network.*

Proof. From Lemma 8.8, the honest parties have $[a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [b'_{\mathsf{iter}}]$ and $[r_{\mathsf{iter}}]$ at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$ in a synchronous network, or eventually in an asynchronous network, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. From Lemma 8.4, it follows that the honest parties have $\{[c^{(1)}_{\mathsf{iter}}], \ldots, [c^{(n)}_{\mathsf{iter}}], [c_{\mathsf{iter}}]\}$ and $\{[c'^{(1)}_{\mathsf{iter}}], \ldots, [c'^{(n)}_{\mathsf{iter}}], [c'_{\mathsf{iter}}]\}$ from the corresponding instances of $\Pi_{\mathsf{BasicMult}}$, either after time $T_{\mathsf{BasicMult}}$ or eventually, based on the network type, where $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ are the underlying core-sets. From Lemma 6.2, the honest parties reconstruct r_{iter} from the corresponding instance of Π_{Rec} after time T_{Rec} in a *synchronous* network or eventually in an *asynchronous* network, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. From the linearity property of secret-sharing, it then follows that the honest parties compute $[e_{\mathsf{iter}}]$ and hence reconstruct e_{iter} from the corresponding instance of Π_{Rec}, after time T_{Rec} in a *synchronous* network or eventually in an *asynchronous* network, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Moreover, $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ will be the underlying core-sets for $[e_{\mathsf{iter}}]$. Again, from the linearity property of secret-sharing, it follows that the honest parties compute $[d_{\mathsf{iter}}]$ with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, followed by reconstructing d_{iter} from the corresponding instance of Π_{Rec}, which takes T_{Rec} time in a *synchronous* network or happens eventually in an *asynchronous* network. Thus, the honest parties will have d_{iter} either at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network, or eventually in an asynchronous network. Now depending upon the value of d_{iter}, the honest parties set $\mathsf{flag}_{\mathsf{iter}}$ to either 0 or 1.

We next claim that if no cheating occurs, then the honest parties output a multiplication-triple, which is linearly secret-shared with IC-signatures.

Lemma 8.10. *Consider an arbitrary* iter*, such that all honest parties participate in the instance* $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$*, where no honest party is present in* $\mathcal{GD}$*. If no party in* $\mathcal{P} \setminus \mathcal{GD}$ *behaves maliciously, then* $d_{\mathsf{iter}} = 0$ *and the honest parties output* $([a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ *at the time* $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ *in a synchronous network or eventually in an asynchronous network, where* $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ *and where* $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ *are the underlying core-sets*

Proof. If no party in $\mathcal{P} \setminus \mathcal{GD}$ behaves maliciously, then from Lemma 8.6, the honest parties compute $[c_{\mathsf{iter}}]$ and $[c'_{\mathsf{iter}}]$ from the respective instances of $\Pi_{\mathsf{BasicMult}}$, such that $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ and $c'_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ holds and where $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ are the underlying core-sets. Moreover, from Lemma 8.9, the honest parties will compute d_{iter} at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network. Furthermore, if $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ and $c'_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ holds, the value d_{iter} will be 0 and consequently, the honest parties will output $([a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$. Furthermore, it is easy to see that $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ will be the underlying core-sets. This is because all the secret-shared values in the protocol are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

We next show that if $d_{\mathsf{iter}} \neq 0$, then the honest parties include at least one *new* maliciously-corrupt party in the set $\mathcal{GD}$.

Lemma 8.11. *Consider an arbitrary* iter*, such that all honest parties participate in the instance* $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$*, where no honest party is present in* $\mathcal{GD}$*. If* $d_{\mathsf{iter}} \neq 0$*, then except with probability* $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$*, the honest parties update* $\mathcal{GD}$ *by adding a new maliciously-corrupt party in* $\mathcal{GD}$*, either at the time* $T_{\mathsf{RandMultCI}} = T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 4T_{\mathsf{Rec}}$ *in a synchronous network or eventually in an asynchronous network.*

Proof. Let $d_{\mathsf{iter}} \neq 0$ and let $\mathsf{Selected}_{\mathsf{iter}}$ be the set of summand-sharing parties across the two instances of $\Pi_{\mathsf{BasicMult}}$ executed in $\Pi_{\mathsf{RandMultCI}}$. That is:

$$\mathsf{Selected}_{\mathsf{iter}} \stackrel{def}{=} \mathsf{Selected}_{\mathsf{iter},c} \cup \mathsf{Selected}_{\mathsf{iter},c'}.$$

Note that there exists no $P_j \in \mathsf{Selected}_{\mathsf{iter}}$ such that $P_j \in \mathcal{GD}$, which follows from Lemma 8.2. We claim that there exists at least one party $P_j \in \mathsf{Selected}_{\mathsf{iter}}$, such that corresponding to $c^{(j)}_{\mathsf{iter}}$ and $c'^{(j)}_{\mathsf{iter}}$, the following holds:

$$r_{\mathsf{iter}} \cdot c^{(j)}_{\mathsf{iter}} + c'^{(j)}_{\mathsf{iter}} \neq r_{\mathsf{iter}} \cdot \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q + \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

Assuming the above holds, the proof now follows from the fact that once the parties reconstruct $d_{\mathsf{iter}} \neq 0$, they proceed to reconstruct the shares $\{[a_{\mathsf{iter}}]_q, [b_{\mathsf{iter}}]_q, [b'_{\mathsf{iter}}]_q\}_{S_q \in \mathbb{S}_{\mathcal{Z}_s}}$ through appropriate instances of Π_{RecShare} and the values $c^{(1)}_{\mathsf{iter}}, \ldots, c^{(n)}_{\mathsf{iter}}, c'^{(1)}_{\mathsf{iter}}, \ldots, c'^{(n)}_{\mathsf{iter}}$ through appropriate instances of Π_{Rec}. From Lemma 8.9, Lemma 6.1 and Lemma 6.2, this happens by the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 4T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Upon reconstructing these values, party P_j will be included in the set $\mathcal{GD}$. Moreover, it is easy to see that P_j is a maliciously-corrupt party since, for every *honest* $P_j \in \mathsf{Selected}_{\mathsf{iter}}$, the following conditions hold:

$$c^{(j)}_{\mathsf{iter}} = \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q \quad \text{and} \quad c'^{(j)}_{\mathsf{iter}} = \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

We prove the above claim through a contradiction. Let the following condition hold for *each* $P_j \in \mathsf{Selected}_{\mathsf{iter}}$:

$$r_{\mathsf{iter}} \cdot c^{(j)}_{\mathsf{iter}} + c'^{(j)}_{\mathsf{iter}} = r_{\mathsf{iter}} \cdot \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q + \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

Next, summing the above equation over all $P_j \in \mathsf{Selected}_{\mathsf{iter}}$, we get that the following holds:

$$\sum_{P_j \in \mathsf{Selected}_{\mathsf{iter}}} r_{\mathsf{iter}} \cdot c^{(j)}_{\mathsf{iter}} + c'^{(j)}_{\mathsf{iter}} = \sum_{P_j \in \mathsf{Selected}_{\mathsf{iter}}} r_{\mathsf{iter}} \cdot \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q + \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

This implies that the following holds:

$$r_{\mathsf{iter}} \cdot \sum_{P_j \in \mathsf{Selected}_{\mathsf{iter}}} c^{(j)}_{\mathsf{iter}} + c'^{(j)}_{\mathsf{iter}} = r_{\mathsf{iter}} \cdot \sum_{P_j \in \mathsf{Selected}_{\mathsf{iter}}} \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c}} [a_{\mathsf{iter}}]_p [b_{\mathsf{iter}}]_q + \sum_{(p,q) \in \mathsf{SIS}^{(j)}_{\mathsf{iter},c'}} [a_{\mathsf{iter}}]_p [b'_{\mathsf{iter}}]_q.$$

Now based on the way $a_{\mathsf{iter}}, b_{\mathsf{iter}}, b'_{\mathsf{iter}}, c_{\mathsf{iter}}$ and c'_{iter} are defined, the above implies that the following holds:

$$r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} + c'_{\mathsf{iter}} = r \cdot a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} + a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$$

This further implies that

$$r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} + c'_{\mathsf{iter}} = (r_{\mathsf{iter}} \cdot b_{\mathsf{iter}} + b'_{\mathsf{iter}}) \cdot a_{\mathsf{iter}}$$

Since in the protocol $e_{\mathsf{iter}} \stackrel{def}{=} r_{\mathsf{iter}} \cdot b_{\mathsf{iter}} + b'_{\mathsf{iter}}$, the above implies that

$$r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} + c'_{\mathsf{iter}} = e_{\mathsf{iter}} \cdot a_{\mathsf{iter}} \quad \Rightarrow \; e_{\mathsf{iter}} \cdot a_{\mathsf{iter}} - r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} - c'_{\mathsf{iter}} = 0 \quad \Rightarrow \; d_{\mathsf{iter}} = 0,$$

where the last equality follows from the fact that in the protocol, $d_{\mathsf{iter}} \stackrel{def}{=} e_{\mathsf{iter}} \cdot a_{\mathsf{iter}} - r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} - c'_{\mathsf{iter}}$. However $d_{\mathsf{iter}} = 0$ is a contradiction since, according to the hypothesis of the lemma, we are given that $d_{\mathsf{iter}} \neq 0$.

We next show that if the honest parties output a secret-shared triple in the protocol, then except with probability $\frac{1}{|\mathbb{F}|}$, the triple is a multiplication-triple. Moreover, the triple will be random for the adversary.

Lemma 8.12. *Consider an arbitrary* iter*, such that all honest parties participate in the instance* $\Pi_{\mathsf{RandMultCI}}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a, \mathbb{S}_{\mathcal{Z}_s}, \mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, \mathcal{GD}, \mathsf{iter})$*, where no honest party is present in* $\mathcal{GD}$*. If* $d_{\mathsf{iter}} = 0$*, then the honest parties output linearly secret-shared* $([a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ *with IC-signatures with* $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ *being the underlying core-sets, at the time* $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ *in a synchronous network or eventually in an asynchronous network where, except with probability* $\frac{1}{|\mathbb{F}|}$*, the condition* $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ *holds. Moreover, the view of* Adv *will be independent of* $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$*.*

Proof. Let $d_{\mathsf{iter}} = 0$. From Lemma 8.4, all honest parties will agree that $d_{\mathsf{iter}} = 0$, either at the time $T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 3T_{\mathsf{Rec}}$ in a synchronous network or eventually in an asynchronous network. Then, from the protocol steps, the honest parties output $([a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. In the protocol $d_{\mathsf{iter}} \stackrel{def}{=} e_{\mathsf{iter}} \cdot a_{\mathsf{iter}} - r_{\mathsf{iter}} \cdot c_{\mathsf{iter}} - c'_{\mathsf{iter}}$, where $e_{\mathsf{iter}} \stackrel{def}{=} r_{\mathsf{iter}} \cdot b_{\mathsf{iter}} + b'_{\mathsf{iter}}$. Since $d_{\mathsf{iter}} = 0$ holds, it implies that the honest parties have verified that the following holds:

$$r_{\mathsf{iter}}(a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} - c_{\mathsf{iter}}) = (c'_{\mathsf{iter}} - a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}).$$

We note that r_{iter} will be a random element from $\mathbb{F}$ and will be unknown to Adv till it is publicly reconstructed, which follows from Lemma 8.8 We also note that r_{iter} will be unknown to Adv, till the outputs for the underlying instances of $\Pi_{\mathsf{BasicMult}}$ are computed, and the honest parties have $[c_{\mathsf{iter}}]$ and $[c'_{\mathsf{iter}}]$. This is because, in the protocol, the honest parties start participating in the instance of Π_{Rec} to reconstruct r_{iter}, only after they compute $[c_{\mathsf{iter}}]$ and $[c'_{\mathsf{iter}}]$. Now we have the following cases with respect to whether any party from $\mathcal{P} \setminus \mathcal{GD}$ behaved maliciously during the underlying instances of $\Pi_{\mathsf{BasicMult}}$.

- **Case I:** $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ **and** $c'_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ — In this case, $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$ is a multiplication-triple.
- **Case II:** $c_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$**, but** $c'_{\mathsf{iter}} \neq a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ — This case is never possible, as this will lead to the contradiction that $r_{\mathsf{iter}}(a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} - c_{\mathsf{iter}}) \neq (c'_{\mathsf{iter}} - a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}})$ holds.
- **Case III:** $c_{\mathsf{iter}} \neq a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$**, but** $c'_{\mathsf{iter}} = a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ — This case is possible only if $r_{\mathsf{iter}} = 0$, as otherwise this will lead to the contradiction that $r_{\mathsf{iter}}(a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} - c_{\mathsf{iter}}) \neq (c'_{\mathsf{iter}} - a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}})$ holds. However, since r_{iter} is a random element from $\mathbb{F}$, it implies that this case can occur only with probability at most $\frac{1}{|\mathbb{F}|}$.
- **Case IV:** $c_{\mathsf{iter}} \neq a_{\mathsf{iter}} \cdot b_{\mathsf{iter}}$ **as well as** $c'_{\mathsf{iter}} \neq a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}$ — This case is possible only if $r_{\mathsf{iter}} = (c'_{\mathsf{iter}} - a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}}) \cdot (a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} - c_{\mathsf{iter}})^{-1}$, as otherwise this will lead to the contradiction that $r_{\mathsf{iter}}(a_{\mathsf{iter}} \cdot b_{\mathsf{iter}} - c_{\mathsf{iter}}) \neq (c'_{\mathsf{iter}} - a_{\mathsf{iter}} \cdot b'_{\mathsf{iter}})$ holds. However, since r_{iter} is a random element from $\mathbb{F}$, it implies that this case can occur only with probability at most $\frac{1}{|\mathbb{F}|}$.

Hence, we have shown that except with probability at most $\frac{1}{|\mathbb{F}|}$, the triple $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$ is a multiplication-triple. To complete the proof, we need to argue that the view of Adv in the protocol will be independent of the triple $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$. For this, we first note that $a_{\mathsf{iter}}, b_{\mathsf{iter}}$ and b'_{iter} will be random for the adversary at the time of their generation, which follows from Lemma 8.8. From Lemma 8.5, Adv learns nothing additional about a_{iter}, b_{iter} and b'_{iter} during the two instances of $\Pi_{\mathsf{BasicMult}}$. While Adv learns the value of e_{iter}, since b'_{iter} is a uniformly distributed for Adv, for every candidate value of b'_{iter} from the view-point of Adv, there is a corresponding value of b_{iter} consistent with the e_{iter} learnt by Adv. Hence, learning e_{iter} does not add any new information about $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$ to the view of Adv. Moreover, Adv will be knowing beforehand that d_{iter} will be 0 and hence, learning this value does not change the view of Adv regarding $(a_{\mathsf{iter}}, b_{\mathsf{iter}}, c_{\mathsf{iter}})$.

We next derive the communication complexity of the protocol $\Pi_{\mathsf{RandMultCI}}$.

Lemma 8.13. *Protocol* $\Pi_{\mathsf{RandMultCI}}$ *incurs a communication of* $\mathcal{O}(|\mathcal{Z}_s| \cdot n^5 \cdot \log |\mathbb{F}| + n^6 \cdot \log |\mathbb{F}| \cdot |\sigma|)$ *bits and makes* $\mathcal{O}(n^2)$ *calls to* Π_{BA}.

Proof. To generate $[a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [b'_{\mathsf{iter}}]$ and $[r_{\mathsf{iter}}]$, $\mathcal{O}(n)$ instances of Π_{LSh} and Π_{BA} are invoked. To compute $[c_{\mathsf{iter}}]$ and $[c'_{\mathsf{iter}}]$, two instances of $\Pi_{\mathsf{BasicMult}}$ are invoked. To publicly reconstruct e_{iter} and d_{iter}, two instances of Π_{Rec} are invoked with $|\mathcal{R}| = n$. Finally, if $d_{\mathsf{iter}} \neq 0$, then $3 \cdot |\mathbb{S}_{\mathcal{Z}_s}|$ instances of Π_{RecShare} and $2n$ instances of Π_{Rec} are invoked, with $|\mathcal{R}| = n$. The communication complexity now follows from the communication complexity of Π_{LSh} (Lemma 8.1), communication complexity of $\Pi_{\mathsf{BasicMult}}$ (Lemma 8.7), communication complexity of Π_{RecShare} (Lemma 6.1) and communication complexity of Π_{Rec} (Lemma 6.2).

F.4 Properties of the Protocol Π_{TripGen}

In this section, we prove the properties of the protocol Π_{TripGen} (see Fig 13 for the formal details.) We begin by showing that each party computes an output in the protocol.

Lemma 8.14. *Let t be the size of the largest set in $\mathcal{Z}_s$. Then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties compute an output during Π_{TripGen}, by the time $T_{\mathsf{TripGen}} = (t+1) \cdot T_{\mathsf{RandMultCI}}$ in a synchronous network, or almost-surely, eventually in an asynchronous network, where $T_{\mathsf{RandMultCI}} = T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + T_{\mathsf{BasicMult}} + 4T_{\mathsf{Rec}}$.*

Proof. From Lemma 8.9, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties will know the outcome of each iteration iter, since all honest parties set the Boolean variable $\mathsf{flag}_{\mathsf{iter}}$ to a common value. For every iteration iter where $\mathsf{flag}_{\mathsf{iter}}$ is set to 1, from Lemma 8.11, a *new* corrupt party is added to $\mathcal{GD}$. Thus, after at most t iterations, all the corrupt parties will be included in $\mathcal{GD}$ and the parties will set $\mathsf{flag}_{\mathsf{iter}}$ to 0 in the next iteration. Moreover, they will output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$, computed during the corresponding instance of $\Pi_{\mathsf{RandMultCI}}$.

We next claim that the output computed by the honest parties is indeed a multiplication-triple.

Lemma 8.15. *If the honest parties output $(\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}, [a_{\mathsf{iter}}], [b_{\mathsf{iter}}], [c_{\mathsf{iter}}])$ during the protocol Π_{TripGen}, then $a_{\mathsf{iter}}, b_{\mathsf{iter}}$ and c_{iter} are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Moreover, $c_{\mathsf{iter}} = a_{\mathsf{iter}} b_{\mathsf{iter}}$ holds, except with probability $\frac{1}{|\mathbb{F}|}$. Furthermore, the view of the adversary remains independent of $a_{\mathsf{iter}}, b_{\mathsf{iter}}$ and c_{iter}.*

Proof. Follows from Lemma 8.12.

We finally derive the communication complexity of the protocol Π_{TripGen}.

Lemma 8.16. *Protocol Π_{TripGen} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s| \cdot n^6 \cdot \log|\mathbb{F}| + n^7 \cdot \log|\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^3)$ calls to Π_{BA}.*

Proof. The proof follows from the communication complexity of $\Pi_{\mathsf{RandMultCI}}$ (Lemma 8.13) and the fact that $\mathcal{O}(n)$ instances of $\Pi_{\mathsf{RandMultCI}}$ are invoked in the protocol.

G Properties of the Circuit Evaluation Protocol

In this section, we prove the properties of the protocol Π_{cktEval} (see Fig 14 for the formal description). We begin by showing that the honest parties compute some output during the pre-processing phase.

Lemma G.1. *Protocol Π_{cktEval} achieves the following during the pre-processing phase.*

- **Synchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, at the time T_{Rand}, the honest parties have $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=1,\ldots,L}$, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, where $L \stackrel{def}{=} n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$. Moreover, the view of the adversary remains independent of the values $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=1,\ldots,L}$. At the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}}$, the honest parties have triples $\{(a^{(\ell)}, b^{(\ell)}, c^{(\ell)})\}_{\ell=1,\ldots,c_M}$, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, where $c^{(\ell)} = a^{(\ell)} \cdot b^{(\ell)}$, except with probability $\frac{1}{|\mathbb{F}|}$. The view of the adversary will be independent of the multiplication-triples.*
- **Asynchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, the honest parties eventually have $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=1,\ldots,L}$, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, where $L \stackrel{def}{=} n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$. Moreover, the view of the adversary remains independent of the values $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=1,\ldots,L}$. Furthermore, the honest parties eventually have triples $\{(a^{(\ell)}, b^{(\ell)}, c^{(\ell)})\}_{\ell=1,\ldots,c_M}$, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, where $c^{(\ell)} = a^{(\ell)} \cdot b^{(\ell)}$, except with probability $\frac{1}{|\mathbb{F}|}$. The view of the adversary will be independent of the multiplication-triples.*

Proof. The proof follows from the $\mathcal{Z}_s$-*correctness*, $\mathcal{Z}_a$-*correctness* and *privacy* of the protocol Π_{Rand} (Theorem 7.2) and from the properties of Π_{TripGen} in the asynchronous and asynchronous network (Lemma 8.14 and Lemma 8.15). We also note that the multiplication-triples will be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. This is because there will be at most $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2$ instances of Π_{LSh} invoked as part of Π_{TripGen} for generating c_M multiplication-triples. And prior to invoking the instance of Π_{TripGen}, the honest parties would have already generated $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$ number of linearly secret-shared random pads with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets through the instance of Π_{Rand}, which can serve $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$ instances of Π_{LSh}.

We next show that during the input phase, the inputs of all honest parties will be linearly secret-shared with IC-signatures in a *synchronous* network and in an *asynchronous* network, the inputs of a subset of the parties will be linearly secret-shared with IC-signatures.

Lemma G.2. *Protocol Π_{cktEval} achieves the following during the input phase.*

- **Synchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the honest parties will have a common subset $\mathcal{CS}$ where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$, such that all honest parties will be present in $\mathcal{CS}$. Moreover, corresponding to every $P_j \in \mathcal{CS}$, there will be some value, say $x^{\star(j)}$, which is the same as $x^{(j)}$, which will be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Fur-*

thermore, the view of the adversary will be independent of the $x^{(j)}$ values, corresponding to the honest parties $P_j \in \mathcal{CS}$.

- **Asynchronous Network**: *Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, the honest parties will eventually have a common subset $\mathcal{CS}$ where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$. Moreover, corresponding to every $P_j \in \mathcal{CS}$, there will be some value, say $x^{\star(j)}$, which is the same as $x^{(j)}$, which will be eventually linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. Furthermore, the view of the adversary will be independent of the $x^{(j)}$ values, corresponding to the honest parties $P_j \in \mathcal{CS}$.*

Proof. We first consider a *synchronous* network. From the protocol steps, the honest parties start participating in the input phase, only after computing output during the instance of Π_{TripGen}, which from Lemma G.1 happens at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}}$. We also note that there can be at most $n^3 \cdot c_M + 4n^2 \cdot c_M + n^2$ instances of Π_{LSh}, invoked as part of the instance of Π_{TripGen}, which will utilize the secret-shared values $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=1,\ldots,n^3 \cdot c_M + 4n^2 \cdot c_M + n^2}$ as pads. Consequently, the remaining linearly secret-shared values $\{r^{(\mathfrak{l})}\}_{\mathfrak{l}=n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + 1,\ldots,n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n}$ will be still available to the honest parties for being used as pads in up to n instances of Π_{LSh}, since these pads will be random from the adversary's point of view.

Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. We claim that by the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, all the parties in $\mathcal{H}$ will have a common subset $\mathcal{CS}$, where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$ and where $\mathcal{H} \subseteq \mathcal{CS}$. The proof for this is exactly the same as that of Lemma 8.8. Namely, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}}$, all the parties in $\mathcal{H}$ would start participating with input 1 in the BA instances $\Pi_{\mathsf{BA}}^{(j)}$, corresponding to the parties $P_j \in \mathcal{H}$, since by this time, the instances of Π_{LSh} invoked by the parties in $\mathcal{H}$ will produce output for all the parties in $\mathcal{H}$. Consequently, these BA instances will produce output 1 at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + T_{\mathsf{BA}}$, after which all the parties in $\mathcal{H}$ will start participating with input 0 in the remaining BA instances (if any). Consequently, by the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, all the n instances of Π_{BA} will produce some output and the parties in $\mathcal{H}$ will have a common $\mathcal{CS}$. Next, it can be shown that corresponding to every $P_j \in \mathcal{CS}$, there exists some value, say $x^{\star(j)}$, which is the same as $x^{(j)}$, such that the parties in $\mathcal{H}$ have a linear secret-sharing with IC-signatures of $x^{\star(j)}$, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. The proof for this follows similar lines as that of Lemma 8.8. It is easy to see that $x^{\star(j)}$ will be linearly secret-shared with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. This is because the instance of Π_{LSh} invoked by P_j utilizes $r^{(n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + j)}$ as the pad, which is linearly secret-shared with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets.

The proof of the lemma for an *asynchronous* network is almost the same as above and follows using similar arguments as used to prove Lemma 8.8 for the case of *asynchronous* network.

Finally, the privacy of the inputs $x^{(j)}$ of the parties $P_j \in (\mathcal{H} \cap \mathcal{CS})$ follows from the *privacy* of Π_{LSh} and the fact that the underlying pads $r^{(n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + j)}$ used in the corresponding instances of Π_{LSh} are still random for the adversary, after the instance of Π_{TripGen}.

We next show that the honest parties compute an output during the circuit-evaluation phase.

Lemma G.3. *Protocol Π_{cktEval} achieves the following during the circuit-evaluation phase, where D_M denotes the multiplicative depth of* ckt*.*

- **Synchronous Network***: Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + (D_M + 1) \cdot T_{\mathsf{Rec}}$, the honest parties will have y, where $y = f(x^{\star(1)}, \ldots, x^{\star(n)})$, such that $x^{\star(j)} = x^{(j)}$ for every honest party $P_j \in \mathcal{CS}$ and where $x^{\star(j)} = 0$ for every $P_j \notin \mathcal{CS}$. Moreover, all honest parties will be present in $\mathcal{CS}$. Furthermore, the view of the adversary will be independent of the $x^{(j)}$ values, corresponding to the honest parties $P_j \in \mathcal{CS}$.*
- **Asynchronous Network***: Except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, the honest parties will eventually have y, where $y = f(x^{\star(1)}, \ldots, x^{\star(n)})$, where $x^{\star(j)} = x^{(j)}$ for every honest party $P_j \in \mathcal{CS}$ and where $x^{\star(j)} = 0$ for every $P_j \notin \mathcal{CS}$. Furthermore, the view of the adversary will be independent of the $x^{(j)}$ values, corresponding to the honest parties $P_j \in \mathcal{CS}$.*

Proof. Let us first consider a *synchronous* network. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. From Lemma G.1, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}}$, the parties in $\mathcal{H}$ will have the triples $\{(a^{(\ell)}, b^{(\ell)}, c^{(\ell)})\}_{\ell=1,\ldots,c_M}$, which are linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets and where $c^{(\ell)} = a^{(\ell)} \cdot b^{(\ell)}$, except with probability $\frac{1}{|\mathbb{F}|}$. Moreover, from Lemma G.2, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}}$, the honest parties will have a common subset $\mathcal{CS}$ where $\mathcal{P} \setminus \mathcal{CS} \in \mathcal{Z}_s$, such that all honest parties will be present in $\mathcal{CS}$. Furthermore, corresponding to every $P_j \in \mathcal{CS}$, there will be some value, say $x^{\star(j)}$, which is the same as $x^{(j)}$ for an *honest* P_j, which will be linearly secret-shared with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. At the end of the input phase, the parties take 0 as the input on behalf of the parties $P_j \notin \mathcal{CS}$ and take the default linear secret-sharing of 0 with IC-signatures, with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. To prove the lemma, we show that all the gates in ckt are correctly evaluated. Namely, for every gate in ckt, given the gate-inputs in a linearly secret-shared fashion with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets, the parties compute the gate-output in a linearly secret-shared fashion with IC-signatures with $\mathcal{GW}_1, \ldots, \mathcal{GW}_{|\mathcal{Z}_s|}$ being the underlying core-sets. While this is true for the linear gates in ckt, which follows from the linearity of the secret-sharing, the same is true even for the multiplication gates, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. This is because, for every multiplication gate, the parties deploy a linearly secret-shared multiplication-triple from the pre-processing phase and apply Beaver's method. And the masked gate-inputs are correctly reconstructed through instances of Π_{Rec}, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. Since all the independent multiplication gates at the same multiplicative depth can be evaluated in parallel, to evaluate the multiplication gates, it will take a total $D_M \cdot T_{\mathsf{Rec}}$ time. Finally, once the circuit-output is ready in a secret-shared fashion, it is publicly reconstructed through an instance of Π_{Rec}, which takes T_{Rec} time and produces the correct output, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$. The *privacy* of the inputs of the honest parties in $\mathcal{CS}$ follows from the *privacy* of Π_{LSh} (Lemma 8.1) and the fact no additional information is revealed during the evaluation of multiplication gates. This is because the underlying multiplication-triples which are deployed while applying Beaver's method are random for the adversary.

The proof for the case of *asynchronous* network follows similar arguments as above and depends upon the properties of the pre-processing phase and input phase in the *asynchronous* network (Lemma G.1 and Lemma G.2).

We finally show that the honest parties terminate the protocol.

Lemma G.4. *If the network is synchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, the honest parties terminate the protocol at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + (D_M + 1) \cdot T_{\mathsf{Rec}} + \Delta$. If the network is asynchronous, then except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, the honest parties eventually terminate the protocol.*

Proof. Let us first consider a *synchronous* network. Let $Z^\star \in \mathcal{Z}_s$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. From Lemma G.3, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + (D_M + 1) \cdot T_{\mathsf{Rec}}$, all the parties in $\mathcal{H}$ will have y. Hence every party in $\mathcal{H}$ will send a ready message for y to all the parties, which gets delivered within Δ time, while the parties in $Z^\star$ may send a ready message for some y' where $y' \neq y$. Now since $\mathcal{Z}_s$ *does not* satisfy the $\mathbb{Q}^{(1)}(Z^\star, \mathcal{Z}_s)$ condition, it follows that no party in $\mathcal{H}$ will ever send a ready message for any y' where $y' \neq y$. As $\mathcal{P} \setminus \mathcal{H} \in \mathcal{Z}_s$, it follows that at the time $T_{\mathsf{Rand}} + T_{\mathsf{TripGen}} + T_{\mathsf{LSh}} + 2T_{\mathsf{BA}} + (D_M + 1) \cdot T_{\mathsf{Rec}} + \Delta$, all the parties in $\mathcal{H}$ will have sufficient number of ready messages for y and hence they terminate with output y.

Next, consider an *asynchronous* network. Let $Z^\star \in \mathcal{Z}_a$ be the set of *corrupt* parties and let $\mathcal{H} = \mathcal{P} \setminus Z^\star$ be the set of *honest* parties. Note that $\mathcal{P} \setminus Z^\star \in \mathcal{Z}_s$, since $\mathcal{Z}_a \subset \mathcal{Z}_s$. From Lemma G.3, except with probability $\mathcal{O}(n^3 \cdot \epsilon_{\mathsf{ICP}})$, almost-surely, all the parties in $\mathcal{H}$ will eventually compute y. Hence every party in $\mathcal{H}$ will eventually send some ready message. We claim that no party in $\mathcal{H}$ will ever send a ready message for any $y' \neq y$. On the contrary, let $P_i \in \mathcal{H}$ be the *first* party, which sends a ready message for $y' \neq y$. From the protocol steps, it follows that P_i sends the ready message for y' after computing y' during the circuit-evaluation phase. Otherwise, there should exist a subset of parties $\mathcal{A}$ where $\mathcal{Z}_s$ satisfies $\mathbb{Q}^{(1)}(\mathcal{A}, \mathcal{Z}_s)$ condition (implying that $\mathcal{A}$ has at least one party from $\mathcal{H}$), who should have sent the ready message for y' to P_i, which is *not* possible, since we are assuming P_i to be *first* party from $\mathcal{H}$ to send a ready message for y'. From Lemma G.3, P_i will *not* compute y' and hence will *not* send a ready message for y'. Now since every party in $\mathcal{H}$ eventually computes y in the circuit-evaluation phase, it eventually sends a ready message for y. And since $\mathcal{P} \setminus Z^\star \in \mathcal{Z}_s$ and $\mathcal{P} \setminus Z^\star \notin \mathcal{Z}_s$, it follows that irrespective of the behaviour of the corrupt parties, the parties in $\mathcal{H}$ will eventually receive a sufficient number of ready messages for y, to terminate with output y.

Let P_h be the *first* party from $\mathcal{H}$, who terminates with output y. This implies that there exists a subset of parties $\mathcal{W}$ with $\mathcal{P} \setminus \mathcal{W} \in \mathcal{Z}_s$, who sends a ready message for y to P_h. Now consider the set $(\mathcal{H} \cap \mathcal{W})$. The set satisfies the $\mathbb{Q}^{(1)}(\mathcal{H} \cap \mathcal{W}, \mathcal{Z}_s)$ condition, due to the $\mathbb{Q}^{(2,1)}(\mathcal{P}, \mathcal{Z}_s, \mathcal{Z}_a)$ condition. The ready messages of these parties (for y) get eventually delivered to every party in $\mathcal{H}$. Consequently, every party in $\mathcal{H}$ (including P_h) who has *not* yet sent any ready message will eventually send the ready message for y, which gets eventually delivered to all the parties. And as a result, every party in $\mathcal{H}$ will eventually have a sufficient number of ready messages for y, to terminate with the output y.

We next derive the communication complexity of the protocol.

Lemma G.5. *Protocol Π_{cktEval} incurs a communication of $\mathcal{O}(|\mathcal{Z}_s|^2 \cdot n^{12} \cdot \log |\mathbb{F}| \cdot |\sigma|)$ bits and makes $\mathcal{O}(n^3)$ calls to Π_{BA}.*

Proof. The communication complexity is dominated by the instance of Π_{Rand} to generate $L \stackrel{def}{=} n^3 \cdot c_M + 4n^2 \cdot c_M + n^2 + n$ random secret-shared values and the instance of Π_{TripGen} to generate $L = c_M$ number of secret-shared multiplication-triples. The proof now follows from the communication complexity of Π_{Rand} (Theorem 7.2) and the communication complexity of the (generalized) Π_{TripGen} protocol (Lemma 8.16).

Theorem 9.1 now easily follows from Lemma G.1-G.5.