Perfectly-Secure Synchronous MPC with Asynchronous Fallback Guarantees

Ananya Appan[*] Anirudh Chandramouli[†] Ashish Choudhury[‡]

Abstract

Secure *multi-party computation* (MPC) is a fundamental problem in secure distributed computing. The optimal resilience for *perfectly-secure* MPC in *synchronous* and *asynchronous* networks is $t < n/3$ and $t < n/4$ respectively, where n is the number of parties and t is the number of corruptions. A natural question is whether there exists a protocol tolerating $t_s < n/3$ corruptions in a synchronous network and $t_a < n/4$ corruptions in an *asynchronous* network. We design such a protocol, if $3t_s + t_a < n$. For our protocol, we present a perfectly-secure *Byzantine agreement* (BA) protocol, tolerating $t < n/3$ corruptions in *any* network and a perfectly-secure *verifiable secret-sharing* (VSS) protocol, tolerating t_s and t_a corruptions in a *synchronous* and an *asynchronous* network respectively.

1 Introduction

Consider a set of mutually distrusting parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each P_i has some private input. There exists a *computationally-unbounded* adversary, controlling up to t parties in a *Byzantine* fashion and forcing them to behave arbitrarily during the execution of any protocol. A *perfectly-secure* MPC protocol [14, 32, 25, 11, 34, 2] allows the parties to securely compute any known function of their inputs, such that the *honest* parties (who are not under adversary's control) obtain the correct output and no additional information is revealed about the inputs of the honest parties, beyond what can be revealed by the function output and the inputs of the corrupt parties.

Traditionally, MPC protocols are designed *assuming* either a *synchronous* or *asynchronous* communication model. In *synchronous* MPC (SMPC) protocols, parties are synchronized, with a *publicly-known* upper bound on message delays. In practice, it might be difficult to guarantee such strict time-outs in real-world networks like the Internet. *Asynchronous* MPC (AMPC) protocols operate assuming an *asynchronous* network, where the messages can be arbitrarily delayed. The only guarantee is that every sent message is *eventually* delivered. In any AMPC protocol, a party cannot distinguish between a *slow* sender party (whose messages are arbitrarily delayed) and a *corrupt* sender party (who does not send any messages). Hence, it is *impossible* to ensure that the inputs of all the *honest* parties are considered for computation in an AMPC protocol, as inputs of up to t (potentially honest) parties may have to be ignored, since waiting for all n inputs may turn out to be an endless wait. However, the advantage of AMPC protocols is that the time taken to produce the output depends upon the *actual* speed of the underlying network. Perfectly-secure SMPC is possible

[*]International Institute of Information Technology, Bangalore India. Email: ananya.appan@iiitb.ac.in.

[†]International Institute of Information Technology, Bangalore India. Email: anirudh.c@iiitb.ac.in.

[‡]International Institute of Information Technology, Bangalore India. Email: ashish.choudhury@iiitb.ac.in. This research is an outcome of the R & D work undertaken in the project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

iff there are at most $t_s < n/3$ corruptions [14], while perfectly-secure AMPC is possible iff there are at most $t_a < n/4$ corruptions [13]. We envision a scenario where the parties *are not* aware of the *exact* network type and ask the following question:

Is there a perfectly-secure MPC protocol that is secure under t_s corruptions in a synchronous network, and t_a corruptions in an asynchronous network, where[1] $t_a < t_s$?

No prior work has addressed the above question. We show the existence of an MPC protocol with above guarantees, if $3t_s + t_a < n$. We often call our protocol (and the underlying building blocks) a *best-of-both-worlds* protocol, since it offers the best security properties, both in the synchronous and the asynchronous communication model.

1.1 Technical Overview

We assume that the function to be securely computed is represented by some arithmetic circuit cir over a finite field $\mathbb{F}$, consisting of linear and non-linear (multiplication) gates. The goal is to securely "evaluate" cir in a secret-shared fashion, such that all the values during the circuit-evaluation are t-shared[2] as per Shamir's secret-sharing scheme [45]. Intuitively, this guarantees that an adversary controlling up to t parties *does not* learn any additional information during circuit-evaluation. The *degree-of-sharing* t is set to $t < n/3$ and $t < n/4$ in SMPC and AMPC protocols respectively. Since, in our protocol, the parties will *not* be aware of the *exact* network type, we need to ensure that *irrespective* of the network type, all the values during circuit-evaluation are secret-shared with the degree-of-sharing being $t = t_s$.

For shared circuit-evaluation, we follow Beaver's paradigm [9], where multiplication gates are evaluated using random t_s-shared *multiplication-triples* of the form (a, b, c), where $c = a \cdot b$ (due to the linearity of Shamir's secret-sharing, linear gates can be evaluated *non-interactively*). The shared multiplication-triples are generated using the framework of [23], which shows how to use any polynomial-based *verifiable secret-sharing* (VSS) [22] and a *Byzantine agreement* (BA) protocol [37, 7] to generate shared random multiplication-triples, in a synchronous or an asynchronous network. However, there are several challenges to adapt the framework if the parties are *unaware* of the exact network type.

First Challenge — A Best-of-Both-Worlds VSS Protocol: Informally, in a polynomial-based VSS protocol, there exists a *dealer* D with a t-degree polynomial. The protocol allows D to distribute points on this polynomial to the parties in a "verifiable" fashion[3], such that the view of the adversary remains independent of D's polynomial for an *honest* D. In a *synchronous* VSS (SVSS) protocol, every party has the correct point after some known time-out (*correctness* property). *Verifiability* guarantees that even a *corrupt* D distributes points on some t-degree polynomial (*strong-commitment* property). Perfectly-secure SVSS is possible iff $t < n/3$ [27]. For an *asynchronous* VSS (AVSS) protocol, the *correctness* property guarantees that for an *honest* D, the honest parties *eventually* receive points on D's polynomial. However, a *corrupt* D may not invoke the protocol in the first place and the parties *cannot* distinguish this scenario from the case when D's messages are delayed. This is unlike the *strong-commitment* of SVSS where, if the parties do not obtain output after a known time-out, then

[1]If $t_s = t_a$, then the necessary condition of AMPC implies $t_s < n/4$ and the question is trivial. This is because one can use any existing perfectly-secure AMPC protocol (with appropriate time-outs), which will work even in the synchronous network, considering the inputs of *all honest* parties.

[2]A value s is t-shared, if there is some t-degree polynomial $f_s(\cdot)$ with $f_s(0) = s$ and every (honest) P_i has a distinct point on $f_s(\cdot)$.

[3]Hence the protocol allows D to generate a t-sharing of the constant term of the polynomial, which is also called as D's *secret*.

the parties *publicly* conclude that D is corrupt. Hence, the *strong-commitment* of AVSS guarantees that if D is *corrupt and* if some honest party obtains a point on D's polynomial, then all honest parties eventually obtain their respective points on this polynomial. Perfectly-secure AVSS is possible iff $t < n/4$ [13, 4].

Existing SVSS protocols [31, 30, 36, 21] become completely insecure in an *asynchronous* network, even if a single expected message from an *honest* party is delayed. On the other hand, existing AVSS protocols [13, 10, 41, 21] only work when D's polynomial has degree $t < n/4$ and become insecure if there are more than $n/4$ corruptions (which can happen in a *synchronous* network). The *first* challenge to adapt the framework of [23] in our setting is to get a perfectly-secure VSS protocol, which provides security against t_s and t_a corruptions in a synchronous and an asynchronous network respectively *and* where D's polynomial is a t_s-degree polynomial, *irrespective* of the network type. We are not aware of any VSS protocol with these guarantees. We present a perfectly-secure VSS protocol satisfying the above properties provided $3t_s + t_a < n$ holds. Our VSS protocol satisfies the *correctness* requirement of SVSS and AVSS in a *synchronous* and an *asynchronous* network respectively. However, it *only* satisfies the *strong-commitment* requirement of AVSS, *even* if the network is *synchronous*. This is because a potentially *corrupt* D *may not* invoke the protocol and the parties will *not* be aware of the exact network type. Since our VSS protocol is slightly technical, we defer the details to Section 4.

Second Challenge — A Best-of-Both-Worlds BA Protocol: A BA protocol [43] allows the parties with private input bits to reach agreement on a common output bit (*consistency*), which is the input-bit of the *honest* parties if all of them have the same input (*validity*). *Perfectly-secure* BA protocols can be designed tolerating $t < n/3$ corruptions, *irrespective* of the network type. However, the *termination* (also called *liveness*) guarantees are *different* for *synchronous* BA (SBA) and *asynchronous* BA (ABA). (Deterministic) SBA protocols ensure that all honest parties obtain their output after some fixed time (*guaranteed liveness*). On the other hand, to circumvent the FLP impossibility result [29], ABA protocols are randomized and have two different liveness guarantees: if all honest parties have the same input, then it is *guaranteed* that the honest parties eventually obtain their output; otherwise the honest parties eventually obtain their output *asymptotically* with probability 1 (*almost-surely liveness*) [3, 8]. SBA protocols become insecure when executed in an *asynchronous* network, while ABA protocols can provide only almost-surely liveness in a *synchronous* network.

The *second* challenge to adapt the framework of [23] is to get a *perfectly-secure* BA protocol, which provides security *both* in a synchronous and an asynchronous network. We are *not* aware of any such BA protocol. We present a perfectly-secure BA protocol tolerating $t < n/3$ corruptions, which provides guaranteed liveness, validity and consistency in a *synchronous* network, while in an *asynchronous* network, it provides validity, consistency and almost-surely liveness.

1.2 Related Work

Best-of-both-worlds protocols have been studied very recently. [16] and [17, 26] show that the condition $2t_s + t_a < n$ is necessary and sufficient for best-of-both-worlds *cryptographically-secure* BA and MPC respectively, tolerating *computationally bounded* adversaries. Another line of work considers asynchronous networks with some form of *partial synchrony* and builds protocols for various secure distributed computing tasks [35, 38, 5, 42].

A common principle used in [16, 17, 26] to design best-of-both-worlds protocols for a specific task T (BA/MPC) is the following: the parties first run a *synchronous* protocol for T with threshold t_s, which also provides certain security guarantees in an asynchronous environment tolerating t_a corruptions. After the known "time-out" of the synchronous protocol, the parties run an *asynchronous* protocol for T with threshold t_a, which also provides certain security guarantees in the presence of t_s corruptions. The input for the asynchronous protocol is decided based on the output the parties receive after the time-out of the synchronous protocol. The overall output is then decided based on the output parties receive from the asynchronous protocol. For MPC this means that the parties need to evaluate the circuit *twice*. We also follow a similar design principle for our BA protocol. However, for MPC, we *do not* require the parties to run two protocols and evaluate the circuit twice. Rather the parties need to evaluate the circuit *only once*.

2 Preliminaries and Definitions

The parties are assumed to be connected by pair-wise secure channels. The underlying network can be synchronous or asynchronous, with parties being *unaware* about the exact type. In a *synchronous* network, every sent message is delivered in the same order, within some known time Δ. In an *asynchronous* network, messages are sent with arbitrary, but finite, delay, and *need not* be delivered in the same order. The only guarantee is that every sent message is *eventually* delivered. A *computationally-unbounded* adversary can control up to t_s and t_a parties in a *synchronous* and an *asynchronous* network respectively, where $t_a < t_s$ and $3t_s + t_a < n$ (hence $t_a < n/4$). All computations in our protocols are done over a finite field $\mathbb{F}$, where $|\mathbb{F}| > 2n$ and $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ are publicly-known, distinct, non-zero elements from $\mathbb{F}$. For simplicity, we assume that each P_i has input $x^{(i)} \in \mathbb{F}$, and parties want to securely compute a function $f : \mathbb{F}^n \rightarrow \mathbb{F}$. Without loss of generality, f is represented by an arithmetic circuit cir over $\mathbb{F}$, consisting of linear and non-linear (multiplication) gates [33], where cir has c_M multiplication gates and a multiplicative depth of D_M.

Termination Guarantees of Our Sub-Protocols: For simplicity, we will *not* be specifying any termination criteria for our sub-protocols and the parties will keep on participating in these sub-protocol instances, even after receiving their outputs. The termination criteria of our MPC protocol will ensure that once a party terminates the MPC protocol, it terminates all underlying sub-protocol instances. We will use existing *randomized* ABA protocols which ensure that the honest parties (eventually) obtain their respective output *almost-surely.* This means that the probability that a correct party obtains its output after participating for *infinitely* many rounds approaches 1 *asymptotically* [3, 40, 8]. That is:

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

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

Polynomials Over a Field: A d-degree *univariate polynomial* over $\mathbb{F}$ is of the form $f(x) = a_0 + \ldots + a_d x^d$, where each $a_i \in \mathbb{F}$. An (ℓ, ℓ)-degree *symmetric bivariate polynomial* is of the form $F(x, y) = \sum_{i,j=0}^{i=\ell,j=\ell} r_{ij} x^i y^j$, where $r_{ij} \in \mathbb{F}$ and $r_{ij} = r_{ji}$ holds for all i, j. Thus $F(\alpha_j, \alpha_i) = F(\alpha_i, \alpha_j)$ for all α_i, α_j and $F(x, \alpha_i) = F(\alpha_i, y)$. We say that a ℓ-degree $F_i(x)$, where $i \in \{1, \ldots, n\}$, *lies* on a (ℓ, ℓ)-degree symmetric bivariate polynomial $F(x, y)$, if $F(x, \alpha_i) = F_i(x)$ holds.

Lemma 2.1 **([24, 6]).** *The following results related to bivariate polynomials are standard.*

- *Let $f_{i_1}(x), \ldots, f_{i_q}(x)$ be ℓ-degree polynomials, where $q \geq \ell + 1$ and $i_1, \ldots, i_q \in \{1, \ldots, n\}$, such that $f_i(\alpha_j) = f_j(\alpha_i)$ $\forall i, j \in \{i_1, \ldots, i_q\}$. Then $f_{i_1}(x), \ldots, f_{i_q}(x)$ lie on a unique (ℓ, ℓ)-degree symmetric bivariate polynomial $F^\star(x, y)$.*

– *Let $\mathcal{C} \subset \mathcal{P}$ and $q_1(\cdot) \neq q_2(\cdot)$ be d-degree polynomials where $d \geq |\mathcal{C}|$ such that $q_1(\alpha_i) = q_2(\alpha_i)$ for all $P_i \in \mathcal{C}$. Then the probability distributions $\left\{\{F(x, \alpha_i)\}_{P_i \in \mathcal{C}}\right\}$ and $\left\{\{F'(x, \alpha_i)\}_{P_i \in \mathcal{C}}\right\}$ are identical, where $F(x, y)$ and $F'(x, y)$ are random (d, d)-degree symmetric bivariate polynomials, such that $F(0, y) = q_1(\cdot)$ and $F'(0, y) = q_2(\cdot)$ holds.*

Definition 2.2 (**d-sharing).** A value $s \in \mathbb{F}$ is d-shared, if there exists a d-degree *sharing-polynomial* $f_s(\cdot)$, with $f_s(0) = s$, such that every (honest) P_i has the *share* $s_i = f_s(\alpha_i)$. The vector of shares of s corresponding to the (honest) parties P_i is called a d-sharing of s, denoted by $[s]_d$. We will omit the degree d from the notation $[\cdot]_d$ if it is clear from the context.

d-sharing satisfies the *linearity* property; i.e. given $[a]_d$ and $[b]_d$, then $[c_1 \cdot a + c_2 \cdot b]_d = c_1 \cdot [a]_d + c_2 \cdot [b]_d$, where $c_1, c_2 \in \mathbb{F}$ are *publicly-known*. In general, consider any arbitrary linear function $g : \mathbb{F}^\ell \to \mathbb{F}^m$ and let $u^{(1)}, \ldots, u^{(\ell)}$ be d-shared. When we say that *parties locally compute* $([v^{(1)}]_d, \ldots, [v^{(m)}]_d) = g([u^{(1)}]_d, \ldots, [u^{(\ell)}]_d)$, we mean that parties *locally* apply the function g on their respective shares of $u^{(1)}, \ldots, u^{(\ell)}$ to get their respective shares of $v^{(1)}, \ldots, v^{(m)}$.

2.1 Existing Primitives

Online Error-Correction (OEC) [13] Let $\mathcal{P}' \subseteq \mathcal{P}$ have at most t corrupt parties and let there exist some d-degree polynomial $q(\cdot)$ with every (honest) $P_i \in \mathcal{P}'$ having a point $q(\alpha_i)$. The goal is to make some *designated* party P_R reconstruct $q(\cdot)$. For this, each $P_i \in \mathcal{P}'$ sends $q(\alpha_i)$ to P_R, who keeps waiting till it receives $d + t + 1$ points, all of which lie on a *unique* d-degree polynomial. This step requires P_R to repeatedly apply the Reed-Solomon (RS) error-correction procedure [39] and try to recover $q(\cdot)$, upon receiving a new point from the parties in $\mathcal{P}'$. Once P_R receives $d + 1 + 1$ points lying on a d-degree polynomial, say $q'(\cdot)$, then $q'(\cdot) = q(\cdot)$. This is because among these $d + t + 1$ points, at least $d + 1$ are from *honest* parties in $\mathcal{P}'$, which uniquely determine $q(\cdot)$. If $d < (|\mathcal{P}'| - 2t)$, then in an *asynchronous* network, P_R *eventually* receives $d + t + 1$ points (from the *honest* parties in $\mathcal{P}'$) lying on $q(\cdot)$ and recovers $q(\cdot)$. Moreover, in a *synchronous* network, it will take at most Δ time for P_R to recover $q(\cdot)$, since the points of the honest parties will be delivered within Δ time. The procedure $\mathsf{OEC}(d, t, \mathcal{P}')$ and its properties are presented in Appendix A.

$(n, t)-$star [13] Let G be an undirected graph over $\mathcal{P}$. Then a pair $(\mathcal{E}, \mathcal{F})$ where $\mathcal{E} \subseteq \mathcal{F} \subseteq \mathcal{P}$ is called an $(n, t)-$star, if $|\mathcal{E}| \geq n - 2t$, $|\mathcal{F}| \geq n - t$ and there exists an edge between every $P_i \in \mathcal{E}$ and every $P_j \in \mathcal{F}$. [13] presents an efficient algorithm $\mathsf{AlgStar}$ which outputs an $(n, t)-$star $(\mathcal{E}, \mathcal{F})$, if G contains a clique of size at least $n - t$.

Asynchronous Reliable Broadcast (Acast) We use the Bracha's Acast protocol [18], which allows a designated *sender* $\mathsf{S} \in \mathcal{P}$ to send some message $m \in \{0, 1\}^\ell$ identically to all the parties in the presence of any $t < n/3$ corruptions. While the protocol has been designed for an *asynchronous* network, it also provides certain guarantees in a *synchronous* network, as stated in Lemma 2.3. The Acast protocol and proof of Lemma 2.3 are available in Appendix A.

Lemma 2.3. *Bracha's Acast protocol Π_{ACast} achieves the following in the presence of up to $t < n/3$ corruptions.*

– *Asynchronous Network:* **(a) Liveness***: If S is honest, then all honest parties eventually obtain some output.* **(b) Validity***: If S is honest, then every honest party with an output, outputs m.* **(c) Consistency***: If S is corrupt and some honest party outputs $m^\star$, then every honest party eventually outputs $m^\star$.*

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

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

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

We begin with the definition of BA, which is a modified version of [16], as we *do not* require any *termination* guarantees.

Definition 3.1 (**BA** [16]). Let Π be a protocol for $\mathcal{P}$, where every P_i has input $b_i \in \{0, 1\}$ and a possible output from $\{0, 1, \perp\}$. Π achieves the following, tolerating up to t corruptions, if the corresponding conditions are satisfied.
– **Guaranteed Liveness**: All honest parties obtain an output.
– **Almost-Surely Liveness**: Almost-surely, all honest parties obtain some output.
– **Validity**: If all honest parties have input b, then every honest party with an output, outputs b.
– **WeakValidity**: Same as **Validity**, except that output of each honest party could be b or $\perp$.
– **Consistency**: All honest parties with an output, output the same value (which can be $\perp$).
– **WeakConsistency**: All honest parties with an output, output either a common $v \in \{0, 1\}$ or $\perp$.

Π is called a *t-perfectly-secure synchronous-BA* (SBA) protocol if in a *synchronous* network, it has *guaranteed liveness*, *validity* and *consistency.* On the other hand, Π is called a *t-perfectly-secure asynchronous-BA* (ABA) protocol, if in an *asynchronous network* it has *almost-surely liveness*, *validity* and *consistency.*

Insecurity of Existing ABA protocols in Synchronous Networks: Existing ABA protocols achieve the following.

Lemma 3.2 ([3, 8]). *Let* $t < n/3$*. Then there exists a randomized protocol* Π_{ABA} *such that:*
– *Asynchronous Network: The protocol is a t-perfectly-secure ABA protocol. If the inputs of all honest parties are same, then* Π_{ABA} *achieves guaranteed liveness, else it achieves almost-surely liveness.*
– *Synchronous Network: The protocol achieves validity, consistency and the following liveness guarantees.*
  – *If the inputs of all honest parties are same, then* Π_{ABA} *achieves guaranteed liveness and all honest parties obtain their output within time* $T_{\mathsf{ABA}} = k \cdot \Delta$ *for some constant* k*. Else it achieves almost-surely liveness and requires* $\mathcal{O}(poly(n) \cdot \Delta)$ *expected time to generate the output.*
– *Communication Complexity:* Π_{ABA} *incurs a communication of* $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ *bits if the inputs of all honest parties are the same. Else, it incurs an expected communication of* $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ *bits.*

Π_{ABA} is designed using a *weaker* "variant" of AVSS called *shunning* AVSS (SAVSS) [3, 8], which *cannot* be used for circuit-evaluation. We provide a brief overview of the ABA protocols of [3, 8] and a proof of Lemma 3.2 in Appendix B. Π_{ABA} *cannot* be considered as a *best-of-the-both-worlds* BA protocol, as it achieves *guaranteed liveness* in a *synchronous* network only when *all* honest parties have the same input. We design a *perfectly-secure* BA protocol, which is secure in *any* network. For this, we adapt a blueprint of [16] for best-of-both-worlds *cryptographically-secure* BA protocols.

A Blueprint for BA: Let Π_{SBA} be a *t-perfectly-secure* SBA protocol with *guaranteed liveness* and *weak validity* in an asynchronous network, such that all honest parties have an output at some local time T_{SBA}. We stress that *existing* SBA protocols *may not* provide the above *asynchronous* properties and later we will design Π_{SBA}, where $T_{\mathsf{SBA}} = (12n - 3) \cdot \Delta$. Given Π_{SBA}, we "combine" it with Π_{ABA} to get a best-of-both-worlds BA protocol Π_{BA} (Fig 1). In Π_{BA}, the parties first run Π_{SBA} and check for a "legitimate" output at time T_{SBA} and either "modify" their input to that output, or stick to their original inputs. The parties then run Π_{ABA} to decide the final output. In protocol Π_{BA} (and all other protocols in this paper), we specify by red color the *maximum* time required by each step in a *synchronous* network.

Protocol Π_{BA}

- **(Time T_{SBA})**: On having the input b_i, participate in an instance of Π_{SBA} and wait for time T_{SBA}. Let v_i be the output from Π_{SBA} after time T_{SBA}. If $v_i \neq \perp$, then set $v_i^\star = v_i$. Else set $v_i^\star = b_i$.
- **(Time T_{ABA})**: Participate in an instance of Π_{ABA} with input $v_i^\star$. Output the result of Π_{ABA}.

Figure 1: The best-of-both-worlds BA protocol that combines Π_{SBA} and Π_{ABA}. The above code is executed by P_i.

Theorem 3.3. *Let $t < n/3$. Let Π_{SBA} be a t-perfectly-secure SBA protocol, with guaranteed liveness and weak validity in asynchronous network, such that all honest parties obtain an output at local time T_{SBA}. Moreover, let Π_{ABA} be a randomized protocol satisfying the conditions as per Lemma 3.2. Then Π_{BA} achieves the following.*

- *The protocol is a t-perfectly-secure SBA protocol and at time $T_{\mathsf{BA}} = T_{\mathsf{SBA}} + T_{\mathsf{ABA}}$, all honest parties have an output. The protocol incurs a communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits.*
- *The protocol is a t-perfectly-secure ABA protocol with an expected communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits.*

Theorem 3.3 is proved in Appendix C. Protocol Π_{BA} is invoked $\mathcal{O}(n^3)$ times in our MPC protocol, which is *independent* of cir; hence we do not focus on the *exact* communication complexity of Π_{BA}. We next proceed to design Π_{SBA}.

3.1 Protocol Π_{SBA}: Synchronous BA Protocol with Asynchronous Guarantees

[16] presents a blueprint for Π_{SBA} with *cryptographic security* (we adapt it for *perfect* security) using a special type of *broadcast* protocol. We next review the definition of broadcast, where we *do not* put any *termination* requirement.

Definition 3.4 (**Broadcast** [16]). Let Π be a protocol, where a sender $\mathsf{S} \in \mathcal{P}$ has input $m \in \{0, 1\}^\ell$, and parties obtain a possible output. Π achieves the following, tolerating up to t corruptions, if the corresponding conditions are satisfied.

- **Liveness**: Irrespective of S, all honest parties obtain some output (which can be $\perp$).
- **Validity**: If S is *honest*, then every honest party with an output, outputs m.
- **Weak Validity**: Same as **Validity**, except that every honest party outputs either m or $\perp$.

– **Consistency**: If S is *corrupt*, then every honest party with an output, outputs a common value (including $\bot$).
– **Weak Consistency**: If S is *corrupt*, then every honest party with an output, outputs a common $m^\star \in \{0,1\}^\ell$ or $\bot$.

Π is called a *t-perfectly-secure broadcast* protocol, if it has *Liveness*, *Validity* and *Consistency.*

A Blueprint for Π_{SBA} Let Π_{BC} be a *t-perfectly-secure broadcast* protocol in the *synchronous* network, with weak validity, weak consistency and liveness in the *asynchronous* network, where all honest parties have an output at some local time T_{BC} (later we will design Π_{BC} with $T_{\mathsf{BC}} = (12n-3)\cdot\Delta$). Given Π_{BC}, we design a protocol Π_{SBA} (Fig 2) where all honest parties have an output at time T_{BC}. In the protocol, every party broadcasts its input bit through an instance of Π_{BC}. At time T_{BC} the parties check if $n-t$ instances have produced outputs different from $\bot$ (which should happen in a *synchronous* network) and output the majority of those values. Theorem 3.5 is proved in Appendix C.

Protocol Π_{SBA}

– **(Time T_{BC})**: On having input $b_i \in \{0,1\}$, broadcast b_i by acting as a sender S and invoking an instance $\Pi_{\mathsf{BC}}^{(i)}$ of Π_{BC}. For $j = 1,\ldots,n$, participate in the instance $\Pi_{\mathsf{BC}}^{(j)}$ invoked by P_j as a sender.
– **(Local Computation)**: For $j = 1,\ldots,n$, let $b_i^{(j)} \in \{0,1,\bot\}$ be the output obtained in $\Pi_{\mathsf{BC}}^{(j)}$ at the end of time T_{BC}.
  – If there are at least $n-t$ $b_i^{(j)}$ values different from $\bot$, then output the majority of these values, else output $\bot$.

Figure 2: SBA protocol using protocol Π_{BC}. The above code is executed by every $P_i \in \mathcal{P}$.

Theorem 3.5. *Let $t < n/3$. Let Π_{BC} be a t-perfectly-secure broadcast protocol in the synchronous network, with weak validity, weak consistency and liveness in an asynchronous network where all honest parties have an output at some local time T_{BC}. Then Π_{SBA} achieves the following, with its communication complexity being n times that of Π_{BC}.*
– *Π_{SBA} is a t-perfectly-secure SBA protocol where honest parties have an output ($\neq \bot$) at time $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$.*
– *The protocol has guaranteed liveness and weak validity in the asynchronous network, such that all honest parties have an output at (local) time $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$.*

3.2 Protocol Π_{BC}: Synchronous Broadcast with Asynchronous Guarantees

From Theorems 3.3 and 3.5, the design of Π_{BA} boils down to Π_{BC}. We first note that Bracha's Acast protocol Π_{ACast} *does not* satisfy the requirements of Π_{BC}, since for a *corrupt* S there is *no* liveness guarantee, and *all* honest parties *may not* obtain an output within the *same* time even in a *synchronous* network (see Lemma 2.3). Interestingly, our instantiation of Π_{BC} is based on Π_{ACast} by "stitching" it with *any existing* t-secure SBA protocol with *guaranteed liveness* in an *asynchronous* network. For communication efficiency, we choose the recursive phase-king based *t-perfectly-secure* SBA protocol Π_{BGP} of [15], which requires a communication of $\mathcal{O}(n^2\ell)$ bits, if the inputs of the parties are of size ℓ bits. In Π_{BGP}, at time $T_{\mathsf{BGP}} = (12n-6)\cdot\Delta$, all honest parties have an output if the network is *synchronous* (see Lemma 10.7 of [1]). To ensure guaranteed liveness in an *asynchronous* network, the parties can simply check if the output-decision criteria of Π_{BGP} is satisfied at local time $(12n-6)\cdot\Delta$ (if the criteria is *not* satisfied, then parties can output $\bot$). We stress that we *only* need guaranteed liveness from Π_{BGP} in an *asynchronous* network; the protocol *need not* provide *weak* *validity* in an *asynchronous* network and hence *cannot* be used as an instantiation of Π_{SBA} in the protocol Π_{BA}.

In protocol Π_{BC}, S first Acasts its message. In a *synchronous* network, at time 3Δ, every honest party should have S's message (for an *honest* S). So the parties start participating in an instance of Π_{BGP} with their respective inputs being the output obtained from S's Acast at time 3Δ; if there is no output at time 3Δ, then the input for Π_{BGP} is $\perp$. Finally, at time $3\Delta + T_{\mathsf{BGP}}$, parties output $m^\star$, if it has been received from Acast of S *and* if it is the output of Π_{BGP} as well, else the parties output $\perp$. Now *consistency* is achieved for a *corrupt* S in a *synchronous* network because if any *honest* party obtains an output $m^\star \neq \perp$, then *at least* one *honest* party must have received $m^\star$ from S's Acast by time 3Δ and so by time $3\Delta + T_{\mathsf{BGP}}$, *all* honest parties will receive $m^\star$ from S's Acast.

Eventual Consistency and Validity in Asynchronous Network: In Π_{BC}, the parties set a "time-out" of $3\Delta + T_{\mathsf{BGP}}$, due to which it provides *weak validity* and *weak consistency* in an *asynchronous* network. This is because some *honest* parties may receive S's message from the Acast of S within the timeout, while others may fail to do so. The time-out is required, as we need *liveness* from Π_{BC} in *both* synchronous and asynchronous network, when used in Π_{SBA}.

Looking ahead, we will use Π_{BC} in our VSS protocol. Due to *weak validity* and *weak consistency* properties, one subset of *honest* parties may output a common value different from $\perp$ at the end of time-out, while others may output $\perp$. For the security of the VSS protocol, we would require the latter subset of (honest) parties to *eventually* output the common non-$\perp$ value, if the parties *continue* participating in Π_{BC}. For this, each P_i who outputs $\perp$ at time $3\Delta + T_{\mathsf{BGP}}$, "switches" its output to $m^\star$, if P_i *eventually* receives $m^\star$ from S's Acast. To differentiate between the two ways of obtaining output, we use the terms *regular-mode* and *fallback-mode*. Regular-mode consists of the process of deciding the output at time $3\Delta + T_{\mathsf{BGP}}$, while fallback-mode consists of the process of deciding the output beyond time $3\Delta + T_{\mathsf{BGP}}$.

If the network is *asynchronous* and S is *honest*, then from the *liveness* and *validity* of Π_{ACast}, every honest party *eventually* obtains m from S's Acast. Moreover, even if S is *corrupt*, the fallback-mode *will not* lead to different honest parties obtaining different non-$\perp$ outputs due to the *consistency* property of Π_{ACast}.

Protocol Π_{BC}

(Regular Mode)

- **(Time $3\Delta + T_{\mathsf{BGP}} = (12n-3)\cdot\Delta$)**:
  - On having the input $m \in \{0,1\}^\ell$, sender S Acasts m.
  - At time 3Δ, each $P_i \in \mathcal{P}$ participates in an instance of Π_{BGP}, where the input of P_i is $m^\star$ if $m^\star \in \{0,1\}^\ell$ is received from the Acast of S, else the input is $\perp$ (encoded as a default ℓ-bit string).
- **(Local Computation)**: At time $3\Delta + T_{\mathsf{BGP}}$ output $m^\star \neq \perp$, if $m^\star$ is received from the Acast of S *and* $m^\star$ is computed as output during the instance of Π_{BGP}. Else, output $\perp$. Keep participating in the protocol even after computing the output.

(Fallback Mode)

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

Figure 3: Synchronous broadcast with asynchronous guarantees.

For the proof of Theorem 3.6, see Appendix C (in the theorem, ℓ denotes the number of bits in S's message).

Theorem 3.6. *Let Π_{BGP} be a t-perfectly-secure SBA protocol with guaranteed liveness in an asynchronous network and a communication complexity of $\mathcal{O}(n^2\ell)$ bits, where all honest parties have an* *output at time $T_{\mathsf{BGP}} = (12n-6)\cdot\Delta$. Then Π_{BC} achieves the following for any $t < n/3$, with a communication complexity of $\mathcal{O}(n^2\ell)$ bits where $T_{\mathsf{BC}} = (12n-3)\cdot\Delta$.*

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

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

4 Best-of-Both-Worlds Perfectly-Secure VSS

In the VSS protocol Π_{VSS}, there exists a *dealer* D with L number of t_s-degree polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}$ where $L \geq 1$ and each (honest) P_i is supposed to "verifiably" receive output *shares* $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$; hence the goal is to generate a t_s-sharing of $\{q^{(\ell)}(0)\}_{\ell=1,\ldots,L}$. If D is *honest*, then in an *asynchronous* network, each (honest) P_i *eventually* gets its shares, while in a *synchronous* network, P_i gets its shares after some *fixed* time, such that the view of the adversary remains independent of D's polynomials. *Verifiability* ensures that if D is *corrupt*, then either no honest party obtains any output (if D does not invoke the protocol), or there exist L number of t_s-degree polynomials, such that each honest P_i gets its shares lying on these polynomials. Note that in the latter case, we *cannot* bound the time within which honest parties will have their shares, even if the network is *synchronous*, as a *corrupt* D may delay sending the messages arbitrarily and the parties *will not* know the exact network type. To design Π_{VSS}, we first design a "weaker" primitive called *weak polynomial-sharing* (WPS), whose security guarantees are *identical* to that of VSS for an *honest* D. However, for a *corrupt* D, the security guarantees are "weakened", as only a subset of honest parties may get the desired shares.

4.1 The Best-of-Both-Worlds Weak Polynomial-Sharing (WPS) Protocol

For simplicity, we explain our WPS protocol Π_{WPS} assuming D has a *single* t_s-degree polynomial $q(\cdot)$ as input; later we discuss the modifications needed to handle L polynomials efficiently. Protocol Π_{WPS} is obtained by "stitching" a *synchronous* WPS protocol with an *asynchronous* WPS protocol. We first explain these two individual protocols, followed by the procedure to stitch them together, where the parties will *not be* knowing the exact network type.

WPS in Asynchronous Network: In an *asynchronous* network, one can consider the following protocol Π_{AWPS}: D embeds $q(\cdot)$ in a random (t_s, t_s)-degree symmetric bivariate polynomial $Q(x, y)$ at $y = 0$ and distributes univariate polynomials lying on $Q(x, y)$ to respective parties. To verify whether D has distributed "consistent" polynomials, the parties check for the pair-wise consistency of their supposedly common points and make public the results through `OK` messages, if the tests are "positive". Based on the `OK` messages, the parties prepare a *consistency graph* and look for an $(n, t_a)-$star, say $(\mathcal{E}', \mathcal{F}')$. If D is *honest*, then $(\mathcal{E}', \mathcal{F}')$ will be obtained eventually, since the honest parties form a clique of size at least $n - t_a$. The existence of $(\mathcal{E}', \mathcal{F}')$ guarantees that the polynomials of the *honest* parties in $\mathcal{F}'$ lie on a single (t_s, t_s)-degree symmetric bivariate polynomial $Q^\star(x, y)$, where $Q^\star(x, y) = Q(x, y)$ for an *honest* D. This is because $\mathcal{E}'$ has at least $t_s + 1$ *honest* parties with pair-wise consistent polynomials, defining $Q^\star(x, y)$, and the polynomial of every *honest* party in $\mathcal{F}'$ is pair-wise consistent with the polynomials of every *honest* party in $\mathcal{E}'$. The parties *outside* $\mathcal{F}'$ obtain their polynomials lying on $Q^\star(x, y)$ by applying OEC on the common points on these polynomials received from the parties in $\mathcal{F}'$. Every P_i then outputs $Q^\star(0, \alpha_i)$ as its share, which is same as $q^\star(\alpha_i)$, where $q^\star(\cdot) = Q^\star(0, y)$. Note that Π_{AWPS} constitutes a VSS in the *asynchronous* network, as for an *honest* D every honest party eventually gets its share. For a *corrupt* D, every honest party eventually gets its share, if some honest party gets its share.

WPS for Synchronous Network: Π_{AWPS} fails in a *synchronous* network with up to t_s corruptions, as only $n - t_s$ honest parties are guaranteed and hence the parties may *fail* to find an $(n, t_a)-$star. The existence of an $(n, t_s)-$star, say $(\mathcal{E}, \mathcal{F})$, in the consistency graph is not "sufficient" to prove that D has distributed consistent polynomials, as for a *corrupt* D, the polynomials of the *honest* parties in $\mathcal{E}$ *need not* lie on a single (t_s, t_s)-degree symmetric bivariate polynomial. This is because now $\mathcal{E}$ is guaranteed to have *only* $n - 2t_s - t_s > t_a$ *honest* parties with pair-wise consistent polynomials, while the polynomials of all the *honest* parties in $\mathcal{F}$ *need not* be pair-wise consistent. Instead, the parties now look for a "special" $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$, where the polynomials of all *honest* parties in $\mathcal{F}$ are guaranteed to lie on a single (t_s, t_s)-degree symmetric bivariate polynomial. Such a special $(\mathcal{E}, \mathcal{F})$ is bound to exist for an *honest* D. Based on the above idea, protocol Π_{SWPS} for a *synchronous* network proceeds as follows. For ease of understanding, we explain the protocol as a sequence of communication *phases*, with the parties being *synchronized* in each phase.

In the *first* phase, D distributes the univariate polynomials, during the *second* phase the parties perform pair-wise consistency tests and during the *third* phase the parties make public the results of *positive* tests. Additionally, the parties *also* make public the results of "negative" tests through `NOK` messages and their respective versions of the disputed points (the `NOK` messages *were not* required for Π_{AWPS}). The parties then construct the consistency graph. Next D *removes* all the parties from consideration in its consistency graph, who have made public "incorrect" `NOK` messages, whose version of the disputed points are *incorrect.* Among the *remaining* parties, D checks for the presence of a set of at least $n - t$ parties $\mathcal{W}$, such that the polynomial of *every* party in $\mathcal{W}$ is publicly confirmed to be pair-wise consistent with the polynomials of at least $n - t$ parties within $\mathcal{W}$. If a $\mathcal{W}$ is found, then D checks for the presence of an $(n, t_s)-$star, say $(\mathcal{E}, \mathcal{F})$ among $\mathcal{W}$ and broadcasts $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ during the *fourth* phase, if D finds $(\mathcal{E}, \mathcal{F})$. The parties upon receiving $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ verify if $\mathcal{W}$ is of size at least $n - t_s$ and every party in $\mathcal{W}$ has an edge with at least $n - t_s$ parties within $\mathcal{W}$ in their local copy of the consistency graph. The parties also check whether indeed $(\mathcal{E}, \mathcal{F})$ constitutes an $(n, t_s)-$star among the parties within $\mathcal{W}$. Furthermore, the parties now *additionally* verify whether any pair of parties P_j, P_k from $\mathcal{W}$ have made public "conflicting" `NOK` messages during the *third* phase. That is, if there exists any $P_j, P_k \in \mathcal{W}$ who made public `NOK` messages with q_{jk} and q_{kj} respectively during the *third* phase such that $q_{jk} \neq q_{kj}$, then $\mathcal{W}$ is *not* accepted. The idea here is that if D is *honest*, then at least one of P_j, P_k is bound to be *corrupt*, whose corresponding `NOK` message is incorrect. Since D also would have seen these public `NOK` messages during the third phase, it should have have discarded the corresponding corrupt party, before finding $\mathcal{W}$. Hence if $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted at the end of fourth phase, then the polynomials of all *honest* parties in $\mathcal{W}$ are guaranteed to be pair-wise consistent and lie on a single (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^\star(x, y)$, where $Q^\star(x, y) = Q(x, y)$

for an *honest* D. This will further guarantee that the polynomials of all *honest* parties in $\mathcal{F}$ *also* lie on $Q^{\star}(x, y)$, as $\mathcal{F} \subseteq \mathcal{W}$.

If $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, then each $P_i \in \mathcal{W}$ outputs the constant term of its univariate polynomial as its share. On the other hand, the parties outside $\mathcal{W}$ attempt to obtain their corresponding polynomials lying on $Q^{\star}(x, y)$ by applying OEC on the common points on these polynomials received from the parties in $\mathcal{F}$ and if a t_s-degree polynomial is obtained, then the constant term of the polynomial is set as the share. For an *honest* D, each *honest* P_i will be present in $\mathcal{W}$ and hence will have the share $q(\alpha_i)$. On the other hand, if a $\mathcal{W}$ is accepted for a *corrupt* D, then all the *honest* parties in $\mathcal{W}$ (which are at least $t_s + 1$ in number) will have their shares lying on $q^{\star}(\cdot) = Q^{\star}(0, y)$. Moreover, even if an *honest* party P_i *outside* $\mathcal{W}$ is able to compute its share, then it is the same as $q^{\star}(\alpha_i)$ due to the OEC mechanism. However, for a *corrupt* D, all the *honest* parties *outside* $\mathcal{W}$ may *not* be able to obtain their desired share, as $\mathcal{F}$ is guaranteed to have only $n - 2t_s > t_s + t_a$ *honest* parties and OEC may fail. It is precisely for this reason that Π_{SWPS} *fails* to qualify as a VSS.

$\Pi_{\mathsf{SWPS}} + \Pi_{\mathsf{AWPS}} \Rightarrow$ **Best-of-Both-Worlds WPS** Π_{WPS} In Π_{WPS} (Fig 4), the parties first run Π_{SWPS} *assuming* a *synchronous* network, where Π_{BC} is used to make any value public by setting $t = t_s$. If D is *honest* then in a *synchronous* network, the first, second, third and fourth phase of Π_{SWPS} would have been over by time $\Delta, 2\Delta, 2\Delta + T_{\mathsf{BC}}$ and $2\Delta + 2T_{\mathsf{BC}}$ respectively and by time $2\Delta + 2T_{\mathsf{BC}}$, the parties should have accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$. However, in an *asynchronous* network, parties may have different opinion regarding the acceptance of $(\mathcal{W}, \mathcal{E}, \mathcal{F})$. Hence at time $2\Delta + 2T_{\mathsf{BC}}$, they run protocol Π_{BA}. If the parties conclude that a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, then the parties compute their *WPS-shares* as per Π_{SWPS}. However, we need to ensure that for a *corrupt* D in a *synchronous* network, if the polynomials of the *honest* parties in $\mathcal{W}$ are *not* pair-wise consistent, then the corresponding conflicting `NOK` messages are received within time $2\Delta + T_{\mathsf{BC}}$ (the time required for the *third* phase of Π_{SWPS} to be over), so that $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is *not* accepted. This is ensured by enforcing even a *corrupt* D to send the respective polynomials of all the *honest* parties in $\mathcal{W}$ by time Δ, so that the pair-wise consistency test between every pair of *honest* parties in $\mathcal{W}$ is over by time 2Δ. For this, the parties are asked to wait for some "appropriate" time, before starting the pair-wise consistency tests and also before making public the results of pair-wise consistency tests. The idea is to ensure that if the polynomials of the *honest* parties in $\mathcal{W}$ are *not* delivered within time Δ, then the results of the pair-wise consistency tests also get *delayed* beyond time $2\Delta + T_{\mathsf{BC}}$ (the time-out of the *third* phase of Π_{SWPS}), thus ensuring that $\mathcal{W}$ is *not* accepted.

If the parties conclude that no $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted at time $2\Delta + 2T_{\mathsf{BC}}$, then either D is *corrupt* or the network is *asynchronous* and hence parties resort to Π_{AWPS}. However, D *need not* have to use a "fresh" bivariate polynomial. Instead, D continues with the consistency graph formed using the `OK` messages received as part of Π_{SWPS} and searches for an $(n, t_a)-$star. If D is *honest* and the network is *asynchronous*, then the parties eventually obtain their shares.

Notice that in an *asynchronous* network, it might be possible that the parties (through Π_{BA}) conclude that $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, if some honest party(ies) accepts $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ within the time-out $2\Delta + 2T_{\mathsf{BC}}$. Even in this case, the polynomials of all *honest* parties in $\mathcal{W}$ lie on a single (t_s, t_s)-degree symmetric bivariate polynomial $Q^{\star}(x, y)$ for a *corrupt* D. This is because there will be at least $n - 2t_s - t_a > t_s$ *honest* parties in $\mathcal{E}$ with pair-wise consistent polynomials, defining $Q^{\star}(x, y)$, and the polynomial of every *honest* party in $\mathcal{F}$ will be pair-wise consistent with the polynomials of *all honest* parties in $\mathcal{E}$ and hence lie on $Q^{\star}(x, y)$. Now consider any *honest* $P_i \in (\mathcal{W} \setminus \mathcal{F})$. As part of Π_{SWPS}, it is ensured that the polynomial of P_i is consistent with the polynomials of at least $n - t_s$ parties among $\mathcal{W}$. Among these $n - t_s$ parties, at least $n - 2t_s - t_a > t_s$ will be *honest* parties from $\mathcal{F}$. Thus, the polynomial of P_i will also lie on $Q^{\star}(x, y)$.

Protocol $\Pi_{\mathsf{WPS}}(\mathsf{D}, q(\cdot))$

- **Phase I (Time Δ) — Sending Polynomials**:
  - D on having the inputs $q(\cdot)$ chooses a random (t_s, t_s)-degree symmetric bivariate polynomial $Q(x, y)$ such that $Q(0, y) = q(\cdot)$ and sends $q_i(x) = Q(x, \alpha_i)$ to each party $P_i \in \mathcal{P}$.
- **Phase II (Time Δ) — Pair-Wise Consistency**: Each P_i on receiving a t_s-degree polynomial $q_i(x)$ from D does the following.
  - Wait till the local time becomes a multiple of Δ and then send $q_{ij} = q_i(\alpha_j)$ to P_j, for $j = 1, \ldots, n$.
- **Phase III (Time T_{BC}) — Publicly Declaring the Results of Pair-Wise Consistency Test**: Each $P_i \in \mathcal{P}$ does the following.
  - Upon receiving q_{ji} from P_j, wait till the local time becomes a multiple of Δ. Then broadcast $\mathtt{OK}(i, j)$ if $q_{ji} = q_i(\alpha_j)$ holds, else broadcast $\mathtt{NOK}(i, j, q_i(\alpha_j))$ if $q_{ji} \neq q_i(\alpha_j)$.
- **Local computation — Constructing Consistency Graph:** Each $P_i \in \mathcal{P}$ does the following.
  - Construct a *consistency graph* G_i over $\mathcal{P}$, where the edge (P_j, P_k) is included in G_i, if $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ is received from the broadcast of P_j and P_k respectively, either through the regular-mode or fall-back mode.
- **Phase IV (Time T_{BC}) — Checking for $(n, t_s)-$star**: D does the following in its consistency graph G_{D} at time $2\Delta + T_{\mathsf{BC}}$.
  - Remove edges incident with P_i, if $\mathtt{NOK}(i, j, q_{ij})$ is received from the broadcast of P_i through regular-mode and $q_{ij} \neq Q(\alpha_j, \alpha_i)$. Set $\mathcal{W} = \{P_i : \mathsf{deg}(P_i) \geq n - t_s\}$, where $\mathsf{deg}(P_i)$ denotes the degree of P_i in G_{D}. Remove P_i from $\mathcal{W}$ if P_i is *not* incident with at least $n - t_s$ parties in $\mathcal{W}$. Repeat this step till no more parties can be removed from $\mathcal{W}$.
  - Run algorithm $\mathsf{AlgStar}$ on $G_{\mathsf{D}}[\mathcal{W}]$, where $G_{\mathsf{D}}[\mathcal{W}]$ denotes the subgraph of G_{D} induced by the vertices in $\mathcal{W}$. If an $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$ is obtained, then broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$.
- **Local Computation — Verifying and Accepting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$:** Each $P_i \in \mathcal{P}$ does the following at time $2\Delta + 2T_{\mathsf{BC}}$.
  - If $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is received from D's broadcast through regular-mode, then *accept* it if following *were true* at time $2\Delta + T_{\mathsf{BC}}$:
    - $\nexists P_j, P_k \in \mathcal{W}$, such that $\mathtt{NOK}(j, k, q_{jk})$ and $\mathtt{NOK}(k, j, q_{kj})$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode, where $q_{jk} \neq q_{kj}$.
    - In G_i, $\mathsf{deg}(P_j) \geq n - t_s$ for all $P_j \in \mathcal{W}$ and P_j has edges with at least $n - t_s$ parties from $G_i[\mathcal{W}]$.
    - $(\mathcal{E}, \mathcal{F})$ *was* an $(n, t_s)-$star in $G_i[\mathcal{W}]$ and $\forall P_j, P_k \in \mathcal{W}$ where the edge (P_j, P_k) is present in G_i, the $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode.
- **Phase V (Time T_{BA}) — Deciding Whether to Go for $(n, t_a)-$star**: At time $2\Delta + 2T_{\mathsf{BC}}$, each $P_i \in \mathcal{P}$ participates in an instance of Π_{BA} with input $b_i = 0$ if a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ was accepted, else with input $b_i = 1$ and waits for time T_{BA}.
- **Local Computation — Computing WPS-share Through $\mathcal{W}$**: If the output of Π_{BA} is 0, then each $P_i \in \mathcal{P}$ computes *WPS-Share* s_i (initially set to $\bot$) as follows.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is not yet received then wait till $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is received from D's broadcast through fall-back mode.
  - If $P_i \in \mathcal{W}$, then output $s_i = q_i(0)$.
  - **(a)** Else, initialise a support set $\mathcal{SS}_i$ to $\emptyset$. Upon receiving q_{ji} from $P_j \in \mathcal{F}$, include q_{ji} to $\mathcal{SS}_i$. Keep executing $\mathsf{OEC}(t_s, t_s, \mathcal{SS}_i)$, till a t_s-degree polynomial $q_i(\cdot)$ is obtained. Then, output $s_i = q_i(0)$.
- **Phase VI (Time T_{BC}) — Broadcasting $(n, t_a)-$star**: If the output of Π_{BA} is 1, then D does the following.
  - After every update in G_{D}, run $\mathsf{AlgStar}$ on G_{D}. If an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ is obtained, then broadcast $(\mathcal{E}', \mathcal{F}')$.
- **Local Computation — Computing WPS-share Through $(n, t_a)-$star**: If the output of Π_{BA} is 1, then each P_i does the following to compute its *WPS-Share*.
  - Waits till an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ is obtained from the broadcast of D, either through regular or fall-back mode. Upon receiving, wait till $(\mathcal{E}', \mathcal{F}')$ becomes an $(n, t_a)-$star in G_i.
  - If $P_i \in \mathcal{F}'$, then output $s_i = q_i(0)$. Else, compute $q_i(x)$ using similar procedure as in step **(a)**

above on the points received from the parties in $\mathcal{F}'$ and output $s_i = q_i(0)$.

Figure 4: The best-of-both-worlds weak polynomial-sharing protocol for a single polynomial.

Π_{WPS} for L Polynomials If D has L polynomials, then it embeds them into L random (t_s, t_s)-degree symmetric bivariate polynomials and distributes the univariate polynomials to respective parties and the parties perform the pair-wise consistency tests. However, P_i broadcasts an $\texttt{OK}(i, j)$ message for P_j if the pair-wise consistency test is positive for *all* the L common values between P_i and P_j. If the test fails for any of the L common values, then P_i broadcasts a *single* $\texttt{NOK}$ message, corresponding to the *least indexed* common value for which the test fails. Hence, a *single* consistency graph is constructed by each party. Similarly, a *single* instance of Π_{BA} is used to decide whether any $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted. The modified protocol will incur a communication of $\mathcal{O}(n^2 L \log |\mathbb{F}| + n^4 \log |\mathbb{F}|)$ bits and invokes 1 instance of Π_{BA}. For the formal details and the proof of Theorem 4.1, see Appendix D.

Theorem 4.1. *Let $n > 3t_s + t_a$ and let D has L number of t_s-degree polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ as input for Π_{WPS} where $L \geq 1$ and let $T_{\mathsf{WPS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Then protocol Π_{WPS} achieves the following properties.*

- **Correctness**: *If D is honest then:* **(a)** *In a synchronous network, each (honest) P_i outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ at time T_{WPS}.* **(b)** *In an asynchronous network, almost-surely, each (honest) P_i eventually outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.*
- **Privacy**: *If D is honest, then the view of adversary remains independent of $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$.*
- **Weak-Commitment**: *If D is corrupt, then either no honest party obtains any output or there exist t_s-degree polynomials $\{q^{\star(\ell)}(\cdot)\}_{\ell=1,\ldots,L}$, such that* **(a)** *In an asynchronous network, almost-surely, each (honest) P_i eventually outputs $\{q^{\star(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.* **(b)** *In a synchronous network, at least $t_s + 1$ honest parties P_i output $\{q^{\star(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ and if any honest P_j outputs $s_j^{(1)}, \ldots, s_j^{(\ell)} \in \mathbb{F}$, then $s_j^{(\ell)} = q^{\star(\ell)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$.*

4.2 The VSS Protocol

Protocol Π_{WPS} fails to serve as a VSS because if D is *corrupt* and the network is *synchronous*, then the (honest) parties *outside* $\mathcal{W}$ may not obtain their desired shares. Protocol Π_{VSS} (see Fig 11) fixes this shortcoming. For ease of understanding, we present the protocol assuming D has a single t_s-degree polynomial as input and later discuss the modifications needed when D has L such polynomials. The protocol has two "layers" of communication involved. The first layer is similar to Π_{WPS} and identifies whether the parties accepted some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ within a specified time-out, such that the polynomials of all honest parties in $\mathcal{W}$ lie on a single (t_s, t_s)-degree symmetric bivariate polynomial $Q^\star(x, y)$. If some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, then the second layer of communication (which is coupled with the first layer) enables even the (honest) parties outside $\mathcal{W}$ to get their corresponding polynomials lying on $Q^\star(x, y)$.

In a more detail, to perform the pair-wise consistency check of the polynomials received from D, each P_j upon receiving $q_j(x)$ from D, shares the polynomial $q_j(x)$ by invoking an instance of Π_{WPS} as a dealer. Any party P_i who computes a WPS-Share in this instance of Π_{WPS} either broadcasts an $\texttt{OK}$ or $\texttt{NOK}$ message for P_j, depending on whether the WPS-share lies on the polynomial which P_i has received from D. The rest of the steps for computing $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ and accepting it remains the same. If some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, then any P_i *outside* $\mathcal{W}$ computes its polynomial lying on $Q^\star(x, y)$ as follows: P_i checks for a subset $\mathcal{SS}_i \subseteq \mathcal{F}$ of $t_s + 1$ parties P_j, such that P_i is able to compute its WPS-share in the Π_{WPS} instance invoked by P_j as a dealer. Such an $\mathcal{SS}_i$ is bound to exist as there are at least $t_s + 1$ *honest* parties in $\mathcal{F}$ who are always included in $\mathcal{SS}_i$. While the WPS-shares corresponding to the *honest* parties in $\mathcal{SS}_i$ will be the common points on $Q^\star(x, \alpha_i)$, the same holds even for *corrupt* parties in $\mathcal{SS}_i$. This is because in order to be included in $\mathcal{F}$, such parties are "forced" to share polynomials lying on $Q^\star(x, y)$, in their respective instances of Π_{WPS}. Now using the WPS-shares corresponding to the parties in $\mathcal{SS}_i$, party P_i will be able to compute $Q^\star(x, \alpha_i)$ and hence, its share.

Protocol $\Pi_{\mathsf{VSS}}(\mathsf{D}, q(\cdot))$

- **Phase I (Time Δ) — Sending Polynomials**: same as in the protocol Π_{WPS}.
- **Phase II (Time T_{WPS}) — Exchanging Common Values**: Each $P_i \in \mathcal{P}$, upon receiving a t_s-degree polynomial $q_i(x)$ from D, waits till the current local time becomes a multiple of Δ and then does the following.
  - Act as a dealer and invoke an instance $\Pi_{\mathsf{WPS}}^{(i)}$ of Π_{WPS} to share $q_i(x)$. For $j = 1, \ldots, n$, participate in the instance $\Pi_{\mathsf{WPS}}^{(j)}$, if invoked by P_j as a dealer, and wait for time T_{WPS}.
- **Phase III (Time T_{BC}) — Publicly Declaring the Results of Pair-Wise Consistency Test**: Each $P_i \in \mathcal{P}$ waits till the local time becomes a multiple of Δ and then does the following.
  - If a WPS-share q_{ji} is computed in $\Pi_{\mathsf{WPS}}^{(j)}$ then broadcast $\mathtt{OK}(i, j)$ if $q_{ji} = q_i(\alpha_j)$ holds, else broadcast $\mathtt{NOK}(i, j, q_i(\alpha_j))$.
- **Local Computation — Constructing Consistency Graph**: Each P_i constructs the *consistency graph* G_i as in Π_{WPS}.
- **Phase IV (Time T_{BC}) — Constructing (n, t_s)−star**: D does the following in its consistency graph G_{D} at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$.
  - Check for $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ as done in Π_{WPS} and if a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is computed, then broadcast it.
- **Local Computation — Verifying and Accepting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$**: Each $P_i \in \mathcal{P}$ does the following at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$.
  - *Accept* any $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ received from D's broadcast through regular-mode, if the conditions for accepting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ in Π_{WPS} were true at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$.
- **Phase V (Time T_{BA}) — Deciding Whether to Go for (n, t_a)−star**: At time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$, each party P_i participates in an instance of Π_{BA} with input $b_i = 0$ if a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, else with input $b_i = 1$ and waits for time T_{BA}.
- **Local Computation — Computing VSS-Share Through $(\mathcal{W}, \mathcal{E}, \mathcal{F})$:** If Π_{BA} outputs 0, then each P_i does the following.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is not yet received, then wait till it is received from D's broadcast through fall-back mode.
  - If $P_i \in \mathcal{W}$, then output $q_i(0)$.
  - Else, initialize $\mathcal{SS}_i$ to $\emptyset$. Include $P_j \in \mathcal{F}$ to $\mathcal{SS}_i$ if a WPS-share q_{ji} is computed during $\Pi_{\mathsf{WPS}}^{(j)}$. Wait till $|\mathcal{SS}_i| \geq t_s + 1$. Then interpolate $\{(\alpha_j, q_{ij})\}_{P_j \in \mathcal{SS}_i}$ to get a t_s-degree polynomial $q_i(x)$ and output $q_i(0)$.
- **Phase VI (Time T_{BC}) — Broadcasting (n, t_a)−star**: If the output of Π_{BA} is 1, then D runs $\mathsf{AlgStar}$ after every update in its consistency graph G_{D} and broadcasts $(\mathcal{E}', \mathcal{F}')$, if it finds a (n, t_a)−star $(\mathcal{E}', \mathcal{F}')$.
- **Local Computation — Computing VSS-Share Through (n, t_a)−star**: If Π_{BA} outputs 1, then each P_i does the following.
  - Participate in any instance of Π_{BC} invoked by D for broadcasting an (n, t_a)−star *only* after time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Wait till some $(\mathcal{E}', \mathcal{F}')$ is obtained from D's broadcast (through any mode), which constitutes an (n, t_a)−star in G_i.
  - If $P_i \in \mathcal{F}'$, then output $q_i(0)$. Else, include $P_j \in \mathcal{F}'$ to $\mathcal{SS}_i$ (initialized to $\emptyset$) if a WPS-share q_{ji} is computed in $\Pi_{\mathsf{WPS}}^{(j)}$. Wait till $|\mathcal{SS}_i| \geq t_s + 1$. Then interpolate $\{(\alpha_j, q_{ij})\}_{P_j \in \mathcal{SS}_i}$ to get a t_s-degree polynomial $q_i(x)$ and output $q_i(0)$.

Figure 5: Best-of-Both-Worlds VSS protocol for a single polynomial.

If D has L polynomials, then we use similar modifications as done for Π_{WPS} handling L polynomials, with each party broadcasting a *single* $\mathtt{OK}/\mathtt{NOK}$ message for every other party. The protocol requires a communication of $\mathcal{O}(n^3 L \log |\mathbb{F}| + n^5 \log |\mathbb{F}|)$ bits and $n+1$ instances of Π_{BA}. The protocol and proof of Theorem 4.2 is available in Appendix D.

Theorem 4.2. *Let $n > 3t_s + t_a$ and let D has L number of t_s-degree polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ as input for Π_{VSS} where $L \geq 1$ and let $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Then protocol Π_{VSS} achieves the following properties.*

- **Correctness***: If D is honest, then:* **(a)** *In a synchronous network, each (honest) P_i outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ at time T_{VSS}.* **(b)** *In an asynchronous network, almost-surely, each (honest) P_i eventually outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.*
- **Privacy***: If D is honest, then the view of adversary remains independent of $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$.*
- **Strong-Commitment***: If D is corrupt, then either honest parties obtain no output or there exist t_s-degree polynomials $\{q^{\star(\ell)}(\cdot)\}_{\ell=1,\ldots,L}$, such that.* **(a)** *In an asynchronous network, almost-surely, each honest P_i eventually outputs $\{q^{\star(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.* **(b)** *In a synchronous network, every honest P_i eventually outputs $\{q^{\star(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$, such that if any honest P_i obtains its output at time T_{VSS}, then all honest parties obtain their output at time T_{VSS}. If any honest P_i obtains its output at time $T > T_{\mathsf{VSS}}$, then every honest party obtains its output by time $T + 2\Delta$.*

5 Protocol for Generating Shared Random Multiplication-Triples

Here, we give a high level overview of how to deploy the framework of [23] for generating shared random multiplication-triples and defer to Appendix E for the complete details. We first review some best-of-both-worlds building blocks.

Beaver's Multiplication Protocol: Given t_s-shared x, y and a t_s-shared triple (a, b, c), protocol Π_{Beaver} outputs a t_s-shared z, where $z = x \cdot y$ iff $c = a \cdot b$. If (a, b, c) is random for the adversary, then x and y remain random for the adversary. In the protocol, the parties first locally compute $[x - a]$ and $[y - b]$ and *publicly* reconstruct $x - a$ and $y - b$ by exchanging their shares and applying the OEC procedure. The parties then locally compute $[z] = (x - a) \cdot (y - b) + (x - a) \cdot [b] + (y - b) \cdot [a] + [c]$.

Triple-Transformation Protocol: Protocol $\Pi_{\mathsf{TripTrans}}$ "transforms" a set of t_s-shared triples $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i=1,\ldots,2d+1}$ into "co-related" t_s-shared triples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})\}_{i=1,\ldots,2d+1}$, such that: **(a):** There exist d-degree polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $2d$-degree polynomial $\mathsf{Z}(\cdot)$, such that $\mathsf{X}(\alpha_i) = \mathbf{x}^{(i)}$, $\mathsf{Y}(\alpha_i) = \mathbf{y}^{(i)}$ and $\mathsf{Z}(\alpha_i) = \mathbf{z}^{(i)}$ holds for $i = 1, \ldots, 2d+1$. **(b):** $\mathbf{z}^{(i)} = \mathbf{x}^{(i)} \cdot \mathbf{y}^{(i)}$ iff $z^{(i)} = x^{(i)} \cdot y^{(i)}$. Hence, $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$, iff *all* the input triples are multiplication-triples. **(c):** Adversary learns $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ iff it knows $(x^{(i)}, y^{(i)}, z^{(i)})$. The protocol requires d invocations of Π_{Beaver}.

Agreement on a Common Subset: In Π_{ACS}, each P_i wants to distribute points on L number of t_s-degree polynomials through Π_{VSS}. As *corrupt* parties may not invoke its instances of Π_{VSS}, the parties may obtain points on the polynomials of only $n - t_s$ parties (even in a *synchronous* network). In an *asynchronous* network, different parties may obtain points on the polynomials of different subsets of $n - t_s$ parties. Π_{ACS} lets the parties agree on a *common subset* $\mathcal{CS}$ of at least $n - t_s$ parties, whose points are received by all honest parties, such that in a *synchronous* network, all *honest* parties are guaranteed to be in $\mathcal{CS}$. The idea is to invoke n instances of Π_{BA}, where the j^{th} instance is to decide if $P_j \in \mathcal{CS}$.

Triple-Sharing Protocol: Protocol Π_{TripSh} allows a dealer D to *verifiably* t_s-share multiplication-triples, where if D is *honest*, the triples remain random for the adversary. For a *corrupt* D, the protocol *need not* produce any output even in a *synchronous* network (as D may not invoke the protocol). However, the "verifiability" of Π_{TripSh} guarantees that if the honest parties obtain any output for a *corrupt* D, then D has t_s-shared multiplication-triples.

Triple-Extraction Protocol: In protocol Π_{TripExt}, there will be a *publicly-known* set of $2d+1$ parties $\mathcal{CS}$ where $d \geq t_s$ and where it will be *ensured* that *each* $P_j \in \mathcal{CS}$ has t_s-shared a multiplication-triple which is random for the adversary, if P_j is *honest*. The protocol outputs $d+1-t_s$ number of t_s-shared multiplication-triples which will be random for the adversary. The protocol requires one instance of $\Pi_{\mathsf{TripTrans}}$ followed by local computation.

5.1 The Triple-Generation Protocol

In protocol Π_{TripGen}, each party shares $L = \frac{c_M}{(\frac{n-t_s-1}{2}+1-t_s)}$ random multiplication-triples through Π_{TripSh}. The parties then agree on a *common* subset $\mathcal{CS}$ of $n-t_s$ parties, who have shared multiplication-triples, by executing instances of Π_{BA}. The multiplication-triples of *corrupt* triple-providers in $\mathcal{CS}$ will be known to adversary. Thus, L instances of Π_{TripExt} are executed to securely extract c_M shared multiplication-triples, which will be random for the adversary.

6 The Circuit-Evaluation Protocol

Protocol Π_{CirEval} for evaluating cir has four phases. In the first phase, the parties generate t_s-sharing of c_M random multiplication-triples through Π_{TripGen}. They also invoke Π_{ACS} to generate t_s-sharing of their respective inputs for f and agree on a *common* subset $\mathcal{CS}$ of *at least* $n-t_s$ parties, whose inputs for f are t_s-shared, while the remaining inputs are set to 0. In a *synchronous* network, all *honest* parties will be in $\mathcal{CS}$. In the second phase, each gate is evaluated in a t_s-shared fashion after which the parties *publicly* reconstruct the secret-shared output in the third phase. The fourth phase is the *termination phase*, where the parties check whether "sufficiently many" parties have obtained the same output, in which case the parties "safely" take that output and terminate the protocol (and all the underlying sub-protocols). For the formal details of Π_{CirEval} and proof of Theorem 6.1, see Appendix F.

Theorem 6.1. *Let $t_a < t_s$, such that $3t_s + t_a < n$. Moreover, let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function represented by an arithmetic circuit cir over $\mathbb{F}$ consisting of c_M number of multiplication gates, and whose multiplicative depth is D_M. Then, Π_{CirEval} incurs a communication of $\mathcal{O}(\frac{n^5}{\frac{t_a}{2}+1} c_M \log|\mathbb{F}| + n^7 \log|\mathbb{F}|)$ bits, invokes $\mathcal{O}(n^3)$ instances of Π_{BA} and achieves the following.*

- *Correctness:* **(a)** *In a synchronous network, all honest parties output $y = f(x^{(1)}, \ldots, x^{(n)})$ at time $(120n + D_M + 6k - 20) \cdot \Delta$, where $x^{(j)} = 0$ for every $P_j \notin \mathcal{CS}$, such that $|\mathcal{CS}| \geq n - t_s$ and every honest $P_j \in \mathcal{CS}$; here k is the constant from Lemma 3.2, as determined by the underlying (existing) perfectly-secure ABA protocol Π_{ABA}.* **(b)** *In an asynchronous network, almost-surely, the honest parties eventually output $y = f(x^{(1)}, \ldots, x^{(n)})$ where $x^{(j)} = 0$ for every $P_j \notin \mathcal{CS}$ and where $|\mathcal{CS}| \geq n - t_s$.*
- *Privacy: The view of the adversary will be independent of the inputs of the honest parties in $\mathcal{CS}$.*

Conclusion and Open Problems In this work, we presented the *first* perfectly-secure MPC protocol which remains secure both in a synchronous as well as an asynchronous network. Our work leaves the following interesting open problems: **(a)**: We could not prove a *necessary* condition for best-of-both-worlds perfectly-secure MPC protocol and conjecture that it is $3t_s + t_a < n$. **(b)**: Our main focus in this work is on the *existence* of best-of-both-worlds perfectly-secure MPC protocols. Improving the efficiency of the protocol is left for future work.

References

[1] Lecture 10: Consensus. https://www.mpi-inf.mpg.de/fileadmin/inf/d1/teaching/summer19/tkds/Lec10.pdf, 2019.

[2] I. Abraham, G. Asharov, and A. Yanai. Efficient Perfectly Secure Computation with Optimal Resilience. In *TCC*, volume 13043 of *Lecture Notes in Computer Science*, pages 66–96. Springer, 2021.

[3] 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.

[4] I. Abraham, D. Dolev, and G. Stern. Revisiting Asynchronous Fault Tolerant Computation with Optimal Resilience. In *PODC*, pages 139–148. ACM, 2020.

[5] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin. Sync HotStuff: Simple and Practical Synchronous State Machine Replication. In *IEEE Symposium on Security and Privacy*, pages 106–118. IEEE, 2020.

[6] G. Asharov and Y. Lindell. A Full Proof of the BGW Protocol for Perfectly Secure Multiparty Computation. *J. Cryptology*, 30(1):58–151, 2017.

[7] H. Attiya and J. L. Welch. *Distributed computing - fundamentals, simulations, and advanced topics (2. ed.)*. Wiley series on parallel and distributed computing. Wiley, 2004.

[8] 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.

[9] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

[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, 2007.

[11] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2008.

[12] M. Ben-Or. Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols (Extended Abstract). In *PODC*, pages 27–30. ACM, 1983.

[13] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous Secure Computation. In *STOC*, pages 52–61. ACM, 1993.

[14] 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.

[15] P. Berman, J. A. Garay, and K. J. Perry. Bit Optimal Distributed Consensus. In *Computer Science Research*, pages 313–322. Springer, 1992.

[16] 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.

[17] E. Blum, C. L. Zhang, and J. Loss. Always Have a Backup Plan: Fully Secure Synchronous MPC with Asynchronous Fallback. In *CRYPTO*, volume 12171 of *Lecture Notes in Computer Science*, pages 707–731. Springer, 2020.

[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 and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *STOC*, pages 42–51. ACM, 1993.

[21] A. Chandramouli, A. Choudhury, and A. Patra. A Survey on Perfectly-Secure Verifiable Secret-Sharing. *IACR Cryptol. ePrint Arch.*, page 445, 2021.

[22] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *FOCS*, pages 383–395. IEEE Computer Society, 1985.

[23] A. Choudhury and A. Patra. An Efficient Framework for Unconditionally Secure Multiparty Computation. *IEEE Trans. Information Theory*, 63(1):428–468, 2017.

[24] R. Cramer and I. Damgård. *Multiparty Computation, an Introduction. Contemporary Cryptography.* Birkhåuser Basel, 2005.

[25] I. Damgård and J. B. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer Verlag, 2007.

[26] G. Deligios, M. Hirt, and C. Liu-Zhang. Round-Efficient Byzantine Agreement and Multi-party Computation with Asynchronous Fallback. In *TCC*, volume 13042 of *Lecture Notes in Computer Science*, pages 623–653. Springer, 2021.

[27] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *J. ACM*, 40(1):17–47, 1993.

[28] P. Feldman and S. Micali. An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.

[29] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382, 1985.

[30] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer, 2006.

[31] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *STOC*, pages 580–589. ACM, 2001.

[32] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In *PODC*, pages 101–111. ACM, 1998.

[33] O. Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications.* Cambridge University Press, 2004.

[34] V. Goyal, Y. Liu, and Y. Song. Communication-Efficient Unconditional MPC with Guaranteed Output Delivery. In *CRYPTO*, volume 11693 of *Lecture Notes in Computer Science*, pages 85–114. Springer, 2019.

[35] Y. Guo, R. Pass, and E. Shi. Synchronous, with a Chance of Partition Tolerance. In *CRYPTO*, volume 11692 of *Lecture Notes in Computer Science*, pages 499–529. Springer, 2019.

[36] J. Katz, C. Y. Koo, and R. Kumaresan. Improving the Round Complexity of VSS in Point-to-point Networks. *Inf. Comput.*, 207(8):889–899, 2009.

[37] Nancy A Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

[38] D. Malkhi, K. Nayak, and L. Ren. Flexible Byzantine Fault Tolerance. In *CCS*, pages 1041–1053. ACM, 2019.

[39] R. J. McEliece and D. V. Sarwate. On Sharing Secrets and Reed-Solomon Codes. *Commun. ACM*, 24(9):583–584, 1981.

[40] 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.

[41] A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation. *J. Cryptology*, 28(1):49–109, 2015.

[42] A. Patra and D. Ravi. On the Power of Hybrid Networks in Multi-Party Computation. *IEEE Trans. Information Theory*, 64(6):4207–4227, 2018.

[43] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.

[44] M. O. Rabin. Randomized Byzantine Generals. In *FOCS*, pages 403–409. IEEE Computer Society, 1983.

[45] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

A Properties of the Existing (Asynchronous) Primitives

In this section we discuss the existing asynchronous primitives in detail.

A.1 Online Error-Correction (OEC)

The OEC procedure uses a Reed-Solomon (RS) error-correcting procedure $\mathsf{RSDec}(d, r, \mathcal{W})$ that takes as input a set $\mathcal{W}$ of distinct points on a d-degree polynomial and tries to output a d-degree polynomial, by correcting at most r *incorrect* points in $\mathcal{W}$. Coding theory [39] says that RS-Dec can correct up to r errors in $\mathcal{W}$ and correctly interpolate back the original polynomial if and only if $|\mathcal{W}| \geq d + 2r + 1$ holds. There are several efficient implementations of RSDec (for example, the algorithm of Berlekamp-Welch).

Suppose $\mathcal{P}' \subseteq \mathcal{P}$ contains at most t corrupt parties and let there exist some d-degree polynomial $q(\cdot)$ with every (honest) $P_i \in \mathcal{P}'$ having a point $q(\alpha_i)$. The goal is to make some *designated* party P_R reconstruct $q(\cdot)$. For this, each $P_i \in \mathcal{P}'$ sends $q(\alpha_i)$ to P_R, who then applies the OEC procedure OEC as described in Fig 6.

Protocol $\mathsf{OEC}(d, t, \mathcal{P}')$

Setting: There exists a subset of parties $\mathcal{P}'$ containing at most t corrupt parties, with each $P_i \in \mathcal{P}'$ having a point $q(\alpha_i)$ on some d-degree polynomial $q(\cdot)$. Every (honest) party in $\mathcal{P}'$ is supposed to send its respective point to P_R, who is designated to reconstruct $q(\cdot)$.

- **Output Computation** — For $r = 0, \ldots, t$, party P_R does the following in iteration r:
  - Let $\mathcal{W}$ denote the set of parties in $\mathcal{P}'$ from whom P_R has received the points and let $\mathcal{I}_r$ denote the points received from the parties in $\mathcal{W}$, when $\mathcal{W}$ contains exactly $d + t + 1 + r$ parties.
  - Wait until $\mathcal{W} \geq d + t + 1 + r$. Execute $\mathsf{RSDec}(d, r, \mathcal{I}_r)$ to get a d-degree polynomial $q_r(\cdot)$. If no polynomial is obtained, then skip the next step and proceed to the next iteration.
  - If for at least $d + t + 1$ values $v_i \in \mathcal{I}_r$ it holds that $q_r(\alpha_i) = v_i$, then output $q_r(\cdot)$. Otherwise, proceed to the next iteration.

Figure 6: The online error-correction procedure.

Lemma A.1 ([19]). *Let $\mathcal{P}' \subseteq \mathcal{P}$ contain at most t corrupt parties and let there exist some d-degree polynomial $q(\cdot)$ with every (honest) $P_i \in \mathcal{P}'$ having a point $q(\alpha_i)$. Then the OEC protocol prescribed in Fig 6 achieves the following for an honest P_R in the presence of up to t corruptions.*

- **Correctness***:*
  - *If $d < (|\mathcal{P}'| - 2t)$, then in a a synchronous network, P_R outputs $q(\cdot)$ by the end of Δ time and in an asynchronous network, P_R eventually outputs $q(\cdot)$.*
  - *If P_R obtains any output, then irrespective of the network type, the output polynomial is the same as $q(\cdot)$.*
- **Communication Complexity***: The protocol incurs a communication of $\mathcal{O}(n \log |\mathbb{F}|)$ bits.*

Proof. The *communication complexity* follows from the fact that each party send its point to P_R. For the second part of *correctness*, we show that if P_R outputs a d-degree polynomial, say $q_r(\cdot)$, during the iteration number r, then $q_r(\cdot)$ is the same as $q(\cdot)$ (irrespective of the network type). However, this follows from the fact that $q_r(\cdot)$ is consistent with $d + t + 1$ values from $\mathcal{I}_r$, out of which at least $d + 1$ values belong to the *honest* parties and thus, they lie on the polynomial $q(\cdot)$ as well. Furthermore, two *different* d-degree polynomials can have at most d distinct points in common.

We next prove the first part of the *correctness*, assuming an *asynchronous* network. We first argue that an *honest* P_R *eventually* obtains some output, provided $d < (|\mathcal{P}'| - 2t)$. Let adversary control $\hat{r}$ parties in $\mathcal{P}'$, where $\hat{r} \leq t$. Assume that $\hat{r}_1$ corrupt parties send incorrect points to P_R and the remaining $\hat{r}_2 = \hat{r} - \hat{r}_1$ corrupt parties do not send anything at all. Then, consider iteration number $t - \hat{r}_2$. Since $\hat{r}_2$ parties never send any value, P_R will receive at least $d + t + 1 + t - \hat{r}_2$ distinct values on $q(\cdot)$, of which $\hat{r}_1$ could be corrupted. Since $|\mathcal{I}_{d+t+1+t-\hat{r}_2}| \geq d + 2\hat{r}_1 + 1$ holds, the algorithm RSDec will correct $\hat{r}_1$ errors and will return the polynomial $q(\cdot)$ during the iteration number $t - \hat{r}_2$. Therefore P_R will obtain an output latest after $(t - \hat{r}_2)$ iterations.

The proof of the first part of *correctness* in the *synchronous* network is the same as above. In this case, it should be noted that the points of all honest parties reach P_R within Δ time. □

A.2 Bracha's Acast Protocol

Bracha's Acast protocol [18] tolerating $t < n/3$ corruptions is presented in Fig 7.

Protocol Π_{ACast}

1. If $P_i = \mathsf{S}$, then on input m, send $(\texttt{init}, \mathsf{S}, m)$ to all the parties.
2. Upon receiving the message $(\texttt{init}, \mathsf{S}, m)$ from S, send $(\texttt{echo}, \mathsf{S}, m)$ to all the parties.
3. Upon receiving $(\texttt{echo}, \mathsf{S}, m^\star)$ from $n - t$ parties, send $(\texttt{ready}, \mathsf{S}, m^\star)$ to all the parties.
4. Upon receiving $(\texttt{ready}, \mathsf{S}, m^\star)$ from $t + 1$ parties, send $(\texttt{ready}, \mathsf{S}, m^\star)$ to all the parties.
5. Upon receiving $(\texttt{ready}, \mathsf{S}, m^\star)$ from $n - t$ parties, output $m^\star$.

Figure 7: Bracha's Acast protocol. The above code is executed by every $P_i \in \mathcal{P}$ including the sender S.

We now prove the properties of the protocol Π_{ACast}.

Lemma 2.3. Protocol Π_{ACast} achieves the following properties in the presence of up to $t < n/3$ corruptions.

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

Proof. We first prove the properties assuming an *asynchronous* network with up to t corruptions. We start with the *validity* property, for which we consider an *honest* S. We show that all honest parties eventually output m. This is because all honest parties complete steps $2 - 5$ in the protocol even if the corrupt parties do not send their messages, as there are at least $n - t$ honest parties, whose messages are eventually selected for delivery. Moreover, the adversary may send at most t `echo` messages for m', where $m' \neq m$, on behalf of corrupt parties. Similarly, the adversary may send at most t `ready` messages for m', where $m' \neq m$, on behalf of corrupt parties. Consequently, no honest party ever generates a `ready` message for m', neither in step 3, nor in step 4.

For *consistency*, we consider a *corrupt* S and let P_h be an *honest* party, who outputs $m^\star$. We next show that all honest parties eventually obtain the output $m^\star$. Since P_h obtains the output $m^\star$, it implies that it receives $n - t$ `ready` messages for $m^\star$ during step 5 of the protocol. Let $\mathcal{H}$ be the set of *honest* parties whose `ready` messages are received by P_h during step 5. It is easy to see that $|\mathcal{H}| \geq t + 1$. The `ready` messages of the parties in $\mathcal{H}$ are eventually delivered to every honest party and hence each honest party (including P_h) eventually executes step 4 and sends a `ready` message for $m^\star$. As there are at least $n - t$ honest parties, it follows that eventually $n - t$ `ready` messages for $m^\star$ are delivered to every honest party (irrespective of whether adversary sends all the required messages), consequently guaranteeing that all honest parties eventually obtain some output. To complete the proof, we show that this output is $m^\star$.

On contrary, let $P_{h'}$ be another honest party, different from P_h, who outputs $m^{\star\star} \neq m^\star$. This implies that $P_{h'}$ received `ready` messages for $m^{\star\star}$ from at least $t+1$ *honest* parties during step 5 of the protocol. Now from the protocol steps, it follow that an honest party generates a `ready` message for some potential m, only if it receives $n-t$ `echo` messages for the m during step 3 or $t+1$ `ready` messages for the m (one of which has to come from an honest party) during step 4. So all in all, in order that $n-t$ `ready` messages are eventually generated for some potential m during step 5, it must be the case that some honest party has to receive $n-t$ `echo` messages for m during step 2 and generate a `ready` message for m. Since P_h receives $n-t$ `ready` messages for $m^\star$, some honest party must have received $n-t$ `echo` messages for $m^\star$, at most t of which could come from the corrupt parties. Similarly, since $P_{h'}$ receives $n-t$ `ready` messages for $m^{\star\star}$, some honest party must have received $n-t$ `echo` messages for $m^{\star\star}$. However, since $n-t > 2t$, it follows that in order that $n-t$ `echo` messages are produced for both $m^\star$ as well as $m^{\star\star}$, it must be the case that some honest party must have generated an `echo` message, both for $m^\star$, as well as $m^{\star\star}$ during step 2, which is impossible. This is because an honest party executes step 2 at most once and hence generates an `echo` message at most once.

The proofs of the properties in the *synchronous* network closely follow the proofs of the properties in the *asynchronous* network. If S is *honest*, then it will send the `init` message for m to all the parties, which will be delivered within time Δ. Consequently, every *honest* party will send an `echo` message for m to all the parties, which will be delivered within time 2Δ. Hence every *honest* party will send a `ready` message for m to all the parties, which will be delivered within time 3Δ. As there are at least $n-t$ honest parties, every honest party will receive `ready` messages for m from at least $n-t$ parties within time 3Δ and output m.

If S is *corrupt* and some honest party P_h outputs $m^\star$ at time T, then it implies that P_h has received `ready` messages for $m^\star$ during step 5 of the protocol at time T from a set $\mathcal{H}$ of at least $t+1$ honest parties. These ready messages are guaranteed to be received by every other honest party within time $T+\Delta$. Consequently, every *honest* party who has not yet executed step 4 will do so and will send a `ready` message for $m^\star$ at time $T+\Delta$. Consequently, by the end of time $T+\Delta$, every honest party would have sent a `ready` message for $m^\star$ to every other honest party, which will be delivered within time $T+2\Delta$. Hence, every honest party will output $m^\star$ latest at time $T+2\Delta$.

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

B An Overview of the Existing Perfectly-Secure ABA Protocols [3, 8]

In this section, we give a very high level overview of the existing perfectly-secure ABA protocols of [3, 8]. Both these protocols are perfectly-secure and tolerate up to $t < n/3$ corruptions and follow the standard framework of Rabin and Ben-Or [44, 12], which uses two building-blocks to get a BA protocol. The first building-block is a *voting* protocol and which is a deterministic protocol (often called *gradecast* or *graded consensus* in the literature). The second building-block is a *coin-flipping* protocol which is a randomized protocol. In the sequel, we review these building blocks and review how they are "combined" to get an ABA protocol. While presenting these building-blocks, unless it is explicitly stated, we assume an *asynchronous* network. Also, for simplicity, we present these building-blocks *without* specifying any *termination* criteria and hence, the parties may keep on running these building-blocks (as well as the ABA protocol) even after obtaining an output[4].

B.1 The Voting Protocol

Informally, the voting protocol does "whatever can be done deterministically" to reach agreement. In a voting protocol, every party has a single bit as input. The protocol tries to find out whether there is a detectable majority for some value among the inputs of the parties. In the protocol, each party's output can have *five* different forms:

- For $\sigma \in \{0, 1\}$, the output $(\sigma, 2)$ stands for "overwhelming majority for σ";
- For $\sigma \in \{0, 1\}$, the output $(\sigma, 1)$ stands for "distinct majority for σ";
- The output $(\Lambda, 0)$ stands for "non-distinct majority".

The protocol code taken from [19] is presented in Fig 8.

Protocol Π_{Vote}

- On having the input x_i, Acast $(\mathtt{input}, P_i, x_i)$.
- Create a dynamic set $\mathcal{X}_i$ which is initialized to $\emptyset$. Add (P_j, x_j) to $\mathcal{X}_i$ if $(\mathtt{input}, P_j, x_j)$ is received from the Acast of P_j.
- Wait until $|\mathcal{X}_i| = n - t$. Then assign $X_i = \mathcal{X}_i$, set a_i to the majority bit among $\{x_j \mid (P_j, x_j) \in X_i\}$. Acast $(\mathtt{vote}, P_i, X_i, a_i)$.
- Create a dynamic set $\mathcal{Y}_i$, which is initialized to $\emptyset$. Add (P_j, X_j, a_j) to $\mathcal{Y}_i$ if $(\mathtt{vote}, P_j, X_j, a_j)$ is received from the Acast of P_j, $X_j \subseteq \mathcal{X}_i$, and a_j is the majority bit of X_j.
- Wait until $|\mathcal{Y}_i| = n - t$. Then assign $Y_i = \mathcal{Y}_i$, set b_i to the majority bit among $\{a_j \mid (P_j, X_j, a_j) \in Y_i\}$ and Acast $(\mathtt{re\text{-}vote}, P_i, Y_i, b_i)$.
- Create a set Z_i, which is initialized to $\emptyset$. Add (P_j, Y_j, b_j) to Z_i if $(\mathtt{re\text{-}vote}, P_j, Y_j, b_j)$ is received from the Acast of P_j, $Y_j \subseteq \mathcal{Y}_i$, and b_j is the majority bit of Y_j.
- Wait until $|Z_i| = n - t$. Then compute the output as follows.
  - If all the parties $P_j \in Y_i$ have the same `vote` $a_j = \sigma$, then output $(\sigma, 2)$.
  - Else if all the parties $P_j \in Z_i$ have the same `re-vote` $b_j = \sigma$, then output $(\sigma, 1)$.
  - Else output $(\Lambda, 0)$.

Figure 8: The vote protocol. The above code is executed by every $P_i \in \mathcal{P}$.

The properties of the voting protocol are stated in Lemma B.1. While these properties hold in an *asynchronous* network, it automatically implies that they hold even for a *synchronous* network. We refer the readers to [19, 8] for the proof of these properties.

Lemma B.1 ([19, 8]). *Protocol Π_{Vote} achieves the following properties, both in the synchronous as well as asynchronous network, for every $t < n/3$.*

- *If each honest party has the same input σ, then each honest party outputs $(\sigma, 2)$;*
- *If some honest party outputs $(\sigma, 2)$, then every other honest party outputs either $(\sigma, 2)$ or $(\sigma, 1)$;*
- *If some honest party outputs $(\sigma, 1)$ and no honest party outputs $(\sigma, 2)$ then each honest party outputs either $(\sigma, 1)$ or $(\Lambda, 0)$.*
- *The protocol incurs a communication of $\mathcal{O}(n^3)$ bits.*

An additional property which protocol Π_{Vote} achieves in a *synchronous* network is that all *honest* parties will receive their output by the end of time 9Δ. Intuitively, this is because the protocol involves three different "phases" of Acast, each of which will produce an output within time 3Δ for *honest* sender parties in a *synchronous* network. Moreover, from Lemma B.1, this output will be $(\sigma, 2)$, if all the *honest* parties have the same input σ. We will require this property later while

[4]Recall that we do not put any termination criteria for any of our sub-protocols, as the termination of the MPC protocol will automatically ensure that all the underlying sub-protocols also get terminated.

claiming the properties of the resultant ABA protocol in a *synchronous* network. Hence, we prove this property.

Lemma B.2. *If the network is synchronous, then in protocol* Π_{Vote}*, all honest parties obtain their output within time* 9Δ*. Moreover, the output will be* $(\sigma, 2)$*, if all the honest parties have the same input* σ*.*

Proof. Consider an arbitrary *honest* P_j. Party P_j will Acast its input x_j and from the *liveness* and *validity* properties of Acast in the *synchronous* network, every honest party will receive the output x_j from the corresponding Acast instance within time 3Δ. As there are at least $n-t$ *honest* parties, it implies that every *honest* P_i will obtain a set X_i of size $n-t$ within time 3Δ. Hence each *honest* P_i will Acast a $(\texttt{vote}, P_i, X_i, a_i)$ message latest at time $3\Delta+1$ and every honest party receives this message from the corresponding Acast instance within time 6Δ. We also note that if there is a *corrupt* P_j such that (P_j, x_j) is included by an *honest* P_i in its set X_i when P_i Acasts X_i, then from the *consistency* property of Acast in the *synchronous* network, every *honest* party P_k will include (P_j, x_j) in its set $\mathcal{X}_k$, latest by time 5Δ. This further implies that upon receiving the message $(\texttt{vote}, P_i, X_i, a_i)$ from the Acast of any *honest* P_i, all *honest* parties P_k will be able to verify this message and include (P_i, X_i, a_i) in their respective $\mathcal{Y}_k$ sets within time 6Δ.

As there are at least $n-t$ honest parties P_i whose $\texttt{vote}$ messages are received and verified by all honest parties P_k within time 6Δ, it follows that every honest party Acasts a $\texttt{re-vote}$ message, latest at time $6\Delta+1$, which is received by every honest party within time 9Δ. Moreover, as argued for the case of $\texttt{vote}$ messages, every *honest* party will be able to verify these $\texttt{re-vote}$ messages and include in their respective Z_i within time 9Δ. Since there are at least $n-t$ honest parties, it follows that the Z_i sets of every honest party will attain the size of $n-t$ within time 9Δ and hence every honest party will obtain an output, latest at time 9Δ.

If all the honest parties have the same input σ, then there will be at most t *corrupt* parties who may Acast $1-\sigma$. Hence *every* party (both honest as well as corrupt) will send a $\texttt{vote}$ message only for[5] σ. Consequently, every honest party will output $(\sigma, 2)$. □

B.2 Coin-Flipping Protocol

The *coin-flipping* protocol denoted by Π_{CoinFlip} (also called as the *common-coin* protocol) is an n-party (asynchronous) protocol, where the parties have local random inputs and the protocol outputs a bit for all the parties. The protocol achieves the following properties in an *asynchronous* (and hence *synchronous*) network in the presence of any $t < n/3$ corruptions.

- In an *asynchronous* network, all honest parties eventually obtain an output, while in a *synchronous* network, the honest parties obtain an output within some fixed time $c \cdot \Delta$, where c is a *publicly-known* constant.
- One of the following holds:
  - If no party deviates from the protocol, then with probability *at least* p, the output bits of all the honest parties are same. The probability p where $p < 1$ is often called as the *success-probability* of the protocol and is a parameter of the protocol.
  - Else, all honest parties will have the same output bit with probability *less than* p. But in this case, the protocol allows some *honest* party(ies) to *locally* identify and shun a (subset) of corrupt party(ies) from any future communication. Namely, the protocol *locally* outputs ordered pairs of the form (P_i, P_j) where P_i is some *honest* party and P_j is

[5]If a *corrupt* party P_j sends a $\texttt{vote}$ message for $1-\sigma$, then it will never be accepted and no honest party P_i will ever include $(P_j, X_j, 1-\sigma)$ in its $\mathcal{Y}_i$ set. This is because $1-\sigma$ will not be the majority among the inputs of the honest parties in X_j.

some *corrupt* party, such that P_i identifies P_j as a corrupt party and does not consider any communication from P_j for the rest of the protocol execution. Such pairs are called as *local-conflicts*. We stress that the local-conflicts are identified only *locally*. For instance, if an *honest* P_i has shunned a *corrupt* P_j during an instance of the coin-flipping protocol, then it is *not* necessary that every other *honest* party P_k also shuns P_j during the same instance, as P_j may decide to behave "honestly" towards P_k.

The coin-flipping protocol of [3] guarantees that at least one *new* local-conflict is identified if, during an instance of the coin-flipping protocol, the parties obtain the same output bit with probability less than p. On the other hand, the coin-flipping protocol of [8] guarantees that $\Theta(n)$ number of *new* local-conflicts are identified, if the parties obtain the same output bit with probability less than p.

Protocol Π_{CoinFlip} is designed using a *weaker* variant of perfectly-secure AVSS called *shunning* AVSS (SAVSS), introduced in [3]. The SAVSS primitive is weaker than AVSS in the following aspects:

– It is *not* guaranteed that every honest party obtains a point on D's sharing-polynomial (and hence a share of D's secret), even if D is *honest*;
– If D is *corrupt*, then it may not participate with a t-degree polynomial and hence, the underlying shared value could be $\bot$, which is different from every element of $\mathbb{F}$;
– Irrespective of D, depending upon the behaviour of the corrupt parties, the honest parties later may either reconstruct the same secret as shared by D or an all-together different value. However, in the latter case, the protocol ensures that at least one *new* local-conflict is identified.

In [3], a perfectly-secure SAVSS protocol is designed with $t < n/3$. By executing n^2 instances of this protocol in *parallel* using the framework of [28, 19], a coin-flipping protocol is presented in [3], where the success-probability p is $\frac{1}{4}$. The communication complexity of the protocol is $\mathcal{O}(\text{poly}(n) \log |\mathbb{F}|)$ bits.

The coin-flipping protocol of [8] also uses the same framework of [28, 19], but substitutes the SAVSS of [3] with a "better" and more efficient SAVSS with $t < n/3$. Their SAVSS ensures that $\Theta(n)$ number of new local-conflicts are identified, if the value reconstructed by the parties is *different* from the one shared by D. The success-probability p remains $\frac{1}{4}$ and the communication complexity of the protocol is $\mathcal{O}(\text{poly}(n) \log |\mathbb{F}|)$ bits.

B.3 Vote + Coin-Flipping $\Rightarrow$ ABA

We now show how to "combine" protocols Π_{Vote} and Π_{CoinFlip} to get the protocol Π_{ABA} (see Fig 9). The current description of Π_{ABA} is taken from [16]. The protocol consists of several iterations, where each iteration consists of two instances of Π_{Vote} protocol and one instance of Π_{CoinFlip}, which are carefully "stitched" together.

In the first instance of Π_{Vote}, the parties participate with their "current input", which is initialized to their respective bits for ABA in the first iteration. Then, independent of the output received from the instance of Π_{Vote}, the parties participate in an instance of Π_{CoinFlip}. Next, the parties decide their respective inputs for the second instance of Π_{Vote} protocol, based on the output they received from the first instance. If a party has received the *highest* grade (namely 2) during the first instance of Π_{Vote}, then the party *continues* with the bit received from that Π_{Vote} instance for the second Π_{Vote} instance. Otherwise, the party *switches* to the output received from Π_{CoinFlip}. The output from the second Π_{Vote} instance is then set as the modified input for the next iteration.

If during any iteration a party obtains the highest grade from the second instance of Π_{Vote}, then it indicates this publicly by sending a `ready` message to every party, along with the bit received. The `ready` message is an indication for the others about the "readiness" of the sender party to consider the corresponding bit as the output. Finally, once a party receives this readiness indication for a common bit b from at least $2t+1$ parties, then that bit is taken as the output. To ensure that every other party also outputs the same bit, a party upon receiving the `ready` message for a common bit from at least $t+1$ honest parties, itself sends a `ready` message for the same bit (if it has not done so earlier).

The intuition behind the protocol is the following. In the protocol there can be two cases. The *first* case is when all the honest parties start with the *same* input bit, say b. Then, they will obtain the output b from all the instances of Π_{Vote} protocol in all the iterations and the outputs from Π_{CoinFlip} will be never considered. Consequently, each honest party will eventually send a `ready` message for b. Moreover, there can be at most t corrupt parties who may send a `ready` message for $1-b$ and hence no honest party ever sends a `ready` message for $1-b$. Hence, each honest party eventually outputs b.

The *second* case is when the honest parties start the protocol with *different* input bits. In this case, the protocol tries to take the help of Π_{CoinFlip} to ensure that all honest parties reach an iteration with a common input bit for that iteration. Once such an iteration is reached, this *second* case gets "transformed" to the *first* case and hence all honest parties will eventually output that common bit. In more detail, in each iteration k, it will be ensured that either every honest party have the same input bit for the second instance of Π_{Vote} with probability at least p or else certain number of new local-conflicts are identified[6]. This is because the input for second instance of Π_{Vote} is either the output bit of the first instance of Π_{Vote} or the output of Π_{CoinFlip}, both of which are *independent* of each other. Hence if the output of Π_{CoinFlip} is same for all the parties with probability p, then with probability p, this bit will be the same as output bit from the first instance of Π_{Vote}. If in any iteration k, it is *guaranteed* that all honest parties have the same inputs for the second instance of Π_{Vote}, then the parties will obtain a common output and with highest grade from the second instance of Π_{Vote}. And then from the next iteration onward, all parties will stick to that common bit and eventually output that common bit.

One can show that it requires $\mathcal{O}(\text{poly}(n))$ number of iterations in *expectation* before a "good" iteration is reached where it is *guaranteed* that all honest parties have the same input for the second instance of Π_{Vote}. Intuitively, this is because there can be $\mathcal{O}(\text{poly}(n))$ number of "bad" iterations in which the honest parties may have different outputs from the corresponding instances of Π_{CoinFlip}. This follows from the fact that the corrupt parties may deviate from the protocol instructions during the instances of Π_{CoinFlip}. There can be at most $t(n-t)$ local-conflicts which may occur (t potentially corrupt parties getting in conflict with $n-t$ honest parties) *overall* during various "failed" instances of Π_{CoinFlip} (where a failed instance means that different honest parties obtain different outputs) and only after all these local-conflicts are identified, the parties may start witnessing "clean" instances of Π_{CoinFlip} where all honest parties shun communication from all corrupt parties and where it is ensured that all honest parties obtain the same output bit with probability p. Now depending upon the number of new local-conflicts which are revealed from a single failed instance of Π_{CoinFlip}, the parties may witness $\mathcal{O}(\text{poly}(n))$ number of bad iterations[7]. Now, once all the bad iterations are over and all potential local-conflicts are identified, in each subsequent iteration, all honest parties will then have the same output from Π_{CoinFlip} (and hence, same input for the second instance of Π_{Vote}) with probability at least p. Consequently, it will take $\Theta(1)$ expected number of such iterations before the parties reach a good iteration where it is guaranteed that all honest parties have the same inputs

[6]The number of local-conflicts identified will depend upon the Π_{CoinFlip} protocol: while the Π_{CoinFlip} protocol of [3] will ensure that at least 1 new local-conflict is identified, the Π_{CoinFlip} protocol of [8] will ensure that $\Theta(n)$ number of new local-confits are identified.

[7]Since each failed instance of the Π_{CoinFlip} protocol of [3] may reveal only 1 new local-conflict, the number of bad iterations could be $\mathcal{O}(n^2)$. On the other hand, each failed instance of the Π_{CoinFlip} protocol of [8] reveals $\Theta(n)$ new local-conflicts and hence there can be $\mathcal{O}(n)$ number of bad iterations.

for the second instance of[8] Π_{Vote}.

```
Protocol $\Pi_{\mathsf{ABA}}$

Input: Party $P_i$ has the bit $b_i$ as input for the ABA protocol.
  – Initialization: Set $b = b_i$, $\mathsf{committed} = \mathsf{false}$ and $k = 1$. Then do the following.
      1. Participate in an instance of $\Pi_{\mathsf{Vote}}$ protocol with input $b$.
      2. Once an output $(b, g)$ is received from the instance of $\Pi_{\mathsf{Vote}}$, participate in an instance of $\Pi_{\mathsf{CoinFlip}}$.
           Let $\mathsf{Coin}_k$ denote the output received from $\Pi_{\mathsf{CoinFlip}}$.
      3. If $g < 2$, then set $b = \mathsf{Coin}_k$.
      4. Participate in an instance of $\Pi_{\mathsf{Vote}}$ protocol with input $b$ and let $(b, g)$ be the output received.
      5. If $g = 2$ and $\mathsf{committed} = \mathsf{false}$, then set $\mathsf{committed} = \mathsf{true}$ and send $(\mathtt{ready}, b)$ to all the parties.
      6. Set $k = k + 1$ and repeat from 1.
  – Output Computation:
      – If $(\mathtt{ready}, b)$ is received from at least $t + 1$ parties, then send $(\mathtt{ready}, b)$ to all the parties.
      – If $(\mathtt{ready}, b)$ is received from at least $2t + 1$ parties, then output $b$.
```

Figure 9: The ABA protocol from Π_{Vote} and Π_{CoinFlip}. The above code is executed by every $P_i \in \mathcal{P}$.

Lemma 3.2 now follows easily from the above discussion. Let $c \cdot \Delta$ be the time within which the protocol Π_{CoinFlip} generates output for the honest parties in a *synchronous* network, where c is some publicly-known constant. Note that c is determined by the underlying SAVSS protocol and is different for the SAVSS protocols of [3] and [8]. If all honest parties have the same input b in a *synchronous* network, then at the end of the first iteration itself, every party will send a `ready` message for b to every other party. Consequently, in this case, all honest parties will obtain their output within time $(c + 18 + 1) \cdot \Delta$. This is because each instance of Π_{Vote} during the first iteration will take at most 9Δ time to produce output, while the instance of Π_{CoinFlip} will take at most $c \cdot \Delta$ time. Additionally, Δ time will be taken by each party to send a `ready` message for b to every other party. Consequently, T_{ABA} will be $(c + 19) \cdot \Delta$.

Lemma 3.2 [3, 8]. *Let $t < n/3$. Then there exists a randomized protocol Π_{ABA} such that:*

- *Asynchronous Network: The protocol is a t-perfectly-secure ABA protocol. If the inputs of all honest parties are same, then Π_{ABA} achieves guaranteed liveness, else it achieves almost-surely liveness.*
- *Synchronous Network:*
  - *The protocol achieves validity and consistency.*
  - *If the inputs of all honest parties are same, then Π_{ABA} achieves guaranteed liveness and all honest parties obtain their output within time $T_{\mathsf{ABA}} = k \cdot \Delta$ for some constant k. Else, it achieves almost-surely liveness and requires $\mathcal{O}(poly(n) \cdot \Delta)$ expected time to generate the output.*
- *Communication Complexity: Π_{ABA} incurs a communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits if the inputs of all honest parties are the same. Else, it incurs an expected communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits.*

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

We start with the proof of the properties of the protocol Π_{BA}.

[8]One can show that if one sets $p = \frac{1}{4}$ as done in [20, 3, 8], then it takes expected 16 iterations *after* all the local-conflicts are identified to reach a good iteration.

Theorem 3.3. *Let $t < n/3$. Let Π_{SBA} be a t-perfectly-secure SBA protocol, with guaranteed liveness and weak validity in asynchronous network, such that all honest parties obtain an output within local time T_{SBA}. Moreover, let Π_{ABA} be a randomized protocol satisfying the conditions as per Lemma 3.2. Then Π_{BA} achieves the following.*

- *The protocol is a t-perfectly-secure SBA protocol and all honest parties obtain an output within time $T_{\mathsf{BA}} = T_{\mathsf{SBA}} + T_{\mathsf{ABA}}$. The protocol incurs a communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits.*
- *The protocol is a t-perfectly-secure ABA protocol with an expected communication of $\mathcal{O}(poly(n) \log |\mathbb{F}|)$ bits.*

Proof. Consider a *synchronous* network with t corruptions. The *consistency* and *guaranteed liveness* properties of Π_{SBA} in the *synchronous* network ensure that all honest parties have the same output in Π_{SBA} by the end of time T_{SBA} and hence, all honest parties participate with a *common* input in the protocol Π_{ABA}. Hence, all honest parties obtain an output by the end of time $T_{\mathsf{SBA}} + T_{\mathsf{ABA}}$, thus ensuring *guaranteed liveness* of Π_{BA}. Moreover, the *consistency* property of Π_{ABA} in the *synchronous* network guarantees that all honest parties have a *common* output, thus proving the *consistency* of Π_{BA}. Furthermore, if all honest parties have the same input bit b, then the *validity* property of Π_{SBA} in the *synchronous* network ensures that all honest parties participate with the input b in Π_{ABA} and hence output b at the end of Π_{ABA}, which follows from the *validity* of Π_{ABA} in the *synchronous* network. This proves the *validity* of Π_{BA}. The communication complexity follows from the fact that it is the sum of the communication complexity of Π_{SBA} and Π_{ABA}.

Next consider an *asynchronous* network with t corruptions. The *consistency* of the protocol Π_{BA} is inherited from the *consistency* of the underlying Π_{ABA} in the *asynchronous* network. The almost-surely liveness of Π_{BA} follows from the *almost-surely liveness* of Π_{ABA} and the fact that every honest party is guaranteed to obtain some output during Π_{SBA} within (local) time T_{SBA}. For the *validity* of Π_{BA}, consider the case when all honest parties start the protocol Π_{BA} with a common input bit b. Since the protocol Π_{SBA} has *weak validity* and produces some output for every honest party within (local) time T_{SBA}, it follows that $v_i \in \{b, \perp\}$ for every honest party P_i and hence, each honest party participates with a *common* input $v_i^\star = b$ in Π_{ABA}. The *validity* of Π_{BA} now inherits from the *validity* of Π_{ABA} in the *asynchronous* network. The communication complexity follows from the fact that the instance of Π_{ABA} will incur an expected communication of $\mathcal{O}(\text{poly}(n) \log |\mathbb{F}|)$ bits. □

C.1 Properties of the Protocol Π_{SBA}

In this section, we prove the properties of the protocol Π_{SBA}.

Theorem 3.5. *Let $t < n/3$. Let Π_{BC} be a t-perfectly-secure broadcast protocol in the synchronous network, with weak validity, weak consistency and liveness in asynchronous network, where honest parties obtain output within (local) time T_{BC}. Then, Π_{SBA} achieves the following with its communication complexity being n times that of Π_{BC}.*

- *Π_{SBA} is a t-perfectly-secure SBA protocol where honest parties obtain an output ($\neq \perp$) within time $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$.*
- *The protocol has guaranteed liveness and weak validity in the asynchronous network, such that all honest parties obtain an output within (local) time $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$.*

Proof. The *guaranteed liveness*, both in the synchronous and asynchronous network trivially follows from the *liveness* of Π_{BC}, 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. Hence, $T_{\mathsf{SBA}} = T_{\mathsf{BC}}$. We next prove the rest of the properties in a *synchronous* network.

In a *synchronous* network, at least $n-t > 2t$ instances of Π_{BC} corresponding to the honest senders, result in an output different from $\bot$ for all honest parties (follows from the *validity* property of Π_{BC} in the synchronous network). Moreover, from the *consistency* property of Π_{BC} in the synchronous network, all honest parties obtain the same output in all the n instances of Π_{BC} instances. Hence, all honest parties output the same value, different from $\bot$ (obtained through the majority rule), proving *consistency* of Π_{SBA}. Finally, if all honest parties have the same input bit b, then there will be at least $n-t$ instances of Π_{BC} with output b and at most t instances with output $\bar{b} = 1-b$. By applying the majority rule, all honest parties output b, proving *validity* of Π_{SBA}.

Next, consider an *asynchronous* network and suppose all honest parties have the same input bit b. Let P_i be an arbitrary *honest* party, who obtains an output, different from $\bot$. This implies that it obtained $n-t$ Boolean output values $b_i^{(j)}$, of which at most t could be $\bar{b}$ (corresponding to t corrupt parties) and hence majority of these Boolean values will be b. Thus P_i outputs b, proving the weak validity in the asynchronous network.

The communication complexity simply follows from the fact that n instances of Π_{BC} are invoked in the protocol. □

C.2 Properties of the Protocol Π_{BC}

In this section, we prove the properties of the protocol Π_{BC}.

Theorem 3.6. *Let Π_{BGP} be a t-perfectly-secure SBA protocol with guaranteed liveness in asynchronous network and a communication complexity of $\mathcal{O}(n^2\ell)$ bits, where all honest parties have an output at time $T_{\mathsf{BGP}} = (12n-6)\cdot\Delta$. Then, Π_{BC} achieves the following for any $t < n/3$, with a communication complexity of $\mathcal{O}(n^2\ell)$ bits.*

- *Synchronous network:* **(a) Liveness***: at time $T_{\mathsf{BC}} = (12n-3)\cdot\Delta$, each honest P_h has an output.* **(b) Validity***: If S is honest, then at time T_{BC}, each honest P_h has the output m.* **(c) Consistency***: If S is corrupt, then the output of every honest party is the same at time T_{BC}.* **(d) Fallback Consistency***: If S is corrupt and some honest P_h outputs $m^\star \neq \bot$ at time T through fallback-mode (hence $T > T_{\mathsf{BC}}$), then every honest party outputs $m^\star$ by time $T + 2\Delta$.*
- *Asynchronous Network:* **(a) Liveness***: same as above.* **(b) Weak Validity***: If S is honest, then at local time T_{BC}, each honest P_h has output m or $\bot$.* **(c) Fallback Validity***: If S is honest, then all honest P_h with output $\bot$ at local time T_{BC}, eventually output m through fallback-mode.* **(d) Weak Consistency***: If S is corrupt, then at local time T_{BC}, each honest P_h has output $m^\star \neq \bot$ or $\bot$.* **(e) Fallback Consistency***: If S is corrupt and some honest party has output $m^\star \neq \bot$ at local time T where $T \geq T_{\mathsf{BC}}$, then every honest party eventually outputs $m^\star$.*

Proof. The *liveness* (both for the *synchronous* as well *asynchronous* network) simply follows from the fact that every honest party outputs something (including $\bot$) at (local) time $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{BGP}}$, where $T_{\mathsf{BGP}} = (12n-6)\cdot\Delta$. We next prove the rest of the properties of the protocol in the *synchronous* network, for which we rely on the properties of Acast and Π_{BGP} in the *synchronous* network.

If S is *honest*, then due to the *liveness* and *validity* properties of Π_{ACast} in the *synchronous* network, at time 3Δ, every *honest* party P_i receives m from the Acast of S. Hence, every honest party participates with input m in the instance of Π_{BGP}. From the *guaranteed liveness* and *validity* of Π_{BGP} in *synchronous* network, at time $3\Delta + T_{\mathsf{BGP}}$, every honest party will have m as the output from Π_{BGP}. Hence, each honest party has the output m at time T_{BC}, thus proving that *validity* is achieved.

For *consistency*, we consider a *corrupt* S. We first note that each honest party will have the *same* output from the instance of Π_{BGP} at time T_{BC}, which follows from the *consistency* property of Π_{BGP} in *synchronous* network. If all honest honest parties have the output $\bot$ for Π_{BC} at time T_{BC}, then consistency holds trivially. So consider the case when some *honest* party, say P_i, has the output $m^\star \neq \bot$ for Π_{BC} at time T_{BC}. This implies that the output of Π_{BGP} is $m^\star$ for every honest party. Moreover, it also implies that at time 3Δ, at least one *honest* party, say P_h, has received $m^\star$ from the Acast of S. Otherwise, all honest parties would participate with input $\bot$ in the instance of Π_{BGP} and from the *validity* of Π_{BGP} in the *synchronous* network, every honest party would compute $\bot$ as the output during Π_{BGP}, which is a contradiction. Since P_h has $m^\star$ from S's Acast at time 3Δ, it follows from the *consistency* property of Π_{ACast} in the *synchronous* network that *all* honest parties will have $m^\star$ from S's Acast by time 5Δ. Moreover, $5\Delta < (12n - 3) \cdot \Delta$ holds. Consequently, at time $(12n - 3) \cdot \Delta$, *all* honest parties will have $m^\star$ from S's Acast and $m^\star$ as the output of Π_{BGP} and hence, output $m^\star$ for Π_{BC}.

For *fallback consistency*, we have to consider a *corrupt* S. Let P_h be an *honest* party who outputs $m^\star \neq \bot$ at time T through fallback-mode, where $T > T_{\mathsf{BC}}$. We first note that this implies that *every* honest party has output $\bot$ at time T_{BC}. This is because, from the proof of the *consistency* property of Π_{BC}, if any *honest* party has an output $m' \neq \bot$ at time T_{BC}, then *all* honest parties (including P_h) also must have computed the output m' at time T_{BC}. Hence, P_h will never change its output to[9] $m^\star$. Since P_h has obtained the output $m^\star$, it means that at time T, it has received $m^\star$ from the Acast of S. It then follows from the *consistency* of Π_{ACast} in the *synchronous* network that every honest party will also receive $m^\star$ from the Acast of S, latest by time $T + 2\Delta$ and output $m^\star$.

We next prove the properties of the protocol Π_{BC} in an *asynchronous* network, for which we depend upon the properties of Π_{ACast} in the *asynchronous* network. The *weak-validity* property follows from the *validity* property of Π_{ACast} in the *asynchronous* network, which ensures that no honest party P_i ever receives an m' from the Acast of S where $m' \neq m$. So if at all P_i outputs a value different from $\bot$ at time T_{BC}, it has to be m. The *weak-consistency* property follows using similar arguments as used to prove *consistency* in the *synchronous* network, but relying instead on the *validity* and *consistency* properties of Π_{ACast} in the asynchronous network. The latter property ensures that even if the adversary has full control over message scheduling in the *asynchronous* network, it cannot ensure that for a *corrupt* S, two different honest parties end up receiving m_1 and m_2 from the Acast of S, where $m_1 \neq m_2$.

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

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

The *communication complexity* follows from the communication complexity of Π_{BGP} and Π_{ACast}. □

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

D Properties of the VSS Protocol

D.1 Proof of the Properties of the Protocol Π_{WPS}

We first present the modified protocol Π_{WPS}, where D has L number of t_s-degree polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ as input.

Protocol $\Pi_{\mathsf{WPS}}(\mathsf{D}, \{q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)\})$

- **Phase I (Time Δ) — Sending Polynomials:**
  - D on having the input $q^{(1)}(\cdot), ..., q^{(L)}(\cdot)$ chooses random (t_s, t_s)-degree symmetric bivariate polynomials $Q^{(1)}(x, y), \ldots, Q^{(L)}(x, y)$ such that $Q^{(\ell)}(0, y) = q^{(\ell)}(\cdot)$ holds for $\ell = 1, \ldots, L$ and sends $q_i^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_i)$ to each party $P_i \in \mathcal{P}$.
- **Phase II (Time Δ) — Exchanging Common Values**: Each P_i upon receiving t_s-degree polynomials $\{q_i^{(\ell)}(x)\}_{\ell=1,\ldots,L}$ from D, does the following.
  - Wait till the local time becomes a multiple of Δ and then for $\ell = 1, \ldots, L$, send $q_{ij}^{(\ell)} = q_i^{(\ell)}(\alpha_j)$ to each $P_j \in \mathcal{P}$.
- **Phase III (Time T_{BC}) — Publicly Declaring the Results of Pair-Wise Consistency Test**: Each $P_i \in \mathcal{P}$ does the following.
  - Upon receiving $\{q_{ji}^{(\ell)}\}_{\ell=1,\ldots,L}$ from P_j, wait till the local time becomes a multiple of Δ. Then broadcast $\mathtt{OK}(i, j)$ if $q_{ji}^{(\ell)} = q_i^{(\ell)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$. Else, set $\mathsf{Conflict}_{ij} = \{\ell | q_{ji}^{(\ell)} \neq q_i^{(\ell)}(\alpha_j)\}$ and broadcast $\mathtt{NOK}(\ell_{\mathsf{min}}, i, j, q_i^{(\ell_{\mathsf{min}})}(\alpha_j))$ where ℓ_{min} is the minimum value in $\mathsf{Conflict}_{ij}$.
- **Local computation — Constructing Consistency Graph:** Each $P_i \in \mathcal{P}$ does the following.
  - Construct a *consistency graph* G_i over $\mathcal{P}$, where the edge (P_j, P_k) is included in G_i, if $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ is received from the broadcast of P_j and P_k respectively, either through the regular-mode or fall-back mode.
- **Phase IV (Time T_{BC}) — Checking for $(n, t_s)-$star**: D does the following in its consistency graph G_{D} at time $2\Delta + T_{\mathsf{BC}}$.
  - Remove edges incident with P_i from G_{D}, if $\mathtt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ is received from the broadcast of P_i through regular-mode such that $q_{ij}^{(\ell)} \neq Q^{(\ell)}(\alpha_j, \alpha_i)$.
  - Compute $\mathcal{W} = \{P_i : \mathsf{deg}(P_i) \geq n - t_s\}$, where $\mathsf{deg}(P_i)$ denotes the degree of P_i in G_{D}. Remove party P_i from $\mathcal{W}$ if in G_{D}, party P_i is *not* incident with at least $n - t_s$ parties in $\mathcal{W}$. Repeat this step till no more parties can be removed from $\mathcal{W}$.
  - Run algorithm $\mathsf{AlgStar}$ on $G_{\mathsf{D}}[\mathcal{W}]$, where $G_{\mathsf{D}}[\mathcal{W}]$ denotes the subgraph of G_{D} induced by the vertices in $\mathcal{W}$. If an $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$ is obtained, then broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$.
- **Local Computation — Verifying and Accepting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$:** Each $P_i \in \mathcal{P}$ does the following at time $2\Delta + 2T_{\mathsf{BC}}$.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is received from the broadcast of D through regular-mode, then *accept* it if following *were true* at time $2\Delta + T_{\mathsf{BC}}$:
    - $\not\exists P_j, P_k \in \mathcal{W}, \not\exists l \in \{1, \ldots, L\}$, such that $\mathtt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\mathtt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode, where $q_{jk}^{(\ell)} \neq q_{kj}^{(\ell)}$.
    - In G_i, $\mathsf{deg}(P_j) \geq n - t_s$ for all $P_j \in \mathcal{W}$ and P_j *had* edges with at least $n - t_s$ parties from $G_i[\mathcal{W}]$.
    - $(\mathcal{E}, \mathcal{F})$ *was* an $(n, t_s)-$star in $G_i[\mathcal{W}]$ and $\forall P_j, P_k \in \mathcal{W}$ where the edge (P_j, P_k) *was* present in G_i, the $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode.
- **Phase V (Time T_{BA}) — Deciding Whether to Go for $(n, t_a)-$star**: At time $2\Delta + 2T_{\mathsf{BC}}$, each $P_i \in \mathcal{P}$ participates in an instance of Π_{BA} with input $b_i = 0$ if a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ was accepted, else with input $b_i = 1$ and waits for time T_{BA}.
- **Local Computation — Computing WPS-shares Through $\mathcal{W}$**: If the output of Π_{BA} is 0, then each $P_i \in \mathcal{P}$ computes *WPS-Shares* $s_i^{(1)}, \ldots, s_i^{(\ell)}$ (initially all set to $\perp$) as follows.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is not yet received then wait till $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is received from D's broadcast through

fall-back mode.
  - If $P_i \in \mathcal{W}$, then output $\{s_i^{(\ell)} = q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.
  - **(a)** Else, initialise L support sets $\{\mathcal{SS}_i^{(\ell)}\}_{\ell=1,\ldots,L}$ to $\emptyset$. Upon receiving $\{q_{ji}^{(\ell)}\}_{\ell=1,\ldots,L}$ from $P_j \in \mathcal{F}$, include $q_{ji}^{(\ell)}$ to $\mathcal{SS}_i^{(\ell)}$. For $\ell = 1,\ldots,L$, keeps executing $\mathsf{OEC}(t_s, t_s, \mathcal{SS}_i^{(\ell)})$, till a t_s-degree polynomial $q_i^{(\ell)}(\cdot)$ is obtained. Then, output $\{s_i^{(\ell)} = q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.
- **Phase VI (Time T_{BC}) — Broadcasting (n, t_a)−star**: If the output of Π_{BA} is 1, then D does the following.
  - After every update in G_{D}, run $\mathsf{AlgStar}$ on G_{D}. If an (n, t_a)−star $(\mathcal{E}', \mathcal{F}')$ is obtained, then broadcast $(\mathcal{E}', \mathcal{F}')$.
- **Local Computation — Computing WPS-shares Through (n, t_a)−star**: If the output of Π_{BA} is 1, then each P_i does the following.
  - Wait till an (n, t_a)−star $(\mathcal{E}', \mathcal{F}')$ is obtained from the broadcast of D, either through regular or fall-back mode. Upon receiving, wait till $(\mathcal{E}', \mathcal{F}')$ becomes an (n, t_a)−star in G_i.
  - If $P_i \in \mathcal{F}'$, then output $\{s_i^{(\ell)} = q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$. Else, compute $\{q_i^{(\ell)}(x)\}_{\ell=1,\ldots,L}$ using similar procedure as in step **(a)** above on the points received from the parties in $\mathcal{F}'$ and output $\{s_i^{(\ell)} = q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.

Figure 10: The best-of-both-worlds weak polynomial-sharing protocol for L polynomials.

Lemma D.1 (Privacy). *In protocol Π_{WPS}, if D is honest, then irrespective of the network type, the view of the adversary remains independent of $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$.*

Proof. Let D be *honest*. We consider the worst case scenario when the adversary controls up to t_s parties. We claim that throughout the protocol, the adversary learns at most t_s univariate polynomials lying on $Q^{(\ell)}(x, y)$, for $\ell = 1, \ldots, L$. Since $Q^{(\ell)}(x, y)$ is a random (t_s, t_s)-degree- symmetric-bivariate polynomial, it then follows from Lemma 2.1 that the view of the adversary will be independent of $q^{(\ell)}(\cdot)$. We next proceed to prove the claim.

Corresponding to every *corrupt* P_i, the adversary learns $Q^{(\ell)}(x, \alpha_i)$. Corresponding to every *honest* P_i, the adversary learns t_s distinct points on P_i's univariate polynomial $Q^{(\ell)}(x, \alpha_i)$ through the pair-wise consistency checks. However, these points were already included in the view of the adversary (through the univariate polynomials under adversary's control). Hence no additional information about the polynomials of the honest parties is revealed during the pair-wise consistency checks. Furthermore, no *honest* P_i broadcasts $\mathtt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ corresponding to any *honest* P_j for any $\ell = 1, \ldots, L$, since the pair-wise consistency check will always pass for every pair of *honest* parties. □

Lemma D.2 (Correctness in Synchronous Network). *In protocol Π_{WPS}, if D is honest and the network is synchronous, then each honest P_i outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ at time $T_{\mathsf{WPS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.*

Proof. Let D be *honest* and the network be *synchronous* with up to t_s corruptions. During phase I, every (honest) party P_j receives $q_j^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_j)$ from D within time Δ, for $\ell = 1, \ldots, L$. Hence during phase II, every *honest* P_j sends $\{q_{jk}^{(\ell)}\}_{\ell=1,\ldots,L}$ to every P_k, which takes at most Δ time to be delivered. Hence, by time 2Δ, every *honest* P_j receives $\{q_{jk}^{(\ell)}\}_{\ell=1,\ldots,L}$ from every *honest* P_k, such that $q_{kj}^{(\ell)} = q_j^{(\ell)}(\alpha_k)$ holds. Consequently, during phase III, every *honest* P_j broadcasts $\mathtt{OK}(j, k)$ corresponding to every *honest* P_k, and vice versa. From the *validity* property of Π_{BC} in the *synchronous* network, it follows that every honest P_i receives $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ from the broadcast of every *honest* P_j and every *honest* P_k through regular-mode, at time $2\Delta + T_{\mathsf{BC}}$. Hence, the edge (P_j, P_k) will be added to the graph G_i, corresponding to every honest P_j, P_k. Furthermore, from the *consistency* property of Π_{BC}, the graph G_i will be the same for every honest party P_i (including D) at time $2\Delta + T_{\mathsf{BC}}$. Moreover, if D receives an *incorrect* $\mathtt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ message from the broadcast of any *corrupt* P_j through regular-mode at time $2\Delta + T_{\mathsf{BC}}$ where $q_{ij}^{(\ell)} \neq Q^{(\ell)}(\alpha_j, \alpha_i)$, then D removes all the edges incident with P_i in D's consistency graph G_{D}. D then computes the set $\mathcal{W}$, and all *honest* parties will be present in $\mathcal{W}$. Moreover, the honest parties will form a clique of size at least $n - t_s$ in the subgraph $G_{\mathsf{D}}[\mathcal{W}]$ at time $2\Delta + T_{\mathsf{BC}}$ and D will find an $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$ in $G_{\mathsf{D}}[\mathcal{W}]$ and broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ during phase IV. By the *validity* of Π_{BC} in *synchronous* network, all honest parties will receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$. Moreover, all honest parties will *accept* $(\mathcal{E}, \mathcal{F})$ and participate with input 0 in the instance of Π_{BA}. Hence, by the *validity* and *guaranteed liveness* of Π_{BA} in the *synchronous* network, *every* honest party obtains the output 0 in Π_{BA} by time $2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Now, consider an arbitrary *honest* party P_i. Since $P_i \in \mathcal{W}$, party P_i outputs $s_i^{(\ell)} = q_i^{(\ell)}(0) = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. □

Lemma D.3 (Correctness in Asynchronous Network). *In protocol* Π_{WPS}*, if* D *is honest and network is asynchronous, then almost-surely, each honest* P_i *eventually outputs* $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.

Proof. Let D be *honest* and network be *asynchronous* with up to t_a corruptions. We first note that every honest party participates with some input in the instance of Π_{BA} at local time $2\Delta + 2T_{\mathsf{BC}}$. Hence almost-surely, the instance of Π_{BA} eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of Π_{BA} is 0**: This means that at least one *honest* party, say P_h, has accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ at local $2\Delta + 2T_{\mathsf{BC}}$, received from the broadcast of D. Hence, by the *weak validity* and *fallback validity* of Π_{BC} in the *asynchronous* network, all honest parties will eventually receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D and *accept* this. This is because by the *fallback consistency* of Π_{BC}, the consistency graphs of all honest parties will eventually have all the edges which were present in G_h at time $2\Delta + 2T_{\mathsf{BC}}$. We claim that *every honest* P_i will eventually get $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This implies that eventually every *honest* P_i outputs $s_i^{(\ell)} = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{W}$: In this case, P_i already has received $Q^{(\ell)}(x, \alpha_i)$ from D.
  - $P_i \notin \mathcal{W}$: In this case, there will be at least $n - t_s > 2t_s + t_a$ parties in $\mathcal{F}$, of which at most t_a could be *corrupt*. Since $Q^{(\ell)}(x, \alpha_i)$ is a t_s-degree polynomial and $t_s < |\mathcal{F}| - 2t_a$, from Lemma A.1 it follows that by applying the OEC procedure on the common points on the polynomials $Q^{(\ell)}(x, \alpha_i)$ received from the parties in $\mathcal{F}$, party P_i will eventually obtain $Q^{(\ell)}(x, \alpha_i)$.
- **The output of Π_{BA} is 1**: Since D is *honest*, every pair of honest parties P_j, P_k eventually broadcast $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ messages, as the pair-wise consistency check between them will eventually be successful. From the *weak validity* and *fallback validity* of Π_{BC}, these messages are eventually delivered to every honest party. Also from the *weak consistency* and *fallback consistency* of Π_{BC}, any $\mathtt{OK}$ message which is received by D from the broadcast of any *corrupt* party, will eventually be received by every other honest party as well. As there will be at least $n - t_a$ honest parties, a clique of size at least $n - t_a$ will eventually form in the consistency graph of every honest party. Hence D will eventually find an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ in its consistency graph and broadcast it. From the *weak validity* and *fallback validity* of Π_{BC}, this star will be eventually delivered to every honest party. Moreover, $(\mathcal{E}', \mathcal{F}')$ will eventually be an $(n, t_a)-$star in every honest party's consistency graph by the *fallback consistency* of Π_{BC}. We claim that *every honest* P_i will eventually get $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This implies that eventually, every *honest* P_i outputs $s_i^{(\ell)} = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim,

consider an arbitrary *honest* party P_i. There are two possible cases.

- $P_i \in \mathcal{F}'$: In this case, P_i already has $Q^{(\ell)}(x, \alpha_i)$, received from D.
- $P_i \notin \mathcal{F}'$: In this case, there will be at least $n - t_a > 3t_s$ parties in $\mathcal{F}'$, of which at most t_a could be *corrupt*, where $t_a < t_s$. Since $Q^{(\ell)}(x, \alpha_i)$ is a t_s-degree polynomial and $t_s < |\mathcal{F}'| - 2t_a$, from Lemma A.1 it follows that by applying the OEC procedure on the common points on the polynomial $Q^{(\ell)}(x, \alpha_i)$ received from the parties in $\mathcal{F}'$, party P_i will eventually obtain $Q^{(\ell)}(x, \alpha_i)$.

□

Before we proceed to prove the *weak commitment* property in the *synchronous* network, we prove a helping lemma.

Lemma D.4. *In protocol* Π_{WPS} *if the network is synchronous and* D *is corrupt and if at least one honest party has received* $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ *from the broadcast of* D *through regular-mode and accepted it at time* $2\Delta + 2T_{\mathsf{BC}}$*, then all the following hold:*

- *All the honest parties in* $\mathcal{W}$ *have received their respective* t_s*-degree univariate polynomials by time* Δ*.*
- *For* $\ell = 1, \ldots, L$*, the univariate polynomials* $q_i^{(\ell)}(x)$ *of all honest parties* P_i *lie on a unique* (t_s, t_s)*-degree symmetric bivariate polynomial, say* ${Q^{\star}}^{(\ell)}(x, y)$*.*
- *At time* $2\Delta + 2T_{\mathsf{BC}}$*, every honest party accepts* $(\mathcal{W}, \mathcal{E}, \mathcal{F})$*.*

Proof. Let D be *corrupt* and network be *synchronous* with up to t_s corruptions. As per the lemma condition, let P_h be an *honest* party, who has received $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode and accepted it at time $2\Delta + 2T_{\mathsf{BC}}$. From the protocol steps, the following must be true for P_h at time $2\Delta + T_{\mathsf{BC}}$:

- $\nexists P_j, P_k \in \mathcal{W}$, $\nexists \ell \in \{1, \ldots, L\}$, such that $\mathtt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\mathtt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages had been received from the broadcast of P_j and P_k respectively through regular-mode, where $q_{jk}^{(\ell)} \neq q_{kj}^{(\ell)}$.
- In P_h's consistency graph G_h, $\mathsf{deg}(P_j) \geq n - t_s$ for all $P_j \in \mathcal{W}$ and P_j has edges with at least $n - t_s$ parties from $\mathcal{W}$.
- $(\mathcal{E}, \mathcal{F})$ *was* an (n, t_s)−star in the induced subgraph $G_h[\mathcal{W}]$, such that $\forall P_j, P_k \in \mathcal{W}$ where the edge (P_j, P_k) is present in G_h, messages $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ *were* received from the broadcast of P_j and P_k respectively through regular-mode.

We prove the first part of the lemma through contradiction. So let $P_j \in \mathcal{W}$ be an *honest* party, who received its t_s-degree univariate polynomials $\{q_j^{\ell}(x)\}_{\ell=1,\ldots,L}$ at time $\Delta + \delta$, where $\delta > 0$. Moreover, let $P_k \in \mathcal{W}$ be an *honest* party different from P_j (note that there are at least $n - 2t_s$ *honest* parties in $\mathcal{W}$). As stated above, at time $2\Delta + T_{\mathsf{BC}}$, party P_h has received the message $\mathtt{OK}(k, j)$ from the broadcast of P_k through regular-mode. From the protocol steps, P_j waits till its local time becomes a multiple of Δ, before it sends the points on its polynomial to other parties. Hence, P_j must have started sending the points after time $c \cdot \Delta$, where $c \geq 2$. Since the network is *synchronous*, the points $\{q_{jk}^{(\ell)} = q_j^{(\ell)}(\alpha_k)\}_{\ell=1,\ldots,L}$ must have been received by P_k by time $(c+1) \cdot \Delta$. Moreover, from the protocol steps, even if P_k receives these points at time T, where $c \cdot \Delta < T < (c+1) \cdot \Delta$, it waits till time $(c+1) \cdot \Delta$, before broadcasting the $\mathtt{OK}(k, j)$ message. Since P_k is *honest*, from the *validity* property of Π_{BC} in *synchronous* network, it will take *exactly* T_{BC} time for the message $\mathtt{OK}(k, j)$ to be received through regular-mode, once it is broadcasted. This implies that P_h will receive the message $\mathtt{OK}(k, j)$ at time $(c+1) \cdot \Delta + T_{\mathsf{BC}}$, where $(c+1) > 2$. However, this is a contradiction, since the $\mathtt{OK}(k, j)$ message has been received by P_h at time $2\Delta + T_{\mathsf{BC}}$.

To prove the second part of the lemma, we will show that for $\ell = 1, \ldots, L$, the univariate polynomials $q_j^{(\ell)}(x)$ of all the *honest* parties in $\mathcal{W}$ are pair-wise consistent. Since there are at least $n - 2t_s > t_s$ *honest* parties in $\mathcal{W}$, from Lemma 2.1 it follows that the univariate polynomials $q_j^{(\ell)}(x)$

of all the *honest* parties in $\mathcal{W}$ lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$. Consider an arbitrary pair of *honest* parties $P_j, P_k \in \mathcal{W}$. From the first part of the lemma, both P_j and P_k must have received their respective univariate polynomials $q_j^{(\ell)}(x)$ and $q_k^{(\ell)}(x)$ by time Δ. This further implies that P_j and P_k must have received the points $q_{kj}^{(\ell)} = q_k^{(\ell)}(\alpha_j)$ and $q_{jk}^{(\ell)} = q_j^{(\ell)}(\alpha_k)$ respectively by time 2Δ. Since P_j and P_k are honest and the network is synchronous, they will agree on the set of conflicts between them and thus, $\mathsf{Conflict}_{jk} = \mathsf{Conflict}_{kj}$. If $\mathsf{Conflict}_{jk} \neq \emptyset$, then P_j and P_k would broadcast $\mathtt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\mathtt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages respectively at time 2Δ, where ℓ is the minimum value in $\mathsf{Conflict}_{jk}$ *and* $\mathsf{Conflict}_{kj}$. From the *validity* property of Π_{BC} in the *synchronous* network, P_h will receive these messages through regular-mode at time $2\Delta + T_{\mathsf{BC}}$. Consequently, P_h *will not* accept $(\mathcal{W}, \mathcal{E}, \mathcal{F})$, which is a contradiction.

To prove the third part of the lemma, we note that since P_h has received $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$, it implies that D must have started broadcasting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ latest at time $2\Delta + T_{\mathsf{BC}}$, since it takes T_{BC} time for the regular-mode of Π_{BC} to produce an output. From the *consistency* property of Π_{BC} in *synchronous* network, it follows that every honest party will also receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode at time $2\Delta + 2T_{\mathsf{BC}}$. Since at time $2\Delta + T_{\mathsf{BC}}$, party P_h has received the $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ messages through regular-mode from the broadcast of every $P_j, P_k \in \mathcal{W}$ where (P_j, P_k) is an edge in P_h's consistency graph, it follows that these messages started getting broadcasted latest at time 2Δ. From the *validity* and *consistency* properties of Π_{BC} in the *synchronous* network, it follows that every honest party receives these broadcast messages through regular-mode at time $2\Delta + T_{\mathsf{BC}}$. Hence $(\mathcal{E}, \mathcal{F})$ will constitute an $(n, t_s)-$star in the induced subgraph $G_i[\mathcal{W}]$ of every honest party P_i's consistency-graph at time $2\Delta + T_{\mathsf{BC}}$ and consequently, every honest party accepts $(\mathcal{W}, \mathcal{E}, \mathcal{F})$. □

Lemma D.5 (Weak Commitment in Synchronous Network). *In protocol* Π_{WPS}*, if* D *is corrupt and network is synchronous, then either no honest party obtains any output or there exist* t_s*-degree polynomials, say* $\{q^{\star^{(\ell)}}(\cdot)\}_{\ell=1,\ldots,L}$*, such that all the following hold.*

- *There are at least* $t_s + 1$ *honest parties* P_i *who output* $\{q^{\star^{(\ell)}}(\alpha_i)\}_{\ell=1,\ldots,L}$.
- *If any honest* P_j *outputs* $s_j^{(1)}, \ldots, s_j^{(\ell)} \in \mathbb{F}$*, then* $s_j^{(\ell)} = q^{\star^{(\ell)}}(\alpha_j)$ *holds for* $\ell = 1, \ldots, L$.

Proof. Let D be *corrupt* and network be *synchronous* with up to t_s corruptions. If no honest party obtains any output, then the lemma holds trivially. So consider the case when some honest party obtains an output, consisting of ℓ elements of $\mathbb{F}$. Now, there are two possible cases.

- **At time $2\Delta + 2T_{\mathsf{BC}}$, at least one honest party P_h has accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$, received from the broadcast of D through regular-mode**: In this case, from Lemma D.4, at time $2\Delta + 2T_{\mathsf{BC}}$, every honest party will accept $(\mathcal{W}, \mathcal{E}, \mathcal{F})$. Hence every honest party participates in Π_{BA} with input 0. From the *validity* and *guaranteed liveness* properties of Π_{BA} in the *synchronous* network, all honest parties will get the output 0 during Π_{BA} by time $T_{\mathsf{WPS}} = 2\Delta + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. From Lemma D.4, the univariate polynomials of all the *honest* parties in $\mathcal{W}$ define L number of (t_s, t_s)-degree symmetric bivariate polynomials, say $\{Q^{\star^{(\ell)}}(x, y)\}_{\ell=1,\ldots,L}$. Let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. Now consider an arbitrary *honest* party P_i, who outputs $s_i^{(1)}, \ldots, s_i^{(\ell)} \in \mathbb{F}$. We want to show that for $\ell = 1, \ldots, L$, the condition $s_i^{(\ell)} = q^{\star^{(\ell)}}(\alpha_i)$ holds and there are at least $t_s + 1$ such *honest* parties P_i who output $s_i^{(1)}, \ldots, s_i^{(\ell)} \in \mathbb{F}$. There are two possible cases.
  - $P_i \in \mathcal{W}$: From the protocol steps, P_i sets $s_i^{(\ell)} = Q^{\star^{(\ell)}}(0, \alpha_i)$ which is the same as $q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$. Since $\mathcal{W}$ contains at least $t_s + t_a + 1$ honest parties, this also shows that at least $t_s + 1$ honest parties P_i output $s_i^{(1)}, \ldots, s_i^{(\ell)} \in \mathbb{F}$.
  - $P_i \notin \mathcal{W}$: In this case, P_i sets $s_i^{(\ell)} = q_i^{(\ell)}(0)$, where $q_i^{(\ell)}(\cdot)$ is a t_s-degree univariate polynomial obtained by applying the OEC procedure with $d = t = t_s$ on the points $q_{ji}^{(\ell)}$ received from

the parties in $\mathcal{F}$. Note that as part of OEC (see Lemma A.1), party P_i verifies that $q_i^{(\ell)}(\cdot)$ is consistent with least $2t_s+1$ $q_{ji}^{(\ell)}$ values received from the parties in $\mathcal{F}$. Now out of those consistent $q_{ji}^{(\ell)}$ values, at least t_s+1 values are from the *honest* parties in $\mathcal{F}$. Furthermore, these $q_{ji}^{(\ell)}$ values from the *honest* parties in $\mathcal{F}$ are the same as $Q^{\star^{(\ell)}}(\alpha_i, \alpha_j)$, which is equal to $Q^{\star^{(\ell)}}(\alpha_j, \alpha_i)$ and uniquely determine $Q^{\star^{(\ell)}}(x, \alpha_i)$; the last property holds since $Q^{\star^{(\ell)}}(x, y)$ is a symmetric bivariate polynomial. This automatically implies that $q_i^{(\ell)}(x)$ is the *same* as $Q^{(\ell)}(x, \alpha_i)$ and hence $s_i^{(\ell)} = q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$.

– **At time $2T_{\mathsf{BC}} + 2\Delta$, no honest party has accepted any** $(\mathcal{W}, \mathcal{E}, \mathcal{F})$: This implies that all honest parties participate in Π_{BA} with input 1 and by the *validity* and *guaranteed liveness* of Π_{BA}, all honest parties obtain the output 1 in Π_{BA}. Let P_h be the *first honest* party who outputs ℓ values from $\mathbb{F}$. This means that P_h has received an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D, such that $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph. By the *consistency* and *fallback consistency* properties of Π_{BC} in synchronous network, *all* honest parties receive $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D. Moreover, since $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph, it will also constitute an $(n, t_a)-$star in every other honest party's consistency graph as well. This is because the $\mathtt{OK}(\star, \star)$ messages which are received by P_h from the broadcast of the various parties in $\mathcal{E}'$ and $\mathcal{F}'$ are also received by every other honest party. The last property follows from *validity, consistency* and *fallback consistency* properties of Π_{BC} in synchronous network. Since $|\mathcal{E}'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, it follows that $\mathcal{E}'$ has at least $t_s + 1$ *honest* parties P_i, whose univariate polynomials $q_i^{(\ell)}(x)$ are pair-wise consistent for $\ell = 1, \ldots, L$ and hence, from Lemma 2.1, lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$. Similarly, since the univariate polynomial $q_i^{(\ell)}(x)$ of every *honest* party in $\mathcal{F}'$ is pair-wise consistent with the univariate polynomials $q_j^{(\ell)}(x)$ of the *honest* parties in $\mathcal{E}'$, it implies that the univariate polynomials $q_i^{(\ell)}(x)$ of all the *honest* parties in $\mathcal{F}'$ also lie on $Q^{\star^{(\ell)}}(x, y)$. Let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. We show that *every honest* P_i outputs $q^{\star^{(\ell)}}(\alpha_i)$. For this it is enough to show that each honest P_i gets $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$, as P_i outputs $q_i^{(\ell)}(0)$, which will be the same as $q^{\star^{(\ell)}}(\alpha_i)$. Consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{F}'$: In this case, P_i already has $Q^{\star^{(\ell)}}(x, \alpha_i)$, received from D, for $\ell = 1, \ldots, L$.
  - $P_i \notin \mathcal{F}'$: In this case, there will be $n - t_a > 3t_s$ parties in $\mathcal{F}'$, of which at most t_s could be corrupt. Moreover, $Q^{\star^{(\ell)}}(x, \alpha_i)$ is a t_s-degree polynomial and $t_s < |\mathcal{F}'| - 2t_s$ holds. Hence from the properties of OEC (Lemma A.1), by applying the OEC procedure on the common points on the polynomial $Q^{\star^{(\ell)}}(x, \alpha_i)$ received from the parties in $\mathcal{F}'$, party P_i will compute $Q^{\star^{(\ell)}}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.

□

Lemma D.6 (**Strong Commitment in Asynchronous Network**). *In protocol* Π_{WPS}*, if* D *is corrupt and network is asynchronous, then either no honest party obtains any output or there exist* t_s*-degree polynomials, say* $\{q^{\star^{(\ell)}}(\cdot)\}_{\ell=1,\ldots,L}$*, such that almost-surely, every honest* P_i *eventually outputs* $\{q^{\star^{(\ell)}}(\alpha_i)\}_{\ell=1,\ldots,L}$.

Proof. Let D be *corrupt* and network be *asynchronous* with up to t_a corruptions. If no honest party obtains any output, then the lemma holds trivially. So consider the case when some honest party obtains an output consisting of ℓ elements of $\mathbb{F}$. We note that every honest party participates with some input in the instance of Π_{BA} at local time $2\Delta + 2T_{\mathsf{BC}}$. Hence, almost-surely, the instance of

Π_{BA} eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of Π_{BA} is 0**: This means that at local time $2\Delta + 2T_{\mathsf{BC}}$, at least one *honest* party, say P_h, has accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$, which has been received by P_h from the broadcast of D. Hence, by the *weak consistency* and *fallback consistency* of Π_{BC} in *asynchronous* network, all honest parties will eventually receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D. There will be at least $n - 2t_s - t_a > t_s$ *honest* parties in $\mathcal{E}$, whose univariate polynomials $q_i^{(\ell)}(x)$ are pair-wise consistent and define a (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$, for $\ell = 1, \ldots, L$. Similarly, the univariate polynomial $q_j^{(\ell)}(x)$ of every *honest* $P_j \in \mathcal{F}$ will be pair-wise consistent with the univariate polynomials $q_i^{(\ell)}(x)$ of all the *honest* parties in $\mathcal{E}$ and lie on $Q^{\star^{(\ell)}}(x, y)$. For $\ell = 1, \ldots, L$, let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. We claim that *every honest* P_i will eventually get $Q^{\star^{(\ell)}}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This implies that eventually every *honest* P_i outputs $s_i^{(\ell)} = Q^{\star^{(\ell)}}(0, \alpha_i) = q^{\star^{(\ell)}}(\alpha_i)$. To prove the claim, consider an arbitrary *honest* party P_i. There are three possible cases.
  - $P_i \in \mathcal{W}$ *and* $P_i \in \mathcal{F}$: In this case, P_i already has $q_i^{(\ell)}(x)$, received from D and since $P_i \in \mathcal{F}$, the condition $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$ holds.
  - $P_i \in \mathcal{W}$ *and* $P_i \notin \mathcal{F}$: In this case, P_i already has $q_i^{(\ell)}(x)$, received from D. Since $|\mathcal{W}| \geq n - t_s$ and $|\mathcal{F}| \geq n - t_s$, $|\mathcal{W} \cap \mathcal{F}| \geq n - 2t_s > t_s + t_a$. From the protocol steps, the polynomial $q_i^{(\ell)}(x)$ is pair-wise consistent with the polynomials $q_j^{(\ell)}(x)$ at least $n - t_s$ parties $P_j \in \mathcal{W}$ (since P_i has edges with at least $n - t_s$ parties P_j within $\mathcal{W}$). Now among these $n - t_s$ parties, at least $n - 2t_s$ parties will be from $\mathcal{F}$, of which at least $n - 2t_s - t_a > t_s$ parties will be *honest*. Hence, $q_i^{(\ell)}(x)$ is pair-wise consistent with the $q_j^{(\ell)}(x)$ polynomials of at least $t_s + 1$ *honest* parties $P_j \in \mathcal{F}$. Now since the $q_j^{(\ell)}(x)$ polynomials of all the *honest* parties in $\mathcal{F}$ lie on $Q^{\star^{(\ell)}}(x, y)$, it implies that $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$ holds.
  - $P_i \notin \mathcal{W}$: In this case, there will be $n - t_s$ parties in $\mathcal{F}$, of which at most t_a could be *corrupt*. Since $Q^{\star^{(\ell)}}(x, \alpha_i)$ is a t_s-degree polynomial, and $t_s < |\mathcal{F}| - 2t_a$, from Lemma A.1 it follows that by applying the OEC procedure on the common points on the $Q^{\star^{(\ell)}}(x, \alpha_i)$ polynomial received from the parties in $\mathcal{F}$, party P_i will eventually obtain $Q^{\star^{(\ell)}}(x, \alpha_i)$.
- **The output of Π_{BA} is 1**: Let P_h be the *first honest* party, who outputs ℓ values from $\mathbb{F}$. This means that P_h has received an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D, such that $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph. By the *weak consistency* and *fallback consistency* properties of Π_{BC} in the *asynchronous* network, *all* honest parties eventually receive $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D. Moreover, since the consistency graphs are constructed based on broadcasted `OK` messages and since $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph, from the *weak validity, fallback validity, weak consistency and fallback consistency* properties of Π_{BC} in the *asynchronous* network, the pair $(\mathcal{E}', \mathcal{F}')$ will eventually constitute an $(n, t_a)-$star in every honest party's consistency graph. Since $|\mathcal{E}'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, it follows that $\mathcal{E}'$ has at least $t_s + 1$ *honest* parties, whose univariate polynomials $q_i^{(\ell)}(x)$ are pair-wise consistent and hence from Lemma 2.1, lie on a unique degree-(t_s, t_s) symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$, for $\ell = 1, \ldots, L$. Similarly, since the univariate polynomial $q_j^{(\ell)}(x)$ of every *honest* party P_j in $\mathcal{F}'$ is pair-wise consistent with the univariate polynomials $q_i^{(\ell)}(x)$ of the *honest* parties in $\mathcal{E}'$, it implies that the univariate polynomial $q_j^{(\ell)}(x)$ of all the *honest* parties in $\mathcal{F}'$ also lie on $Q^{\star^{(\ell)}}(x, y)$, for $\ell = 1, \ldots, L$. Let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. We show that *every honest* P_i eventually outputs $\{q^{\star^{(\ell)}}(\alpha_i)\}_{\ell=1,\ldots,L}$. For this it is enough to show that

each honest P_i eventually gets $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$, as P_i outputs $q_i^{(\ell)}(0)$, which will be same as $q^{\star^{(\ell)}}(\alpha_i)$. Consider an arbitrary *honest* P_i. There are two possible cases.

- $P_i \in \mathcal{F}'$: In this case, P_i already has $Q^{\star^{(\ell)}}(x, \alpha_i)$, received from D, for $\ell = 1, \ldots, L$.
- $P_i \notin \mathcal{F}'$: In this case, $\mathcal{F}'$ has at least $n - t_a > 3t_s$ parties, of which at most t_a could be corrupt. Since $Q^{\star^{(\ell)}}(x, \alpha_i)$ is a t_s-degree polynomial and $t_s < |\mathcal{F}'| - 2t_a$, from Lemma A.1, it follows that by applying the OEC procedure on the common points on the polynomial $Q^{\star^{(\ell)}}(x, \alpha_i)$ received from the parties in $\mathcal{F}'$, party P_i will eventually obtain $Q^{\star^{(\ell)}}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.

$\square$

Lemma D.7 (Communication Complexity). *Protocol* Π_{WPS} *incurs a communication of* $\mathcal{O}(n^2 L \log |\mathbb{F}| + n^4 \log |\mathbb{F}|)$ *bits and invokes* 1 *instance of* Π_{BA}.

Proof. In the protocol, D sends L number of t_s-degree univariate polynomials to every party. As part of the pair-wise consistency checks, each pair of parties exchanges $2L$ field elements. In addition, an *honest* party may broadcast *only one* NOK message, corresponding to the smallest instance in which a conflict occurred, for every other party. As part of the NOK message, the honest party also broadcasts the corresponding common point on its univariate polynomial. Each such common point can be represented by $\log |\mathbb{F}|$ bits. The communication complexity now follows from the communication complexity of the protocol Π_{BC} (see Theorem 3.6). $\square$

The proof of Theorem 4.1 now follows easily from Lemmas D.1-D.7.

D.2 Proof of the Properties of the Protocol Π_{VSS}

In this section, we prove the properties of the protocol Π_{VSS}. We first present the modified Π_{VSS} protocol where D has L number of t_s-degree polynomials as input.

Protocol $\Pi_{\mathsf{VSS}}(\mathsf{D}, \{q^{(\ell)}(\cdot)\}_{\ell=1,\ldots,L})$

- **Phase I (Time Δ) — Sending Polynomials**:
  - D on having the input $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ chooses random (t_s, t_s)-degree symmetric bivariate polynomials $Q^{(1)}(x, y), \ldots, Q^{(L)}(x, y)$ such that $Q^{(\ell)}(0, y) = q^{(\ell)}(\cdot)$, for $\ell = 1, \ldots, L$ and sends $q_i^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_i)$ to each party $P_i \in \mathcal{P}$.
- **Phase II (Time T_{WPS}) — Exchanging Common Values**: Each P_i, upon receiving t_s-degree polynomials $\{q_i^{(\ell)}(x)\}_{\ell=1,\ldots,L}$ from D, waits till the current local time becomes a multiple of Δ and then does the following.
  - Act as a dealer and invoke L instances $\Pi_{\mathsf{WPS}}^{(i,\ell)}$ of Π_{WPS} to share $q_i^{(\ell)}(x)$, for $\ell = 1, \ldots, L$. For $j = 1, \ldots, n$, participate in the instances $\Pi_{\mathsf{WPS}}^{(j,\ell)}$, if invoked by P_j as a dealer, and wait for time T_{WPS}.
- **Phase III (Time T_{BC}) — Publicly Declaring the Results of Pair-Wise Consistency Test**: Each $P_i \in \mathcal{P}$ waits till the local time becomes a multiple of Δ and does the following.
  - If WPS-shares $q_{ji}^{(1)}, \ldots, q_{ji}^{(\ell)}$ are computed during the instances $\Pi_{\mathsf{WPS}}^{(j,1)}, \ldots, \Pi_{\mathsf{WPS}}^{(j,\ell)}$ respectively, then broadcast $\mathtt{OK}(i, j)$ if $q_{ji}^{(\ell)} = q_i^{(\ell)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$. Else, set $\mathsf{Conflict}_{ij} = \{\ell | q_{ji}^{(\ell)} \neq q_i^{(\ell)}(\alpha_j)\}$ and broadcast $\mathtt{NOK}(\ell_{\mathsf{min}}, i, j, q_i^{(\ell_{\mathsf{min}})}(\alpha_j))$ where ℓ_{min} is the minimum value in $\mathsf{Conflict}_{ij}$.
- **Local Computation — Constructing Consistency Graph**: Each P_i constructs the *consistency graph* G_i as in Π_{WPS}.
- **Phase IV (Time T_{BC}) — Constructing $(n, t_s)-$star**: D does the following in its consistency graph G_{D} at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$.

- Remove edges incident with P_i from G_{D}, if $\texttt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ is received from the broadcast of P_i through regular-mode such that $q_{ij}^{(\ell)} \neq Q^{(\ell)}(\alpha_j, \alpha_i)$.
- Compute $\mathcal{W} = \{P_i : \mathsf{deg}(P_i) \geq n - t_s\}$, where $\mathsf{deg}(P_i)$ denotes the degree of P_i in G_{D}. Remove party P_i from $\mathcal{W}$ if in G_{D}, party P_i is *not* incident with at least $n - t_s$ parties in $\mathcal{W}$. Repeat this step till no more parties can be removed from $\mathcal{W}$.
- Run algorithm $\mathsf{AlgStar}$ on $G_{\mathsf{D}}[\mathcal{W}]$, where $G_{\mathsf{D}}[\mathcal{W}]$ denotes the subgraph of G_{D} induced by the vertices in $\mathcal{W}$. If an $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$ is obtained, then broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$.

- **Local Computation — Verifying and Accepting** $(\mathcal{W}, \mathcal{E}, \mathcal{F})$: Each $P_i \in \mathcal{P}$ does the following at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is received from the broadcast of D through regular-mode, then *accept* it if following *were true* at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$:
    - $\not\exists P_j, P_k \in \mathcal{W}$, $\not\exists l \in \{1, \ldots, L\}$, such that $\texttt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\texttt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode, where $q_{jk}^{(\ell)} \neq q_{kj}^{(\ell)}$.
    - In G_i, $\mathsf{deg}(P_j) \geq n - t_s$ for all $P_j \in \mathcal{W}$ and P_j *had* edges with at least $n - t_s$ parties from $G_i[\mathcal{W}]$.
    - $(\mathcal{E}, \mathcal{F})$ *was* an $(n, t_s)-$star in $G_i[\mathcal{W}]$ and $\forall P_j, P_k \in \mathcal{W}$ where the edge (P_j, P_k) *was* present in G_i, the $\texttt{OK}(j, k)$ and $\texttt{OK}(k, j)$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode.
- **Phase V (Time T_{BA}) — Deciding Whether to Go for** $(n, t_a)-$**star**: At time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$, each party P_i participates in an instance of Π_{BA} with input $b_i = 0$ if a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted, else with input $b_i = 1$ and waits for time T_{BA}.
- **Local Computation — Computing VSS-Shares Through** $(\mathcal{W}, \mathcal{E}, \mathcal{F})$**:** If Π_{BA} outputs 0, then each P_i does the following.
  - If a $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ has not been received from D's broadcast, then wait till it is received from D's broadcast through fall-back mode.
  - If $P_i \in \mathcal{W}$, then output $\{q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.
  - Else, initialize $\mathcal{SS}_i^{(\ell)}$ to $\emptyset$, for $\ell = 1, \ldots, L$. Include $P_j \in \mathcal{F}$ to $\mathcal{SS}_i^{(\ell)}$ if a WPS-share $q_{ji}^{(\ell)}$ is computed during the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$. Wait till $|\mathcal{SS}_i^{(\ell)}| \geq t_s + 1$, for $\ell = 1, \ldots, L$. Then interpolate $\{(\alpha_j, q_{ij}^{(\ell)})\}_{P_j \in \mathcal{SS}_i^{(\ell)}}$ to get a t_s-degree polynomial $q_i^{(\ell)}(x)$ and output $\{q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.
- **Phase VI (Time T_{BC}) — Broadcasting** $(n, t_a)-$**star**: If the output of Π_{BA} is 1, then D does the following.
  - After every update in G_{D}, run $\mathsf{AlgStar}$ on G_{D}. If an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ is obtained, then broadcast $(\mathcal{E}', \mathcal{F}')$.
- **Local Computation — Computing VSS-Shares Through** $(n, t_a)-$**star**: If Π_{BA} outputs 1, then each P_i does the following.
  - Start participating in any instance of Π_{BC} invoked by D for broadcasting an $(n, t_a)-$star *only* after time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Wait till some $(\mathcal{E}', \mathcal{F}')$ is obtained from the broadcast of D (through any mode), which constitutes an $(n, t_a)-$star in G_i.
  - If $P_i \in \mathcal{F}'$, then output $\{q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$. Else, initialize $\mathcal{SS}_i^{(\ell)}$ to $\emptyset$, for $\ell = 1, \ldots, L$. Include $P_j \in \mathcal{F}'$ to $\mathcal{SS}_i^{(\ell)}$ if a WPS-share $q_{ji}^{(\ell)}$ is computed during the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$. Wait till $|\mathcal{SS}_i^{(\ell)}| \geq t_s + 1$, for $\ell = 1, \ldots, L$. Then interpolate $\{(\alpha_j, q_{ij}^{(\ell)})\}_{P_j \in \mathcal{SS}_i^{(\ell)}}$ to get a t_s-degree polynomial $q_i^{(\ell)}(x)$ and output $\{q_i^{(\ell)}(0)\}_{\ell=1,\ldots,L}$.

Figure 11: The best-of-both-worlds VSS protocol for L polynomials.

Lemma D.8 (Privacy). *In protocol Π_{VSS}, if D is honest, then irrespective of the network type, the view of the adversary remains independent of $\{q^{(\ell)}(\cdot)\}_{\ell=1,\ldots,L}$.*

Proof. Let D be *honest*. We consider the worst case scenario when adversary controls up to t_s parties. We claim that throughout the protocol, the adversary learns at most t_s univariate polynomials lying on $Q^{(\ell)}(x, y)$, for $\ell = 1, \ldots, L$. Since $Q^{(\ell)}(x, y)$ is a random (t_s, t_s)-degree symmetric-bivariate polynomial, it then follows from Lemma 2.1, that the view of the adversary will be independent of $q^{(\ell)}(\cdot)$. We next proceed to prove the claim.

Corresponding to every *corrupt* P_i, the adversary learns $Q^{(\ell)}(x, \alpha_i)$. Corresponding to every *honest* P_i, the adversary learns t_s number of $q_i^{(\ell)}(\alpha_j)$ values through pair-wise consistency tests. However, these values are already included in the view of the adversary (through the univariate polynomials under adversary's control). Additionally, from the *privacy* property of Π_{WPS}, the view of the adversary remains independent of $q_i^{(\ell)}(x)$ during $\Pi_{\mathsf{WPS}}^{(i,\ell)}$, if P_i is *honest*. Hence no additional information about the polynomials of the honest parties is revealed during the pair-wise consistency checks. Furthermore, no *honest* P_i broadcasts $\texttt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ corresponding to any *honest* P_j, since the pair-wise consistency check will always pass for every pair of *honest* parties. □

Lemma D.9 (Correctness in Synchronous Network). *In protocol* Π_{VSS}*, if* D *is honest and network is synchronous, then each honest* P_i *outputs* $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ *at time* $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$.

Proof. Let D be *honest* and network be *synchronous* with up to t_s corruptions. During phase I, all honest parties receive $q_i^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$ from D within time Δ. Consequently during phase II, each *honest* P_i shares the polynomials $\{q_i^{(\ell)}(x)\}_{\ell=1,\ldots,L}$. From the *correctness* of Π_{WPS} in *synchronous* network, corresponding to each *honest* P_j, every *honest* P_i computes the WPS-share $q_{ji}^{(\ell)} = q_j^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$, at time $\Delta + T_{\mathsf{WPS}}$. Consequently, during phase III, every *honest* party broadcasts an $\texttt{OK}$ message for every other *honest* party. From the *validity* property of Π_{BC} in *synchronous* network, these $\texttt{OK}$ messages are received by every honest party through regular-mode at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$. Hence, there will be an edge between every pair of *honest* parties in the consistency graph of every honest party. Moreover, if D receives an *incorrect* $\texttt{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ message from the broadcast of any *corrupt* P_j through regular-mode at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$ where $q_{ij}^{(\ell)} \neq Q^{(\ell)}(\alpha_j, \alpha_i)$, then D removes all the edges incident with P_i in D's consistency graph G_{D}. D then computes the set $\mathcal{W}$ and all *honest* parties will be present in $\mathcal{W}$. Moreover, the honest parties will form a clique of size at least $n - t_s$ in the subgraph $G_{\mathsf{D}}[\mathcal{W}]$ at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$ and D will find an $(n, t_s)-$star $(\mathcal{E}, \mathcal{F})$ in $G_{\mathsf{D}}[\mathcal{W}]$ and broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ during phase IV. By the *validity* of Π_{BC} in *synchronous* network, all honest parties will receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ through regular-mode at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. Moreover, all honest parties will accept *accept* $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ and participate with input 0 in the instance of Π_{BA}. By the *validity* and *guaranteed liveness* of Π_{BA}, the output of Π_{BA} will be 0 for every honest party at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Now consider an arbitrary *honest* party P_i. Since $P_i \in \mathcal{W}$, P_i outputs $s_i^{(\ell)} = q_i^{(\ell)}(0) = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$ □

Lemma D.10 (Correctness in Asynchronous Network). *In protocol* Π_{VSS}*, if* D *is honest and network is asynchronous, then almost-surely, each honest* P_i *eventually outputs* $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.

Proof. Let D be *honest* and network be *asynchronous* with up to t_a corruptions. We first note that every *honest* P_i eventually broadcasts $\texttt{OK}(i, j)$ message, corresponding to every *honest* P_j. This is because both P_i and P_j eventually receive $q_i^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_i)$ and $q_j^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_j)$ respectively from D, for $\ell = 1, \ldots, L$. Moreover, P_j shares $q_j^{(\ell)}(\cdot)$ during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ and from the *correctness* of Π_{WPS} in *asynchronous* network, party P_i eventually computes the WPS-share $q_{ji}^{(\ell)} = q_j^{(\ell)}(\alpha_i)$ during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$. Moreover, $q_{ji}^{(\ell)} = q_{ij}^{(\ell)}$ holds, for $\ell = 1, \ldots, L$. Note that every honest party participates with some input in the instance of Π_{BA} at local time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. Hence almost-surely, the instance of Π_{BA} eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of Π_{BA} is** 0: This means that at least one *honest* party, say P_h, has received $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D and accepted this. Hence, by the *weak validity* and *fallback validity* of Π_{BC}, all honest parties will eventually receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D. We claim that *every honest* P_i will eventually get $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This will imply that eventually every *honest* P_i outputs the VSS-share $s_i^{(\ell)} = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{W}$: In this case, P_i already has $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$, received from D.
  - $P_i \notin \mathcal{W}$: In this case, we first note that there will be at least $t_s + 1$ parties, who are eventually included in $\mathcal{SS}_i^{(\ell)}$, for $\ell = 1, \ldots, L$. This follows from the fact that there are at least $t_s + 1$ *honest* parties P_j in $\mathcal{F}$. And corresponding to every *honest* $P_j \in \mathcal{F}$, party P_i will eventually compute the WPS-share $q_{ji}^{(\ell)}$ in the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ (follows from the *correctness* of Π_{WPS} in *asynchronous* network). We next claim that corresponding to *every* $P_j \in \mathcal{SS}_i^{(\ell)}$, the value $q_{ij}^{(\ell)}$ computed by P_i is the *same* as $Q^{(\ell)}(\alpha_j, \alpha_i)$.

    The claim is obviously true for every *honest* $P_j \in \mathcal{SS}_i^{(\ell)}$, so consider a *corrupt* $P_j \in \mathcal{SS}_i^{(\ell)}$. We first note that the polynomial $q_j^{(\ell)}(x)$ shared by P_j during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ is the *same* as $Q^{(\ell)}(x, \alpha_j)$. This is because $P_j \in \mathcal{W}$ (since $\mathcal{F} \subseteq \mathcal{W}$) and hence P_j has edges with at least $n - t_s$ parties in $\mathcal{W}$ and hence with at least $n - t_s - t_s > t_s$ *honest* parties in $G_{\mathsf{D}}[\mathcal{W}]$. Let $\mathcal{H}$ be the set of *honest* parties in $\mathcal{W}$ with which P_j has edges in $G_{\mathsf{D}}[\mathcal{W}]$. This implies that *every* $P_k \in \mathcal{H}$ has broadcasted $\texttt{OK}(k, j)$ message after verifying that $q_{jk}^{(\ell)} = q_k^{(\ell)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$, where the polynomial $q_k^{(\ell)}(x)$ held by P_k is the same as $Q^{(\ell)}(x, \alpha_k)$ and where the WPS-share $q_{jk}^{(\ell)}$ computed by P_k during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ is the *same* as $q_j^{(\ell)}(\alpha_k)$ (follows from the *weak commitment* of Π_{WPS} in *synchronous* network). Since $|\mathcal{H}| > t_s$, it implies that at least $t_s + 1$ *honest* parties P_k have verified that $q_j^{(\ell)}(\alpha_k) = Q^{(\ell)}(\alpha_j, \alpha_k)$ holds. This further implies that $q_j^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_j)$. Since P_i has computed WPS-share $q_{ji}^{(\ell)}$ during $\Pi_{\mathsf{WPS}}^{(j,l)}$, from the *weak commitment* of Π_{WPS} in *synchronous* network, it follows that $q_{ji}^{(\ell)} = q_j^{(\ell)}(\alpha_i) = Q^{(\ell)}(\alpha_i, \alpha_j) = Q^{(\ell)}(\alpha_j, \alpha_i)$, for $\ell = 1, \ldots, L$. The last equality follows since each $Q^{(\ell)}(x, y)$ is a symmetric bivariate polynomial.
- **The output of Π_{BA} is** 1: As mentioned earlier, since D is *honest*, every pair of honest parties eventually broadcast $\texttt{OK}$ messages corresponding to each other, as the pair-wise consistency check between them will be eventually positive. From the *weak validity* and *fallback validity* of Π_{BC} in *asynchronous* network, these messages are eventually delivered to every honest party. Also from the *weak consistency* and *fallback consistency* of Π_{BC} in *asynchronous* network, any $\texttt{OK}$ message which is received by D will be eventually received by every other honest party as well. As there will be at least $n - t_a$ honest parties, a clique of size at least $n - t_a$ will eventually form in the consistency graph of every honest party. Hence D will eventually find an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ in its consistency graph and broadcast it. From the *weak validity* and *fallback validity* of Π_{BC} in *asynchronous* network, this star will be eventually received by every honest party. Moreover, $(\mathcal{E}', \mathcal{F}')$ will be eventually an $(n, t_a)-$star in every honest party's consistency graph. We claim that *every honest* P_i will eventually get $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This will imply that eventually every *honest* P_i outputs $s_i^{(\ell)} = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{F}'$: In this case, P_i already has $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$, received from D.
  - $P_i \notin \mathcal{F}'$: In this case, we note that there will be will be at least $t_s + 1$ parties, who are eventually included in $\mathcal{SS}_i^{(\ell)}$, for $\ell = 1, \ldots, L$, such that corresponding to *every* $P_j \in \mathcal{SS}_i^{(\ell)}$, the value $q_{ji}^{(\ell)}$ computed by P_i is the same as $Q^{(\ell)}(\alpha_j, \alpha_i)$. The proof for this will be similar

as for the case when $P_i \notin \mathcal{W}$ and the output of Π_{BA} is 0 and so we skip the proof.

$\square$

Before we proceed to prove the *strong commitment* property in the *synchronous* network, we prove a helping lemma.

Lemma D.11. *In protocol Π_{VSS} if the network is synchronous and D is corrupt and if at least one honest party has received some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode and accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$, then all the following hold:*

- *All the honest parties in $\mathcal{W}$ have received their respective t_s-degree univariate polynomials by time Δ.*
- *For $\ell = 1, \ldots, L$, the univariate polynomials $q_i^{(\ell)}(x)$ of all honest parties P_i lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star(\ell)}(x, y)$.*
- *At time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$, every honest party accepts $(\mathcal{W}, \mathcal{E}, \mathcal{F})$.*

Proof. Let D be *corrupt* and network be *synchronous* with up to t_s corruptions. As per the lemma condition, let P_h be an *honest* party, who has received some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode and accepted it at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. From the protocol steps, the following must be true for P_h at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$:

- $\nexists P_j, P_k \in \mathcal{W}$ and $\nexists \ell \in \{1, \ldots, L\}$, such that $\mathtt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\mathtt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode, where $q_{jk}^{(\ell)} \neq q_{kj}^{(\ell)}$.
- In P_h's consistency graph G_h, $\mathsf{deg}(P_j) \geq n - t_s$ for all $P_j \in \mathcal{W}$ and P_j has edges with at least $n - t_s$ parties from $\mathcal{W}$.
- $(\mathcal{E}, \mathcal{F})$ *was* an $(n, t_s)-$star in the induced subgraph $G_h[\mathcal{W}]$, such that $\forall P_j, P_k \in \mathcal{W}$ where the edge (P_j, P_k) is present in G_h, the $\mathtt{OK}(j, k)$ and $\mathtt{OK}(k, j)$ messages *were* received from the broadcast of P_j and P_k respectively through regular-mode.

We prove the first part of the lemma through contradiction. So let $P_j \in \mathcal{W}$ be an *honest* party, who received its t_s-degree univariate polynomials $\{q_j^{(\ell)}(x)\}_{\ell=1,\ldots,L}$ at time $\Delta + \delta$, where $\delta > 0$. Moreover, let $P_k \in \mathcal{W}$ be an *honest* party such that P_j has an edge with P_k (note that P_j has edges with at least $n - 2t_s > t_s + t_a$ *honest* parties in $\mathcal{W}$). As stated above, at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$, party P_h has received the message $\mathtt{OK}(k, j)$ from the broadcast of P_k through regular-mode. From the protocol steps, P_j waits till its local time becomes a multiple of Δ, before it shares its polynomial $q_j^{(\ell)}(\cdot)$ in the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$. Hence, P_j must have started sharing $q_j^{(\ell)}(\cdot)$ through $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ at time $c \cdot \Delta$, where $c \geq 2$. Since the network is *synchronous*, from the *correctness* of Π_{WPS} in *synchronous* network, party P_k will compute its WPS-share $q_{jk}^{(\ell)}$ in $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ at time $c \cdot \Delta + T_{\mathsf{WPS}}$. Hence the result of the pair-wise consistency test with P_j will be available to P_k at time $c \cdot \Delta + T_{\mathsf{WPS}}$ and so P_k starts broadcasting $\mathtt{OK}(k, j)$ message only at time $c \cdot \Delta + T_{\mathsf{WPS}}$. Since P_k is *honest*, from the *validity* property of Π_{BC} in *synchronous* network, it will take *exactly* T_{BC} time for the message $\mathtt{OK}(k, j)$ to be received through regular-mode, once it is broadcasted. This implies that P_h will receive the message $\mathtt{OK}(k, j)$ at time $c \cdot \Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$, where $c \geq 2$. However, this is a contradiction, since the $\mathtt{OK}(k, j)$ message has been received by P_h at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$.

To prove the second part of the lemma, we will show that for $\ell = 1, \ldots, L$, the univariate polynomials $q_i^{(\ell)}(x)$ of all the *honest* parties $P_i \in \mathcal{W}$ are pair-wise consistent. Since there are at least $n - 2t_s > t_s$ *honest* parties in $\mathcal{W}$, from Lemma 2.1 it follows that these polynomials lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star(\ell)}(x, y)$, for $\ell = 1, \ldots, L$. Consider an arbitrary pair of *honest* parties $P_j, P_k \in \mathcal{W}$. From the first part of the claim, both P_j and P_k must have received their respective univariate polynomials $q_j^{(\ell)}(x)$ and $q_k^{(\ell)}(x)$ by time Δ, for $\ell = 1, \ldots, L$. Moreover, from the *correctness* property of Π_{WPS} in *synchronous* network, P_j and P_k will compute the WPS-shares $q_{kj}^{(\ell)} = q_k^{(\ell)}(\alpha_j)$ and $q_{jk}^{(\ell)} = q_j^{(\ell)}(\alpha_k)$ at time $\Delta + T_{\mathsf{WPS}}$ in $\Pi_{\mathsf{WPS}}^{(k,\ell)}$ and $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ respectively, for $\ell = 1, \ldots, L$. Since P_j and P_k are honest and the network is synchronous, they will agree on the set of conflicts between them and thus, $\mathsf{Conflict}_{jk} = \mathsf{Conflict}_{kj}$. If $\mathsf{Conflict}_{jk} \neq \emptyset$, then P_j and P_k would broadcast $\texttt{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\texttt{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ messages respectively at time $\Delta + T_{\mathsf{WPS}}$, where ℓ is the minimum value in $\mathsf{Conflict}_{jk}$ *and* $\mathsf{Conflict}_{kj}$. From the *validity* property of Π_{BC} in the *synchronous* network, P_h will receive these messages through regular-mode at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$. Consequently, P_h *will not* accept $(\mathcal{W}, \mathcal{E}, \mathcal{F})$, which is a contradiction.

To prove the third part of the lemma, we note that since P_h has received $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$, it implies that D must have started broadcasting $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ latest at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$, since it takes T_{BC} time for the regular-mode of Π_{BC}. From the *consistency* property of Π_{BC} in *synchronous* network, it follows that every honest party will also receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. Since at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$, party P_h has received the $\texttt{OK}(j, k)$ and $\texttt{OK}(k, j)$ messages through regular-mode from the broadcast of every $P_j, P_k \in \mathcal{W}$ where (P_j, P_k) is an edge in P_h's consistency graph, it follows that these messages started getting broadcasted latest at time $\Delta + T_{\mathsf{WPS}}$. From the *validity* and *consistency* properties of Π_{BC} in the *synchronous* network, it follows that every honest party receives these broadcast messages through regular-mode at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$. Hence $(\mathcal{E}, \mathcal{F})$ will constitute an $(n, t_s)-$star in the induced subgraph $G_i[\mathcal{W}]$ of every honest party P_i's consistency-graph at time $\Delta + T_{\mathsf{WPS}} + T_{\mathsf{BC}}$ and consequently, every honest party accepts $(\mathcal{W}, \mathcal{E}, \mathcal{F})$.

□

Lemma D.12 (Strong Commitment in Synchronous Network). *In protocol Π_{VSS}, if D is corrupt and network is synchronous, then either no honest party obtains any output or there exist t_s-degree polynomials, say $\{q^{\star^{(\ell)}}(\cdot)\}_{\ell=1,\ldots,L}$, such that each honest P_i eventually outputs $\{q^{\star^{(\ell)}}(\alpha_i)\}_{\ell=1,\ldots,L}$, where the following hold.*

- *If any honest P_i obtains its output at time $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$, then every honest party obtains its output at time T_{VSS}.*
- *If any honest P_i obtains its output at time T where $T > T_{\mathsf{VSS}}$, then every honest party obtains its output by time $T + 2\Delta$.*

Proof. Let D be *corrupt* and network be *synchronous* with up to t_s corruptions. If no honest party obtains any output, then the lemma holds trivially. So consider the case when some *honest* party obtains an output. Now, there are two possible cases.

- **At least one honest party, say P_h, has received some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode and accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$:** In this case, from Lemma D.11, the polynomials $q_i^{(\ell)}(x)$ of all *honest* parties in $\mathcal{W}$ are guaranteed to lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$, for $\ell = 1, \ldots, L$, As per the protocol steps, P_h has also verified that $\mathcal{F} \subseteq \mathcal{W}$ (by checking that $(\mathcal{E}, \mathcal{F})$ constitutes an $(n, t_s)-$star in the induced subgraph $G_h[\mathcal{W}]$) and hence the polynomials $q_i^{(\ell)}(x)$ of all *honest* parties in $\mathcal{F}$ also lie on $Q^{\star^{(\ell)}}(x, y)$, for $\ell = 1, \ldots, L$. Moreover, from Lemma D.11, all honest parties accept $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. Hence, every honest party participates in Π_{BA} with input 0 and by the *validity* and *guaranteed liveness* properties of Π_{BA} in *synchronous* network, all honest parties compute the output 0 from Π_{BA} at time $T_{\mathsf{VSS}} = \Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Let $q^{\star^{(\ell)}}(\cdot) = Q^{\star^{(\ell)}}(0, y)$ and consider an arbitrary *honest* party P_i. We wish to show that P_i has $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$ at time T_{VSS}, which will imply that P_i outputs the VSS-share $s_i^{(\ell)} = q_i^{(\ell)}(0)$ at time T_{VSS}, which will be the same as $q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$. There are two possible cases.

  - $P_i \in \mathcal{W}$: In this case, P_i has already received $q_i^{(\ell)}(x)$ from D within time Δ, for $\ell = 1, \ldots, L$. This follows from Lemma D.11.
  - $P_i \notin \mathcal{W}$: In this case, we claim that at time T_{VSS}, there will be will be at least $t_s + 1$ parties from $\mathcal{F}$, who are included in $\mathcal{SS}_i^{(\ell)}$, for $\ell = 1, \ldots, L$, such that corresponding to *every* $P_j \in \mathcal{SS}_i^{(\ell)}$, party P_i will have the value $q_{ji}^{(\ell)}$, which will be the same as $Q^{\star^{(\ell)}}(\alpha_j, \alpha_i)$. Namely, there are at least $t_s + 1$ *honest* parties in $\mathcal{F}$, who will be included in $\mathcal{SS}_i^{(\ell)}$ and the claim will be trivially true for those parties, due to the *correctness* property of Π_{WPS} in synchronous network. On the other hand, if any *corrupt* $P_j \in \mathcal{F}$ is included in $\mathcal{SS}_i^{(\ell)}$, then the polynomial shared by P_j during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ will be pair-wise consistent with the polynomials of at least $t_s + 1$ *honest* parties in $\mathcal{W}$ and hence will be the same as $Q^{\star^{(\ell)}}(x, \alpha_j)$. Moreover, from the *weak commitment* of Π_{WPS} in *synchronous* network, the WPS-share $q_{ji}^{(\ell)}$ computed by P_i during $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ will be same as $Q^{\star^{(\ell)}}(\alpha_i, \alpha_j)$, which will be the same as $Q^{\star^{(\ell)}}(\alpha_j, \alpha_i)$, since $Q^{\star^{(\ell)}}(x, y)$ is a symmetric bivariate polynomial. Hence, P_i will interpolate $q_i^{(\ell)}(x)$, for $\ell = 1, \ldots, L$.

– **No honest party has received any $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D through regular-mode and accepted $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$**: This implies that all honest parties participate in Π_{BA} with input 1 and by the *validity* and *guaranteed liveness* of Π_{BA} in *synchronous* network, all honest parties obtain the output 1 from Π_{BA} at time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Let P_h be the *first honest* party, who obtains an output. This means that P_h has received an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D, such that $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph. Let T be the time when $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph. This implies that at time T, party P_h has $(\mathcal{E}', \mathcal{F}')$ from D's broadcast and also all the $\mathtt{OK}(\star, \star)$ messages, from the broadcast of respective parties in $\mathcal{E}'$ and $\mathcal{F}'$. From the protocol steps, $T > T_{\mathsf{VSS}}$, since the honest parties participate in the instance of Π_{BC} through which D has broadcasted $(\mathcal{E}', \mathcal{F}')$ *only* after time T_{VSS}. By the *consistency* and *fallback consistency* properties of Π_{BC} in *synchronous* network, *all* honest parties will receive $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D by time $T + 2\Delta$. Moreover, $(\mathcal{E}', \mathcal{F}')$ will constitute an $(n, t_a)-$star in every honest party's consistency graph by time $T + 2\Delta$. This is because all the $\mathtt{OK}$ messages which are received by P_h from the broadcast of various parties in $\mathcal{E}'$ and $\mathcal{F}'$ are guaranteed to be received by every honest party by time $T + 2\Delta$. Since $|\mathcal{E}'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, it follows that $\mathcal{E}'$ has at least $t_s + 1$ *honest* parties. Moreover, for $\ell = 1, \ldots, L$, the univariate polynomials $(q_j^{(\ell)}(x), q_k^{(\ell)}(x))$ of every pair of *honest* parties $P_j, P_k \in \mathcal{E}'$ will be pair-wise consistent and hence lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)$ Similarly, the univariate polynomial $q_i^{(\ell)}(x)$ of every *honest* party P_i in $\mathcal{F}'$ is pair-wise consistent with the univariate polynomials $q_j^{(\ell)}(x)$ of *all* the *honest* parties in $\mathcal{E}'$ and hence lie on $Q^{\star^{(\ell)}}(x, y)$ as well, for $\ell = 1, \ldots, L$. Let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. We show that *every honest* P_i outputs $q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$, by time $T + 2\Delta$. For this it is enough to show that each honest P_i has $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$ by time $T + 2\Delta$, as P_i outputs $q_i^{(\ell)}(0)$, which will be the same as $q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$. Consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{F}'$: In this case, P_i has already received $Q^{\star^{(\ell)}}(x, \alpha_i)$, for $\ell = 1, \ldots, L$ from D, well before time $T + 2\Delta$.
  - $P_i \notin \mathcal{F}'$: In this case, we claim that by time $T + 2\Delta$, there will be will be at least $t_s + 1$ parties from $\mathcal{F}'$, who are included in $\mathcal{SS}_i^{(\ell)}$, such that corresponding to *every* $P_j \in \mathcal{SS}_i^{(\ell)}$, party P_i will have the value $q_{ij}^{(\ell)}$, which will be the same as $Q^{\star^{(\ell)}}(\alpha_j, \alpha_i)$, for $\ell = 1, \ldots, L$.

The proof for this is very similar to the previous case when $P_i \notin \mathcal{W}$ and the output of Π_{BA} is 0. Namely every *honest* $P_j \in \mathcal{F}'$ will be included in $\mathcal{SS}_i^{(\ell)}$. This is because P_j starts broadcasting `OK` messages for other parties in $\mathcal{E}'$ only after sharing its polynomial in the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$. Hence, by time $T + 2\Delta$, the WPS-share $q_{ji}^{(\ell)}$ from the instance $\Pi_{\mathsf{WPS}}^{(j,\ell)}$ will be available with P_i, for $\ell = 1, \ldots, L$. On the other hand, if a *corrupt* $P_j \in \mathcal{F}'$ is included in $\mathcal{SS}_i^{(\ell)}$, then also the claim holds (the proof for this is similar to the proof of the *correctness* property in *asynchronous* network in Lemma D.10).

□

Lemma D.13 (Strong Commitment in Asynchronous Network). *In protocol Π_{VSS}, if* D *is corrupt and network is asynchronous, then either no honest party obtains any output or there exist t_s-degree polynomials, say $\{q^{\star(\ell)}(\cdot)\}_{\ell=1,\ldots,L}$, such that almost-surely, every honest P_i eventually outputs $\{q^{\star(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$.*

Proof. Let D be *corrupt* and the network be *asynchronous* with up to t_a corruptions. If no honest party obtains any output, then the lemma holds trivially. So, consider the case when some honest party obtains an output. We note that every *honest* party participates with some input in the instance of Π_{BA} at local time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$. Hence, almost-surely, the instance of Π_{BA} eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of Π_{BA} is 0**: This means that at least one honest party, say P_h, has received some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D and accepted this at local time $\Delta + T_{\mathsf{WPS}} + 2T_{\mathsf{BC}}$ Hence, by the *weak consistency* and *fallback consistency* of Π_{BC} in *asynchronous* network, all honest parties will eventually receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ from the broadcast of D. There will be at least $n - 2t_s - t_a > t_s$ *honest* parties P_i in $\mathcal{E}$, whose univariate polynomials $q_i^{(\ell)}(x)$ are pair-wise consistent and hence lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star(\ell)}(x, y)$, for $\ell = 1, \ldots, L$. Similarly, for $\ell = 1, \ldots, L$, the univariate polynomial $q_i^{(\ell)}(x)$ of every *honest* $P_i \in \mathcal{F}$ will be pair-wise consistent with the univariate polynomials $q_j^{(\ell)}(x)$ of *all* the *honest* parties P_j in $\mathcal{E}$ and hence will lie on $Q^{\star(\ell)}(x, y)$ as well. Let $q^{\star(\ell)}(\cdot) \overset{def}{=} Q^{\star(\ell)}(0, y)$. We claim that *every honest P_i* will eventually have $Q^{\star(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$. This will imply that eventually every *honest* P_i outputs the VSS-share $s_i^{(\ell)} = Q^{\star(\ell)}(0, \alpha_i) = q^{\star(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary *honest* party P_i. There are three possible cases.
  - *$P_i \in \mathcal{W}$ and $P_i \in \mathcal{F}$*: In this case, P_i has received the polynomials $q_i^{(\ell)}(x)$ from D and since $P_i \in \mathcal{F}$, the condition $q_i^{(\ell)}(x) = Q^{\star(\ell)}(x, \alpha_i)$ holds.
  - *$P_i \in \mathcal{W}$ and $P_i \notin \mathcal{F}$*: In this case, P_i has received the polynomials $q_i^{(\ell)}(x)$ from D. Since $|\mathcal{W}| \geq n - t_s$ and $|\mathcal{F}| \geq n - t_s$, $|\mathcal{W} \cap \mathcal{F}| \geq n - 2t_s > t_s + t_a$. From the protocol steps, the polynomial $q_i^{(\ell)}(x)$ is pair-wise consistent with the polynomials $q_j^{(\ell)}(x)$ at least $n - t_s$ parties $P_j \in \mathcal{W}$ (since P_i has edges with at least $n - t_s$ parties P_j within $\mathcal{W}$). Now among these $n - t_s$ parties, at least $n - 2t_s$ parties will be from $\mathcal{F}$, of which at least $n - 2t_s - t_a > t_s$ parties will be *honest*. Hence, $q_i^{(\ell)}(x)$ is pair-wise consistent with the $q_j^{(\ell)}(x)$ polynomials of at least $t_s + 1$ *honest* parties $P_j \in \mathcal{F}$. Now since the $q_j^{(\ell)}(x)$ polynomials of all the *honest* parties in $\mathcal{F}$ lie on $Q^{\star(\ell)}(x, y)$, it implies that $q_i^{(\ell)}(x) = Q^{\star(\ell)}(x, \alpha_i)$ holds.
  - *$P_i \notin \mathcal{W}$*: In this case, similar to the proof of Lemma D.10, one can show that P_i eventually includes at least $t_s + 1$ parties from $\mathcal{F}$ in $\mathcal{SS}_i^{(\ell)}$, for $\ell = 1, \ldots, L$ and the value computed by P_i corresponding to *any* $P_j \in \mathcal{SS}_i^{(\ell)}$ will be the same as $Q^{\star(\ell)}(\alpha_j, \alpha_i)$. Hence, P_i will eventually interpolate $Q^{\star(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.

- **The output of Π_{BA} is 1**: Let P_h be the *first honest* party, who obtains an output in Π_{VSS}. This means that P_h has received an $(n, t_a)-$star $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D, such that $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph. By the *weak consistency* and *fallback consistency* properties of Π_{BC} in *asynchronous* network, *all* honest parties eventually receive $(\mathcal{E}', \mathcal{F}')$ from the broadcast of D. Moreover, since the consistency graphs are constructed based on broadcasted `OK` messages and since $(\mathcal{E}', \mathcal{F}')$ constitutes an $(n, t_a)-$star in P_h's consistency graph, from the *weak validity, fallback validity, weak consistency and fallback consistency* properties of Π_{BC} in *asynchronous* network, the pair $(\mathcal{E}', \mathcal{F}')$ will eventually constitute an $(n, t_a)-$star in every honest party's consistency graph, as the corresponding `OK` messages are eventually received by every honest party. Since $|\mathcal{E}'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, it follows that $\mathcal{E}'$ has at least $t_s + 1$ *honest* parties P_i, whose univariate polynomials $q_i^{(\ell)}(x)$ are pair-wise consistent and hence lie on a unique (t_s, t_s)-degree symmetric bivariate polynomial, say $Q^{\star^{(\ell)}}(x, y)\}$, for $\ell = 1, \ldots, L$. Similarly, since the univariate polynomials $q_j^{(\ell)}(x)$ of every *honest* party P_j in $\mathcal{F}'$ is pair-wise consistent with the univariate polynomials $q_i^{(\ell)}(x)$ of all the *honest* parties P_i in $\mathcal{E}'$, it implies that the polynomials $q_j^{(\ell)}(x)$ of all the *honest* parties P_j in $\mathcal{F}'$ also lie on $Q^{\star^{(\ell)}}(x, y)$ as well, for $\ell = 1, \ldots, L$. Let $q^{\star^{(\ell)}}(\cdot) \overset{def}{=} Q^{\star^{(\ell)}}(0, y)$. We show that *every honest* P_i eventually outputs $q^{\star^{(\ell)}}(\alpha_i)$, for $\ell = 1, \ldots, L$. For this it is enough to show that each honest P_i eventually gets $q_i^{(\ell)}(x) = Q^{\star^{(\ell)}}(x, \alpha_i)$, as P_i outputs $q_i^{(\ell)}(0)$, which will be the same as $q^{\star^{(\ell)}}(\alpha_i)$. Consider an arbitrary *honest* party P_i. There are two possible cases.
  - $P_i \in \mathcal{F}'$: In this case, P_i already has received $Q^{\star^{(\ell)}}(x, \alpha_i)$ from D, for $\ell = 1, \ldots, L$.
  - $P_i \notin \mathcal{F}'$: Again in this case, one can show that P_i eventually includes at least $t_s + 1$ parties from $\mathcal{F}'$ in $\mathcal{SS}_i^{(\ell)}$, for $\ell = 1, \ldots, L$ and the value computed by P_i corresponding to *any* $P_j \in \mathcal{SS}_i^{(\ell)}$ will be the same as $Q^{\star^{(\ell)}}(\alpha_j, \alpha_i)$, for $\ell = 1, \ldots, L$. Hence, P_i will eventually interpolate $Q^{\star^{(\ell)}}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.

□

Lemma D.14 (Communication Complexity). *Protocol* Π_{VSS} *incurs a communication of* $\mathcal{O}(n^3 L \log|\mathbb{F}| + n^5 \log|\mathbb{F}|)$ *bits and invokes* $n + 1$ *instance of* Π_{BA}.

Proof. The proof follows from Lemma D.7 and the fact that each party acts as a dealer and invokes an instance of Π_{WPS} to share L polynomials. Hence, the total communication cost due to Π_{WPS} in Π_{VSS} will be $\mathcal{O}(n \cdot (n^2 L \log|\mathbb{F}| + n^4 \log|\mathbb{F}|)) = \mathcal{O}(n^3 L \log|\mathbb{F}| + n^5 \log|\mathbb{F}|)$ bits, along with n instances of Π_{BA}. Additionally, there is an instance of Π_{BA} invoked in Π_{VSS} to agree on whether some $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ is accepted. □

The proof of Theorem 4.2 now follows easily from Lemmas D.8-D.14.

E The Protocol for Generating Shared Random Multiplication-Triples

In this section, we describe how we get our "best-of-both-worlds" protocol for generating the shared random multiplication-triples by plugging in the protocols Π_{BA} and Π_{VSS} in the framework of [23].

E.1 The Building Blocks

We first start with the various building blocks required in the framework of [23].

E.1.1 Beaver's Circuit-Randomization Protocol

Let x, y be two t_s-shared values and let (a, b, c) be a t_s-shared triple. Then the Beaver's protocol Π_{Beaver} for computing $x \cdot y$ is presented in Fig 12.

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

- **Masking Input Values (Local Computation)** — parties locally compute $[e] = [x] - [a]$ and $[d] = [y] - [b]$.
- **Publicly Reconstructing Masked Inputs (Time Δ)** — each $P_i \in \mathcal{P}$ does the following:
  - Send the share of e and d to every party in $\mathcal{P}$ and wait for Δ time.
  - Apply the $\mathsf{OEC}(t_s, t_s, \mathcal{P})$ procedure on the received shares of e and d to compute e and d.
- **Output Computation (Local Computation)** — parties locally compute $[z] = d \cdot e + e \cdot [b] + d \cdot [a] + [c]$ and output $[z]$.

Figure 12: Beaver's protocol for multiplying two t_s shared values.

Lemma E.1. *Let x and y be two t_s-shared values and let (a, b, c) be a t_s-shared triple. Then protocol Π_{Beaver} achieves the following properties in the presence of up to t_s corruptions.*

- **Correctness***: If the network is synchronous, then after time Δ, the parties hold a t_s-sharing of z, while in an asynchronous network, a t_s-sharing of z is eventually generated. Moreover, $z = x \cdot y$ holds, if and only if (a, b, c) is a multiplication-triple.*
- **Privacy***: If (a, b, c) is random from the point of view of the adversary, then the view of the adversary remains independent of x and y.*
- **Communication Complexity***: The protocol incurs a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

Proof. Since x, y and the triple (a, b, c) are all t_s-shared, the values $d = (x - a)$ and $e = (y - b)$ will be t_s-shared, which follows from the linearity of t_s-sharing. Let there be up to t_s corruptions. If the network is *synchronous*, then from the *correctness* property of OEC in the *synchronous* network (see Lemma A.1), at time Δ, *every* honest P_i will have d and e and hence the parties hold a t_s-sharing of z at time Δ. On the other hand, if the network is *asynchronous*, then from the *correctness* property of OEC in *asynchronous* network, *every* honest P_i *eventually* reconstructs d and e and hence the honest parties eventually hold a t_s-sharing of z. In the protocol, $z = (x-a)\cdot(y-b)+(x-a)\cdot b+(y-b)\cdot a+c = x \cdot y - a \cdot b + c$ holds. Hence it follows that $z = x \cdot y$ holds if and only if $c = a \cdot b$ holds.

In the protocol, adversary learns the values d and e, as they are publicly reconstructed. However, if a and b are random from the point of view of the adversary, then d and e leak no information about x and y. Namely, for every candidate x and y, there exist unique a and b, consistent with d and e.

The communication complexity follows from the fact each party needs to send 2 shares to every other party. □

E.1.2 The Triple-Transformation Protocol $\Pi_{\mathsf{TripTrans}}$

Protocol $\Pi_{\mathsf{TripTrans}}$ takes as input a set of $2d+1$ t_s-shared triples $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i=1,\ldots,2d+1}$, where the triples may not be "related". The protocol outputs "co-related" t_s-shared triples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})\}_{i=1,\ldots,2d+1}$, such that all the following hold (irrespective of the network type):

- There exist d-degree polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $2d$-degree polynomial $\mathsf{Z}(\cdot)$, such that $\mathsf{X}(\alpha_i) = \mathbf{x}^{(i)}$, $\mathsf{Y}(\alpha_i) = \mathbf{y}^{(i)}$ and $\mathsf{Z}(\alpha_i) = \mathbf{z}^{(i)}$ holds for $i = 1, \ldots, 2d+1$.
- The triple $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ is a multiplication-triple iff $(x^{(i)}, y^{(i)}, z^{(i)})$ is a multiplication-triple. This further implies that $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds iff all the $2d+1$ input triples are multiplication-triples.

– Adversary learns the triple $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ iff it knows the input triple $(x^{(i)}, y^{(i)}, z^{(i)})$.

The idea behind $\Pi_{\mathsf{TripTrans}}$ is as follows: the polynomials $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ are "defined" by the first and second components of the *first* $d+1$ input triples. Hence the first $d+1$ points on these polynomials are already t_s-shared. The parties then compute d "new" points on the polynomials $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ in a shared fashion. This step requires the parties to perform only *local computations*. This is because from the property of Lagrange's interpolation, computing any new point on $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ involves computing a *publicly-known linear* function (which we call Lagrange's linear function) of "old" points on these polynomials. Since the old points are t_s-shared, by applying corresponding Lagrange's functions, the parties can compute a t_s-sharing of the new points. Finally, the parties compute a t_s-sharing of the product of the d new points using Beaver's technique, making use of the *remaining* d input triples. The $\mathsf{Z}(\cdot)$ polynomial is then defined by the d computed products and the third component of the first $d+1$ input triples. The protocol is presented in Fig 13.

Protocol $\Pi_{\mathsf{TripTrans}}(d, \{[x^{(i)}], [y^{(i)}], [z^{(i)}]\}_{i=1,\ldots,2d+1})$

- **Defining** $\mathsf{X}(\cdot)$ **and** $\mathsf{Y}(\cdot)$ **Polynomials (Local Computation)** — The parties locally do the following:
  - For $i = 1, \ldots, d+1$, set $[\mathbf{x}^{(i)}] = [x^{(i)}]$, $[\mathbf{y}^{(i)}] = [y^{(i)}]$ and $[\mathbf{z}^{(i)}] = [z^{(i)}]$.
  - Let $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ be the unique d-degree polynomials passing through the points $\{(\alpha_i, \mathbf{x}^{(i)})\}_{i=1,\ldots,d+1}$ and $\{(\alpha_i, \mathbf{y}^{(i)})\}_{i=1,\ldots,d+1}$ respectively.
    - For $i = d+2, \ldots, 2d+1$, compute $[\mathbf{x}^{(i)}] = [\mathsf{X}(\alpha_i)]$ from $\{[\mathbf{x}^{(i)}]\}_{i=1,\ldots,d+1}$ by applying the corresponding Lagrange's linear function.
    - For $i = d+2, \ldots, 2d+1$, compute $[\mathbf{y}^{(i)}] = [\mathsf{Y}(\alpha_i)]$ from $\{[\mathbf{y}^{(i)}]\}_{i=1,\ldots,d+1}$ by applying the corresponding Lagrange's linear function.
- **Computing Points on the** $\mathsf{Z}(\cdot)$ **Polynomial (Time Δ)** — The parties do the following:
  - For $i = d+2, \ldots, 2d+1$, participate in the instance $\Pi_{\mathsf{Beaver}}(([\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}]), ([x^{(i)}], [y^{(i)}], [z^{(i)}]))$ of Π_{Beaver}. Let $[\mathbf{z}^{(i)}]$ be the output obtained from this instance.
  - Output $\{[\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}], [\mathbf{z}^{(i)}]\}_{i=1,\ldots,2d+1}$.

Figure 13: Protocol for transforming a set of t_s-shared triples to a set of correlated t_s-shared triples.

Lemma E.2. *Let $\{[x^{(i)}], [y^{(i)}], [z^{(i)}]\}_{i=1,\ldots,2d+1}$ be a set of t_s-shared triples. Then protocol $\Pi_{\mathsf{TripTrans}}$ achieves the following properties in the presence of up to t_s corruptions.*

- **Correctness**: *All the following hold.*
  - *If the network is synchronous, then after time Δ, the parties have t_s-shared triples $\{[\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}], [\mathbf{z}^{(i)}]\}_{i=1,\ldots,2}$ while in an asynchronous network, the parties eventually have t_s-shared triples $\{[\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}], [\mathbf{z}^{(i)}]\}_{i=1,\ldots,2d+}$*
  - *There exist d-degree polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $2d$-degree polynomial $\mathsf{Z}(\cdot)$, such that $\mathsf{X}(\alpha_i) = \mathbf{x}^{(i)}$, $\mathsf{Y}(\alpha_i) = \mathbf{y}^{(i)}$ and $\mathsf{Z}(\alpha_i) = \mathbf{z}^{(i)}$ holds for $i = 1, \ldots, 2d+1$.*
  - *$(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ is a multiplication-triple iff $(x^{(i)}, y^{(i)}, z^{(i)})$ is a multiplication-triple.*
- **Privacy**: *For $i = 1, \ldots, 2d+1$, no additional information about $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ is revealed to the adversary, if the triple $(x^{(i)}, y^{(i)}, z^{(i)})$ is random from the point of view of the adversary.*
- **Communication Complexity**: *The protocol incurs a communication of $\mathcal{O}(dn^2 \log |\mathbb{F}|)$ bits.*

Proof. Consider an adversary who controls up to t_s parties. In the protocol, irrespective of the network type, the parties *locally* compute the t_s-sharings $\{[\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}]\}_{i=1,\ldots,2d+1}$ and t_s-sharings $\{[\mathbf{z}^{(i)}]\}_{i=1,\ldots,d+1}$. If the network is *synchronous*, then from the *correctness* property of Π_{Beaver} in the *synchronous* network (see Lemma E.1), it follows that after time Δ, all honest parties will have their respective output in all the d instances of Π_{Beaver}. Hence after time Δ, the parties have t_s-sharings $\{[\mathbf{z}^{(i)}]\}_{i=d+2,\ldots,2d+1}$. On the other hand, if the network is *asynchronous*, then from the *correctness* property of Π_{Beaver} in the *asynchronous* network (see Lemma E.1), it follows that all the d instances of Π_{Beaver} *eventually* produce an output for each honest party. Hence the parties eventually compute the t_s-sharings $\{[\mathbf{z}^{(i)}]\}_{i=d+2,\ldots,2d+1}$ and hence eventually obtain their output.

Next consider *any* $i \in \{1, \ldots, d+1\}$. Since $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)}) = (x^{(i)}, y^{(i)}, z^{(i)})$, it follows that $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ will be a multiplication-triple if and only if $(x^{(i)}, y^{(i)}, z^{(i)})$ is a multiplication-triple. Now consider *any* $i \in \{d+2, \ldots, 2d+1\}$. Since $[\mathbf{z}^{(i)}]$ is the output of the instance $\Pi_{\mathsf{Beaver}}(([\mathbf{x}^{(i)}], [\mathbf{y}^{(i)}]), ([x^{(i)}], [y^{(i)}], [z^{($ it follows from the *correctness* property of Π_{Beaver} that $\mathbf{z}^{(i)} = \mathbf{x}^{(i)} \cdot \mathbf{y}^{(i)}$ holds, if and only if $(x^{(i)}, y^{(i)}, z^{(i)})$ is a multiplication-triple.

From the protocol steps, it is easy to see that the polynomials $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ defined in the protocols are d-degree polynomials, as they are defined through $d+1$ distinct points $\{(\alpha_i, \mathbf{x}^{(i)})\}_{i=1,\ldots,d+1}$ and $\{(\alpha_i, \mathbf{y}^{(i)})\}_{i=1,\ldots,d+1}$ respectively. On the other hand, $\mathsf{Z}(\cdot)$ is a $2d$-degree polynomial, as it is defined through the $2d+1$ distinct points $\{(\alpha_i, \mathbf{z}^{(i)})\}_{i=1,\ldots,2d+1}$.

For any $i \in \{1, \ldots, d+1\}$, if $(x^{(i)}, y^{(i)}, z^{(i)})$ is random from the point of view of the adversary, then $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ is also random from the point of view of the adversary, since $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)}) = (x^{(i)}, y^{(i)}, z^{(i)})$. On the other hand for any $i \in \{d+2, \ldots, 2d+1\}$, if $(x^{(i)}, y^{(i)}, z^{(i)})$ is random from the point of view of the adversary, then from the *privacy* property of Π_{Beaver} (see Lemma E.1), it follows that no additional information is learnt about $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})$.

The communication complexity follows from the fact that there are d instances of Π_{Beaver} invoked in the protocol. □

E.1.3 Agreement on a Common Subset

In the protocol Π_{ACS}, every party P_i will have a set of L number of t_s-degree polynomials, such that P_i would like to acts as a dealer and distribute points on these polynomials by invoking an instance of Π_{VSS}. As *corrupt* parties may not invoke their instances of Π_{VSS}, the parties may obtain points on the polynomials of only $n - t_s$ parties (even in a *synchronous* network). On the other hand, in an *asynchronous* network, different parties may obtain points on the polynomials of different subsets of $n - t_s$ parties. Protocol Π_{ACS} lets the parties agree on a *common subset* $\mathcal{CS}$ of at least $n - t_s$ parties, whose points are received by all honest parties, such that in a *synchronous* network, all *honest* parties are guaranteed to be present in $\mathcal{CS}$.

The ACS protocol is presented in Fig 14, where for simplicity we assume that $L = 1$. However, the protocol can be easily generalized for any $L \geq 1$. In the protocol, each party acts as a dealer and invokes an instance of Π_{VSS} to distribute points on its polynomial. After the time-out T_{VSS} of Π_{VSS} (within which the honest parties would have received points corresponding to the polynomials of the *honest* dealers in a *synchronous* network), the parties *locally* check for the instances of Π_{VSS} in which they have received an output and then start participating in n instances of Π_{BA} where the j^{th} instance is to decide whether P_j should be included in $\mathcal{CS}$. The input criteria for these Π_{BA} instances is the following: if a party has received an output in the Π_{VSS} instance with P_j as the dealer, then the party starts participating with input 1 in the corresponding Π_{BA} instance. Now once 1 is obtained as the output from $n - t_s$ instances of Π_{BA}, then the parties start participating with input 0 in any of the remaining Π_{BA} instances for which the parties may have not provided any input yet. Finally, once output is obtained from all the n instances of Π_{BA}, party P_j is included in $\mathcal{CS}$ iff the output of the corresponding Π_{BA} instance is 1. Since the parties wait for time T_{VSS} *before* starting the Π_{BA} instances, it is ensured that all *honest* dealers are included in $\mathcal{CS}$ in a *synchronous* network.

Protocol Π_{ACS}

- **Phase I — Distributing Points on the Polynomials (Time T_{VSS})**:
  - On having the input $f_i(\cdot)$, act as the dealer D and invoke an instance $\Pi_{\mathsf{VSS}}^{(i)}$ of Π_{VSS} with input $f_i(\cdot)$.
  - For $j = 1, \ldots, n$, participate in the instance $\Pi_{\mathsf{VSS}}^{(j)}$ invoked by P_j and wait for time T_{VSS}.

    - Initialize a set $\mathcal{C}_i = \emptyset$ after T_{VSS} time and include P_j in $\mathcal{C}_i$, if an output is obtained in $\Pi^{(j)}_{\mathsf{VSS}}$.
- **Phase II — Identifying the Set of Polynomial-Providers (Maximum Time $2T_{\mathsf{BA}}$):**
    - For $j = 1, \ldots, n$, participate in an instance of $\Pi^{(j)}_{\mathsf{BA}}$ of Π_{BA} with input 1, if $P_j \in \mathcal{C}_i$.
    - Once $n - t_s$ instances of Π_{BA} have produced an output 1, then participate with input 0 in all the Π_{BA} instances $\Pi^{(j)}_{\mathsf{BA}}$ such that $P_j \notin \mathcal{C}_i$.
    - Once all the n instances of Π_{BA} have produced a binary output, then output $\mathcal{CS}$, which is the set of parties P_j, such that 1 is obtained as the output in the instance $\Pi^{(j)}_{\mathsf{BA}}$.

Figure 14: Agreement on common subset of $n - t_s$ parties where each party has a single t_s-degree polynomial. The above code is executed by every $P_i \in \mathcal{P}$.

Lemma E.3. *Let every party P_i has a t_s-degree polynomial $f_i(\cdot)$ as input for the protocol Π_{ACS}. Then protocol Π_{ACS} achieves the following properties*

- *Synchronous Network: The following is achieved in the presence of up to t_s corruptions.*
    - **Correctness**: *at time $T_{\mathsf{ACS}} = T_{\mathsf{VSS}} + 2T_{\mathsf{BA}}$ the parties have a common subset $\mathcal{CS}$ of size at least $n - t_s$, such that all the following hold:*
        - *All honest parties will be present in $\mathcal{CS}$.*
        - *Corresponding to every honest $P_j \in \mathcal{CS}$, every honest P_i has $f_j(\alpha_i)$.*
        - *Corresponding to every corrupt $P_j \in \mathcal{CS}$, there exists some t_s-degree polynomial, say $f^\star_j(\cdot)$, such that every honest P_i has $f^\star_j(\alpha_i)$.*
- *Asynchronous Network: The following is achieved in the presence of up to t_a corruptions.*
    - **Correctness**: *almost-surely, the parties eventually output a common subset $\mathcal{CS}$ of size at least $n - t_s$, such that all the following hold:*
        - *Corresponding to every honest $P_j \in \mathcal{CS}$, every honest P_i eventually has $f_j(\alpha_i)$.*
        - *Corresponding to every corrupt $P_j \in \mathcal{CS}$, there exists some t_s-degree polynomial, say $f^\star_j(\cdot)$, such that every honest P_i eventually has $f^\star_j(\alpha_i)$.*
- **Privacy**: *The view of the adversary remains independent of the $f_i(\cdot)$ polynomials of the honest parties.*
- **Communication Complexity**: *the protocol incurs a communication of $\mathcal{O}(n^6 \log |\mathbb{F}|)$ bits and invokes $\mathcal{O}(n^2)$ instances of Π_{BA}.*

Proof. The *privacy* property simply follows from the *privacy* property of Π_{VSS}, while *communication complexity* follows from the *communication complexity* of Π_{VSS} (see Theorem 4.2) and the fact that $\mathcal{O}(n)$ instances of Π_{VSS} are invoked. We next prove the *correctness* property.

We first consider a *synchronous* network, with up to t_s corruptions. Let $\mathcal{H}$ be the set of parties, where $|\mathcal{H}| \geq n - t_s$. Corresponding to each $P_j \in \mathcal{H}$, every *honest* P_i obtains the output $f_j(\alpha_i)$ at time T_{VSS} during $\Pi^{(j)}_{\mathsf{VSS}}$, which follows from the *correctness* of Π_{VSS} in *synchronous* network (see Theorem 4.2). Consequently, at time T_{VSS}, the set $\mathcal{C}_i$ will be of size at least $n - t_s$ for every honest P_i. Now corresponding to each $P_j \in \mathcal{H}$, each honest P_i participates with input 1 in the instance $\Pi^{(j)}_{\mathsf{BA}}$ at time T_{VSS}. From the *validity* and *guaranteed liveness* of Π_{BA} in the *synchronous* network, it follows that at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, every honest P_i obtains the output 1 during the instance $\Pi^{(j)}_{\mathsf{BA}}$, corresponding to every $P_j \in \mathcal{H}$. Consequently, at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, every *honest* party will start participating in the remaining Π_{BA} instances for which no input has been provided yet (if there are any) and from the *guaranteed liveness* and *consistency* of Π_{BA} in the *synchronous* network, these Π_{BA} instances will produce common outputs for every honest party at time $T_{\mathsf{ACS}} = T_{\mathsf{VSS}} + 2T_{\mathsf{BA}}$. Hence, at time T_{ACS}, every honest party outputs a common $\mathcal{CS}$ of size at least $n - t_s$ and each $P_j \in \mathcal{H}$ will be present in $\mathcal{CS}$. We next wish to show that corresponding to *every* $P_j \in \mathcal{CS}$, every honest party has received its point on P_j's polynomial.

Consider an *arbitrary* party $P_j \in \mathcal{CS}$. If P_j is *honest*, then as argued above, every honest P_i gets $f_j(\alpha_i)$ at time T_{VSS} itself. Next, consider a *corrupt* $P_j \in \mathcal{CS}$. Since $P_j \in \mathcal{CS}$, it follows that the instance $\Pi_{\mathsf{BA}}^{(j)}$ produces the output 1. This further implies that at least one *honest* P_i must have obtained some output from the instance $\Pi_{\mathsf{VSS}}^{(j)}$ by time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$ (implying that $P_j \in \mathcal{C}_i$) and participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(j)}$. This is because if at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, party P_j *does not* belong to the $\mathcal{C}_i$ set of *any* honest P_i, then it implies that *all honest* parties participate with input 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ from time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$. Then, from the *validity* of Π_{BA} in the *synchronous* network, every honest party would obtain the output 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ and hence P_j will not be present in $\mathcal{CS}$, which is a contradiction. Now if P_i has obtained some output from $\Pi_{\mathsf{VSS}}^{(j)}$ at time $T_{\mathsf{VSS}} + T_{\mathsf{BA}}$, then from the *strong-commitment* of Π_{VSS}, it follows that P_j has some t_s-degree polynomial, say $f_j^{\star}(\cdot)$, such that every honest party P_i obtains $f_j^{\star}(\alpha_i)$ by time $T_{\mathsf{VSS}} + T_{\mathsf{BA}} + 2\Delta$. Since $2\Delta < T_{\mathsf{BA}}$, it follows that at time T_{ACS}, every honest P_i has $f_j^{\star}(\alpha_i)$, thus proving the *correctness* property in a *synchronous* network.

We next consider an *asynchronous* network, with up to t_a corruptions. Let $\mathcal{H}$ be the set of parties, where $|\mathcal{H}| \geq n - t_a \geq n - t_s$. We first note that irrespective of way messages are scheduled, there will be at least $n - t_s$ instances of Π_{BA} in which all honest parties eventually participate with input 1. This is because corresponding to every $P_j \in \mathcal{H}$, every *honest* P_i *eventually* obtains the output $f_j(\alpha_i)$ in the instance $\Pi_{\mathsf{VSS}}^{(j)}$, which follows from the *correctness* of Π_{VSS} in *asynchronous* network (see Theorem 4.2). So even if the *corrupt* parties P_j do not invoke their respective $\Pi_{\mathsf{VSS}}^{(j)}$ instances, there will be at least $n - t_s$ instances of Π_{BA} in which all *honest* parties eventually participate with input 1. Consequently, *almost-surely* these Π_{BA} instances eventually produce the output 1. Hence, all honest parties eventually participate with some input in the remaining Π_{BA} instances, which almost-surely produce some output for every honest party eventually. From the security of Π_{BA} in the *asynchronous* network (see Theorem 3.3), it follows that all the honest parties output the same $\mathcal{CS}$.

Now consider an arbitrary party $P_j \in \mathcal{CS}$. It implies that the honest parties obtain the output 1 from 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 receiving some output in the instance $\Pi_{\mathsf{VSS}}^{(j)}$. Now if P_j is *honest*, then the *correctness* of Π_{VSS} in *asynchronous* network (see Theorem 4.2) guarantees that every honest party P_i *eventually* obtains the output $f_j(\alpha_i)$ during $\Pi_{\mathsf{VSS}}^{(j)}$. On the other hand, if P_j is *corrupt*, then *strong commitment* of Π_{VSS} in *asynchronous* network (see Theorem 4.2) guarantees that there exists some t_s-degree polynomial, say $f_j^{\star}(\cdot)$, such that every honest party P_i eventually obtains the output $f_j^{\star}(\alpha_i)$ in the instance $\Pi_{\mathsf{VSS}}^{(j)}$. □

Protocol Π_{ACS} for Multiple Polynomials: Protocol Π_{ACS} can be easily extended if each party has L number of t_s-degree polynomials as input. In this case, each party now has to share ℓ polynomials in its corresponding instance of Π_{VSS}. The rest of the protocol steps remain the same. The protocol will incur a communication of $\mathcal{O}(n^4 L \log |\mathbb{F}| + n^6 \log |\mathbb{F}|)$ bits and invokes $\mathcal{O}(n^2)$ instances of Π_{BA}.

E.1.4 Triple-Sharing Protocol

Protocol Π_{TripSh} allows a dealer D to *verifiably* t_s-share L multiplication-triples, where if D is *honest*, then the triples remain random for the adversary. For a *corrupt* D, the protocol *need not* produce any output even in a *synchronous* network (as D may not invoke the protocol). However, the "verifiability" of Π_{TripSh} guarantees that if the honest parties obtain any output for a *corrupt* D, then

D has t_s-shared L multiplication-triples.

For simplicity, we present the protocol assuming D has a *single* multiplication-triple to share and the protocol can be easily generalized for any $L \geq 1$. The idea behind the protocol is as follows: D picks a random multiplication-triple and t_s-shares it by invoking an instance of Π_{VSS}. To prove that it has indeed shared a multiplication-triple, D actually t_s-shares $2t_s + 1$ random multiplication-triples. The parties then run an instance of $\Pi_{\mathsf{TripTrans}}$ and "transform" these shared triples into "co-related" shared triples, constituting distinct points on the triplet of polynomials $(\mathsf{X}(\cdot), \mathsf{Y}(\cdot), \mathsf{Z}(\cdot))$, which are guaranteed to be obtained from $\Pi_{\mathsf{TripTrans}}$. Then, to check if all the triples shared by D are multiplication-triples, it is sufficient to verify if $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds. To verify the latter, we incorporate a mechanism which enables the parties to *publicly* learn if $\mathsf{Z}(\alpha_j) = \mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j)$ holds under P_j's "supervision" in such a way that if P_j is *honest*, then the supervised verification of the triplet $(\mathsf{X}(\alpha_j), \mathsf{Y}(\alpha_j), \mathsf{Z}(\alpha_j))$ is "successful" iff the triplet is a multiplication-triple. Moreover, the privacy of the triplet will be maintained during the supervised verification for an *honest* D *and* P_j. The goal is to check whether there are at least $2t_s + 1$ successful supervised-verifications, performed under the supervision of *honest* supervisors P_j, which will then confirm that indeed $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds. This is because $\mathsf{Z}(\cdot)$ is a $2t_s$-degree polynomial. Upon confirming that $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds, the parties compute a "new" point on the polynomials (in a shared fashion), which is taken as the output triple shared *on behalf of* D. We stress that the output triple is well defined and will be "known" to D, as it is *deterministically* determined from the triples shared by D. If D is *honest*, then the privacy of the output triple is guaranteed from the fact that during the supervised verification, an adversary may learn at most t_s distinct points on the polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$, corresponding to the *corrupt* supervisors.

The supervised verification of the (shared) points on the polynomials is performed as follows: the parties invoke an instance of Π_{ACS}, where the input for each party is a triplet of random t_s-degree polynomials, whose constant terms constitute a random multiplication-triple, called *verification-triple*. The instance of Π_{ACS} is invoked in parallel with D's invocation of Π_{VSS}. Through the instance of Π_{ACS}, the parties agree upon a set $\mathcal{W}$ of at least $n - t_s$ supervisors, whose shared verification-triples are used to verify the points on the polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$. Namely, if $P_j \in \mathcal{W}$ has shared the verification-triple $(u^{(j)}, v^{(j)}, w^{(j)})$, then in the supervised verification under P_j, parties *publicly* reconstruct and check if $\mathsf{Z}(\alpha_j) - \mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j) = 0$ holds. For this, the parties recompute $\mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j)$ in a shared fashion using Beaver's method, by deploying the shared verification-triple $(u^{(j)}, v^{(j)}, w^{(j)})$. If $\mathsf{Z}(\alpha_j) - \mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j)$ *does not* turn out to be 0 (implying that either D is *corrupt* or P_j's verification-triple is *not* a multiplication-triple), then the parties *publicly* reconstruct and check if $(\mathsf{X}(\alpha_j), \mathsf{Y}(\alpha_j), \mathsf{Z}(\alpha_j))$ is a multiplication-triple and discard D if the triple *does not* turn out to be a multiplication-triple.

An *honest* D will *never* be discarded. Moreover, in a *synchronous* network, *all honest* parties P_j are guaranteed to be present in $\mathcal{W}$ (follows from the *correctness* of Π_{ACS} in a *synchronous* network) and hence, there will be at least $n - t_s > 2t_s + 1$ *honest* supervisors in $\mathcal{W}$. On the other hand, even in an *asynchronous* network, there will be at least $n - t_s - t_a > 2t_s$ *honest* supervisors in $\mathcal{W}$. Hence if a *corrupt* D is *not* discarded, then it is guaranteed that D has shared multiplication-triples.

Protocol Π_{TripSh}

- **Phase I — Sharing Triples and Verification-Triples (Time T_{ACS}):**
  - D selects $2t_s + 1$ random multiplication-triples $\{(x^{(j)}, y^{(j)}, z^{(j)})\}_{j=1,\ldots,2t_s+1}$. It then selects random t_s-degree polynomials $\{f_{x^{(j)}}(\cdot), f_{y^{(j)}}(\cdot), f_{z^{(j)}}(\cdot)\}_{j=1,\ldots,2t_s+1}$, such that $f_{x^{(j)}}(0) = x^{(j)}$, $f_{y^{(j)}}(0) = y^{(j)}$ and $f_{z^{(j)}}(0) = z^{(j)}$. D then invokes an instance of Π_{VSS} with input $\{f_{x^{(j)}}(\cdot), f_{y^{(j)}}(\cdot), f_{z^{(j)}}(\cdot)\}_{j=1,\ldots,2t_s+1}$ and the parties in $\mathcal{P}$ participate in this instance.

- In parallel, each party $P_i \in \mathcal{P}$ randomly selects a *verification multiplication-triple* $(u^{(i)}, v^{(i)}, w^{(i)})$ and random t_s-degree polynomials $f_{u^{(i)}}(\cdot), f_{v^{(i)}}(\cdot)$ and $f_{w^{(i)}}(\cdot)$ where $f_{u^{(i)}}(0) = u^{(i)}$, $f_{v^{(i)}}(0) = v^{(i)}$ and $f_{w^{(i)}}(0) = w^{(i)}$. With these polynomials as inputs, P_i participates in an instance of Π_{ACS} and waits for time T_{ACS}. Let $\mathcal{W}$ be the set of parties obtained as the output from Π_{ACS} where $|\mathcal{W}| \geq n - t_s$.

- **Phase II — Transforming** D**'s Triples (Time Δ)**:
  - Upon receiving an output in the instance of Π_{VSS} invoked by D, the parties participate in an instance $\Pi_{\mathsf{TripTrans}}(t_s, \{[x^{(j)}], [y^{(j)}], [z^{(j)}]\}_{j=1,\ldots,2t_s+1})$ of $\Pi_{\mathsf{TripTrans}}$.
  - Let $\{[\mathbf{x}^{(j)}], [\mathbf{y}^{(j)}], [\mathbf{z}^{(j)}]\}_{j=1,\ldots,2t_s+1})$ be the set of t_s-shared triples obtained from $\Pi_{\mathsf{TripTrans}}$ and let $\mathsf{X}(\cdot)$ and $\mathsf{Y}(\cdot)$ be the t_s-degree polynomials and $\mathsf{Z}(\cdot)$ be the $2t_s$-degree polynomial, which are guaranteed to be obtained from $\Pi_{\mathsf{TripTrans}}$, such that $\mathsf{X}(\alpha_j) = \mathbf{x}^{(j)}$, $\mathsf{Y}(\alpha_j) = \mathbf{y}^{(j)}$ and $\mathsf{Z}(\alpha_j) = \mathbf{z}^{(j)}$, for $j = 1, \ldots, 2t_s + 1$.
  - For $j = 2t_s + 2, \ldots, n$, the parties locally compute $[\mathbf{x}^{(j)}] = [\mathsf{X}(\alpha_j)]$ from $\{[\mathbf{x}^{(j)}]\}_{j=1,\ldots,t_s+1}$, $[\mathbf{y}^{(j)}] = [\mathsf{Y}(\alpha_j)]$ from $\{[\mathbf{y}^{(j)}]\}_{j=1,\ldots,t_s+1}$ and $[\mathbf{z}^{(j)}] = [\mathsf{Z}(\alpha_j)]$ from $\{[\mathbf{z}^{(j)}]\}_{j=1,\ldots,2t_s+1}$ respectively by using appropriate Lagrange's linear functions.
- **Phase III — Verifying Transformed Triples (Maximum Time 3Δ)**: The parties do the following.
  - **Phase III(a) — Recomputing the Products (Time Δ)**:
    - Corresponding to each $P_j \in \mathcal{W}$, participate in the instance $\Pi_{\mathsf{Beaver}}(([\mathbf{x}^{(j)}], [\mathbf{y}^{(j)}]), ([u^{(j)}], [v^{(j)}], [w^{(j)}]))$ of Π_{Beaver} to obtain $[\mathfrak{z}^{(j)}]$.
  - **Phase III(b) — Computing the Differences and Publicly Reconstructing (Time Δ)**:
    - Corresponding to every $P_j \in \mathcal{W}$, the parties locally compute $[\gamma^{(j)}] = [\mathbf{z}^{(j)}] - [\mathfrak{z}^{(j)}]$.
    - Corresponding to every $P_j \in \mathcal{W}$, the parties publicly reconstruct $\gamma^{(j)}$ by exchanging their respective shares of $\gamma^{(j)}$, followed by applying the $\mathsf{OEC}(t_s, t_s, \mathcal{P})$ procedure on the received shares.
    - Corresponding to $P_j \in \mathcal{W}$, party $P_i \in \mathcal{P}$ upon reconstructing $\gamma^{(j)}$, sets a Boolean variable $\mathsf{flag}_i^{(j)}$ to 0 if $\gamma^{(j)} = 0$, else it sets $\mathsf{flag}_i^{(j)}$ it to 1.
  - **Phase III(c) — Checking the Suspected Triples (Time Δ)**: Each $P_i \in \mathcal{P}$ does the following.
    - For every $P_j \in \mathcal{W}$ such that $\mathsf{flag}_i^{(j)} = 1$, send the shares corresponding to $[\mathbf{x}^{(j)}], [\mathbf{y}^{(j)}]$ and $[\mathbf{z}^{(j)}]$ to every party.
    - For every $P_j \in \mathcal{W}$ such that $\mathsf{flag}_i^{(j)} = 1$, apply the $\mathsf{OEC}(t_s, t_s, \mathcal{P})$ procedure on the received shares corresponding to $[\mathbf{x}^{(j)}], [\mathbf{y}^{(j)}]$ and $[\mathbf{z}^{(j)}]$, to reconstruct the triple $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$.
    - For every $P_j \in \mathcal{W}$, reset $\mathsf{flag}_i^{(j)}$ to 0 if $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$ is a multiplication-triple.
    - If $\mathsf{flag}_i^{(j)} = 0$, corresponding to every $P_j \in \mathcal{W}$, then set $\mathsf{flag}_i = 0$, else set $\mathsf{flag}_i = 1$.
- **Output Computation (Local Computation)**: Each party $P_i \in \mathcal{P}$ does the following.
  - If $\mathsf{flag}_i = 0$ then output shares corresponding to t_s-shared triple $([a], [b], [c])$ *on behalf of* D, where $a = \mathsf{X}(\beta)$, $b = \mathsf{Y}(\beta)$ and $c = \mathsf{Z}(\beta)$ and where $[a]$, $[b]$ and $[c]$ are locally computed from $\{[\mathbf{x}^{(j)}]\}_{j=1,\ldots,t_s+1}$, $\{[\mathbf{y}^{(j)}]\}_{j=1,\ldots,t_s+1}$ and $\{[\mathbf{z}^{(j)}]\}_{j=1,\ldots,2t_s+1}$ respectively by using appropriate Lagrange's linear functions. Here β is a non-zero element from $\mathbb{F}$, distinct from $\alpha_1, \ldots, \alpha_{2t_s+1}$.
  - If $\mathsf{flag}_i = 1$ then output default-shares (namely all shares being 0) corresponding to t_s-shared triple $([0], [0], [0])$ *on behalf of* D.

Figure 15: A protocol for verifiably sharing a single multiplication triple.

Lemma E.4. *Protocol* Π_{TripSh} *achieves the following properties.*

- **Correctness**: *If* D *is honest, then the following hold:*
  - *If the network is synchronous, then after time* $T_{\mathsf{TripSh}} = T_{\mathsf{ACS}} + 4\Delta$*, the honest parties have a* t_s*-shared multiplication-triple on behalf of* D.
  - *If the network is asynchronous, then almost-surely, the (honest) parties eventually have a* t_s*-shared multiplication-triple on behalf of* D.
- **Privacy**: *If* D *is honest, then the view of adversary remains independent of the multiplication-triple shared on behalf of* D.

- **Strong-Commitment**: *If D is corrupt, then the following hold:*
  - *No honest party obtains any output or depending upon the network type, the following hold:*
    - *Synchronous Network: The (honest) parties eventually output a t_s-shared multiplication-triple on behalf of D. Moreover, if some honest party has its output share at time T, then by time $T + 2\Delta$, all honest parties will have their respective output shares.*
    - *Asynchronous Network: The (honest) parties eventually output a t_s-shared multiplication-triple on behalf of D.*
- **Communication Complexity**: *The protocol incurs a communication of $\mathcal{O}(n^6 \log |\mathbb{F}|)$ bits and invokes $\mathcal{O}(n^2)$ instances of Π_{BA}.*

Proof. We first consider an *honest* D and prove the corresponding properties. We first consider a *synchronous* network with up to t_s corruptions. At time T_{VSS}, the multiplication-triples $\{(x^{(j)}, y^{(j)}, z^{(j)})\}_{j=1,\ldots,2t_s+1}$ will be t_s-shared. This follows from the *correctness* property of Π_{VSS} in *synchronous* network (see Theorem 4.2). Moreover, these triples will be random from the point of view of the adversary, which follows from the *privacy* property of Π_{VSS} (see Theorem 4.2). Since the instance of Π_{ACS} is invoked in parallel with the instance of Π_{VSS} invoked by D, at time T_{ACS}, all honest parties will have a common subset $\mathcal{W}$ from the instance of Π_{ACS}, with *every honest P_j* being present in the $\mathcal{W}$. This follows from the *correctness* property of Π_{ACS} in *synchronous* network (see Lemma E.3). At time $T_{\mathsf{ACS}} + \Delta$, the multiplication-triples shared by D will be transformed and parties will have t_s-shared multiplication-triples $\{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})\}_{j=1,\ldots,2t_s+1}$ and there will exist t_s-degree polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $2t_s$-degree polynomial $\mathsf{Z}(\cdot)$ where $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds. This follows from the *correctness* property of $\Pi_{\mathsf{TripTrans}}$ in a *synchronous* network (see Lemma E.2).

Next, corresponding to *every honest* $P_j \in \mathcal{W}$, the value $\mathfrak{z}^{(j)}$ will be the same as $\mathbf{x}^{(j)} \cdot \mathbf{y}^{(j)}$, which follows from the *correctness* of Π_{Beaver} and the fact that the corresponding verification-triple $(u^{(j)}, v^{(j)}, w^{(j)})$ will be a multiplication-triple. Hence, $\gamma^{(j)} = \mathbf{z}^{(j)} - \mathfrak{z}^{(j)}$ will be 0 and so each *honest* P_i will set $\mathsf{flag}_i^{(j)}$ to 0, without suspecting and reconstructing the triple $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$. Moreover, in this case, no additional information about $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$ is revealed, which follows from the *privacy* property of Π_{Beaver} and the fact that the verification-triple $(u^{(j)}, v^{(j)}, w^{(j)})$ remains random from the point of view of the adversary. On the other hand, if $P_j \in \mathcal{W}$ is *corrupt*, then $\gamma^{(j)}$ may not be 0. However, in this case each *honest* P_i will reset $\mathsf{flag}_i^{(j)}$ to 0 after reconstructing the corresponding suspected-triple $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$, since it will be a multiplication-triple. The process of computing $\mathfrak{z}^{(j)}$ and the difference $\gamma^{(j)}$ will take 2Δ time and additionally Δ time might be required to publicly reconstruct suspected-triples corresponding to *corrupt* $P_j \in \mathcal{W}$. Hence, at time $T_{\mathsf{ACS}} + 4\Delta$, each *honest* P_i sets $\mathsf{flag}_i = 1$ and hence, the honest parties output t_s-shared triple (a, b, c). Moreover, the triple will be a multiplication-triple, since (a, b, c) is the same as $(\mathsf{X}(\beta), \mathsf{Y}(\beta), \mathsf{Z}(\beta))$. Since at most t_s triples $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$ may be publicly reconstructed corresponding to the *corrupt* parties $P_j \in \mathcal{W}$, it follows that adversary will learn at most t_s distinct points on the $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$ polynomials. This further implies that $(\mathsf{X}(\beta), \mathsf{Y}(\beta), \mathsf{Z}(\beta))$ will be random from the point of view of the adversary, since $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ are t_s-degree polynomials and $\mathsf{Z}(\cdot)$ is a $2t_s$-degree polynomial. This completes the proof of the *correctness* and *privacy* properties in a *synchronous* network.

If D is *honest* and the network is *asynchronous* with up to t_a corruptions, then the proof of the *correctness* property is similar to the above proof, except that now we now use the *correctness* properties of $\Pi_{\mathsf{VSS}}, \Pi_{\mathsf{ACS}}, \Pi_{\mathsf{Beaver}}$ in an *asynchronous* network. Moreover, the privacy property holds since adversary now corrupt $t_a < t_s$ parties.

We next consider a *corrupt* D and prove the strong-commitment property. We first consider a *synchronous* network with up to t_s corruptions. Note that irrespective of whether D shares any triples through instance of Π_{VSS} or not, the instance of Π_{ACS} will produce a $\mathcal{W}$ at time T_{ACS}, with *every honest P_j* being present in the $\mathcal{W}$, which follows from the *correctness* property of Π_{ACS} in *synchronous* network (see Lemma E.3). If no honest party obtains any output in the protocol, then strong-commitment holds trivially. So consider the case when some honest party obtains an output. This implies that at least one *honest* party, say P_h must have received an output during the instance of Π_{VSS} invoked by D, as otherwise no honest party obtains any output in the protocol. Let T be the time at which P_h has the output for the instance of Π_{VSS} invoked by D. Note that $T \geq T_{\mathsf{VSS}}$ and T could be greater than T_{ACS}, as a *corrupt* D may delay the start of the instances of Π_{VSS}. From the *strong-Commitment* property of Π_{VSS} in *synchronous* network (see Theorem 4.2), it then follows that by time $T + 2\Delta$, *all* honest parties obtain their output in the instance of Π_{VSS} invoked by D. Hence at time $T + 2\Delta$, there are $2t_s + 1$ triples which are t_s-shared by D.

If $T \leq T_{\mathsf{ACS}} - 2\Delta$, then at time T_{ACS}, all honest parties will hold their respective shares corresponding to the t_s-shared triples of D, as well as the set $\mathcal{W}$ and shares corresponding to the verification-triples shared by the parties in $\mathcal{W}$. The instance of $\Pi_{\mathsf{TripTrans}}$ will produce its output at time $T_{\mathsf{ACS}} + \Delta$. The follow up instances of Π_{Beaver} to recompute the products will take Δ time, followed by Δ time for publicly reconstructing the difference values $\gamma^{(j)}$. Additionally, the parties may take Δ time to publicly reconstruct any suspected triples. Hence in this case, *all honest* parties will have their respective output shares at time $T_{\mathsf{ACS}} + 4\Delta$.

On the other hand, if $T > T_{\mathsf{ACS}} - 2\Delta$, then each honest party obtains its output from the instance of $\Pi_{\mathsf{TripTrans}}$, either at time $T + \Delta$ or at time $T + 3\Delta$. Then, each honest party obtains its output from the instances of Π_{Beaver}, either at time $T + 2\Delta$ or at time $T + 4\Delta$. This implies that difference values $\gamma^{(j)}$ are available with the honest parties, either at time $T + 3\Delta$ or $T + 5\Delta$. Consequently, the suspected triples (if any) will be available with the honest parties, either at time $T + 4\Delta$ or $T + 6\Delta$. Hence each honest party obtains its output share in the protocol either at time $T + 4\Delta$ or $T + 6\Delta$. Notice that in this case there might be a difference of at most 2Δ time within which the honest parties obtain their output in the protocol, due to a possible difference of 2Δ time in getting the output in the instances of Π_{VSS} invoked by the *corrupt* D.

If the triples shared by D during the instances of Π_{VSS} are all *multiplication-triples*, then similar to the proof of the correctness property for an *honest* D, it follows that the honest parties will output a t_s-shared multiplication-triple on behalf of D. So consider the case when all the triples shared by D are *not* multiplication-triples. This implies that $\mathsf{Z}(\cdot) \neq \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$, where $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ are the t_s-degree polynomials and $\mathsf{Z}(\cdot)$ is the $2t_s$-degree polynomial, which are guaranteed to exist from the protocol $\Pi_{\mathsf{TripTrans}}$. Let P_j be an *honest* party, such that $\mathsf{Z}(\alpha_j) \neq \mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j)$. This further implies that the transformed triple $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$ is *not* a multiplication-triple. Such a P_j is bound to exist. This is because there are at least $2t_s + 1$ honest parties P_j. And if $\mathsf{Z}(\alpha_j) = \mathsf{X}(\alpha_j) \cdot \mathsf{Y}(\alpha_j)$ holds corresponding to *every* honest P_j, then it implies that $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds (due to the degrees of the respective polynomials), which is a contradiction.

We next show that each *honest* P_i will set $\mathsf{flag}_i^{(j)} = 1$ and hence $\mathsf{flag}_i = 1$. For this, we note that $P_j \in \mathcal{W}$. This follows from the *correctness* property of Π_{ACS}, which guarantees that *all* honest parties (and not just P_j) will be present in $\mathcal{W}$. Since the verification-triple $(u^{(j)}, v^{(j)}, w^{(j)})$ shared by P_j will be a multiplication-triple, from the *correctness* property of Π_{Beaver} in *synchronous* network, it follows that $\mathfrak{z}^{(j)} = \mathbf{x}^{(j)} \cdot \mathbf{y}^{(j)}$ holds. But since $\mathbf{z}^{(j)} \neq \mathbf{x}^{(j)} \cdot \mathbf{y}^{(j)}$, it follows that $\gamma^{(j)} = \mathbf{z}^{(j)} - \mathfrak{z}^{(j)} \neq 0$. Consequently, the parties will publicly reconstruct the suspected-triple $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)})$ and find that it is not a multiplication-triple. Hence each *honest* P_i will set $\mathsf{flag}_i^{(j)} = 1$ and hence $\mathsf{flag}_i = 1$. So the parties output a default t_s-sharing of the multiplication-triple $(0, 0, 0)$ on behalf of D.

The proof for the *strong-commitment* property in the *asynchronous* network is similar to the above proof, except that we now use the *strong-Commitment* property of Π_{VSS} and *correctness* properties of Π_{ACS} and Π_{Beaver} in *asynchronous* network. Moreover, there will be at least $n - t_s - t_a \geq 2t_s + 1$ honest parties in $\mathcal{W}$, who will lead the verification of at least $2t_s + 1$ distinct points on the polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$, through their respective verification-triples.

The communication complexity follows from communication complexity of $\Pi_{\mathsf{ACS}}, \Pi_{\mathsf{VSS}}, \Pi_{\mathsf{TripTrans}}$ and Π_{Beaver}. □

Protocol Π_{TripSh} for Sharing L Multiplication-Triples: Protocol Π_{TripSh} can be easily generalized so that L multiplication-triples are shared on behalf of D. Namely, D now has to share $L \cdot (2t_s + 1)$ random multiplication-triples through Π_{VSS}, while each party P_j will need to select L verification-triples during the instance of Π_{ACS}. Moreover, there will be L instances of $\Pi_{\mathsf{TripTrans}}$ to transform D's shared triples, resulting in L triplets of shared polynomials $(\mathsf{X}(\cdot), \mathsf{Y}(\cdot), \mathsf{Z}(\cdot))$, each of which is independently verified by performing supervised verification. To avoid repetition, we do not provide the formal details. The modified Π_{TripSh} protocol will incur a communication of $\mathcal{O}(n^4 L \log |\mathbb{F}| + n^6 \log |\mathbb{F}|)$ bits and invokes $\mathcal{O}(n^2)$ instances of Π_{BA}.

E.1.5 The Triple-Extraction Protocol

Protocol Π_{TripExt} (Fig 16) takes as input a *publicly-known* subset $\mathcal{CS}$ of $2d + 1$ parties, where $d \geq t_s$ and where it will be *ensured* that each party $P_j \in \mathcal{CS}$ has t_s-shared a multiplication-triple. It will also be ensured that if P_j is *honest*, then the multiplication-triple is random from the point of view of the adversary. The protocol outputs $d + 1 - t_s$ number of t_s-shared multiplication-triples, which will be random from the point of view of the adversary. The high level idea of the protocol is very simple. The parties first invoke an instance of $\Pi_{\mathsf{TripTrans}}$ to "transform" the input triples into a set of co-related triples. Since all the input triples are multiplication-triples, the output triples will also be multiplication-triples. Let $(\mathsf{X}(\cdot), \mathsf{Y}(\cdot), \mathsf{Z}(\cdot))$ be the triplet of shared polynomials which is guaranteed to exist after $\Pi_{\mathsf{TripTrans}}$. From the *privacy* property of $\Pi_{\mathsf{TripTrans}}$, it follows that adversary will know at most t_s distinct points on these polynomials and hence at least $d+1-t_s$ points on these polynomials are random for the adversary. Hence, the parties output $d+1-t_s$ "new" points on these polynomials (in a t_s-shared fashion), which are guaranteed to be random from the point of view of the adversary. This requires the parties to perform *only* local computation.

Protocol $\Pi_{\mathsf{TripExt}}(\mathcal{CS}, \{[x^{(j)}], [y^{(j)}], [z^{(j)}]\}_{P_j \in \mathcal{CS}})$

- **Transforming the Input Multiplication-Triples (Time Δ)** — The parties jointly do the following:
  - Participate in an instance $\Pi_{\mathsf{TripTrans}}(d, \{[x^{(j)}], [y^{(j)}], [z^{(j)}]\}_{P_j \in \mathcal{CS}})$ of $\Pi_{\mathsf{TripTrans}}$.
    - Let $\{[\mathbf{x}^{(j)}], [\mathbf{y}^{(j)}], [\mathbf{z}^{(j)}]\}_{P_j \in \mathcal{CS}}$ be the shared multiplication-triples obtained from $\Pi_{\mathsf{TripTrans}}$. Moreover, let $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ be the d-degree polynomials and $\mathsf{Z}(\cdot)$ be the $2d$-degree polynomial where $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ and where $\mathsf{X}(\alpha_j) = \mathbf{x}^{(j)}$, $\mathsf{Y}(\alpha_j) = \mathbf{y}^{(j)}$ and $\mathsf{Z}(\alpha_j) = \mathbf{z}^{(j)}$ holds corresponding to every $P_j \in \mathcal{CS}$.
    - For $j = 1, \ldots, d + 1 - t_s$, locally compute $[\mathbf{a}^{(j)}]$, $[\mathbf{b}^{(j)}]$ and $[\mathbf{c}^{(j)}]$ from $\{[\mathbf{x}^{(j)}]\}_{j=1,\ldots,d+1}$, $[\{\mathbf{y}^{(j)}]\}_{j=1,\ldots,d+1}$ and $\{[\mathbf{z}^{(j)}]\}_{j=1,\ldots,2d+1}$ respectively by applying the corresponding Lagrange's linear function. Here $\mathbf{a}^{(j)} = \mathsf{X}(\beta_j)$, $\mathbf{b}^{(j)} = \mathsf{Y}(\beta_j)$ and $\mathbf{c}^{(j)} = \mathsf{Z}(\beta_j)$, where $\beta_1, \ldots, \beta_{d+1-t_s}$ are distinct, non-zero elements from $\mathbb{F}$, different from $\alpha_1, \ldots, \alpha_n$.
    - Output $\{[\mathbf{a}^{(j)}], [\mathbf{b}^{(j)}], [\mathbf{c}^{(j)}]\}_{j=1,\ldots,d+1-t_s}$.

Figure 16: Protocol for extracting $d + 1 - t_s$ random t_s-shared random multiplication-triples from a set of $|\mathcal{CS}| = 2d + 1$ t_s-shared multiplication triples, where $d \geq t_s$.

Lemma E.5. *Let $\mathcal{CS}$ be a set of $2d + 1$ parties where $d \geq t_s$, such that each party $P_j \in \mathcal{CS}$ has a multiplication-triple $(x^{(j)}, y^{(j)}, z^{(j)})$ which is t_s-shared. Moreover, if P_j is honest, then the multiplication-triple is random from the point of view of the adversary. Then protocol Π_{TripExt} achieves the following properties.*

- **Correctness**: *If the network is synchronous, then after time Δ the parties have t_s-shared multiplication-triples $\{\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}\}_{j=1,\ldots,d+1-t_s}$. If the network is asynchronous, then the parties eventually have t_s-shared multiplication-triples $\{\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}\}_{j=1,\ldots,d+1-t_s}$.*
- **Privacy**: *The triples $\{\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}\}_{j=1,\ldots,d+1-t_s}$ will be random from the point of view of the adversary.*
- **Communication Complexity**: *The protocol incurs a communication of $\mathcal{O}(dn^2 \log |\mathbb{F}|)$ bits.*

Proof. If the network is *synchronous*, then from the *correctness* property of $\Pi_{\mathsf{TripTrans}}$ in the *synchronous* network (see Lemma E.2), it follows that after time Δ, the honest parties have t_s-shared triples $\{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)}\}_{P_j \in \mathcal{CS}}$. Moreover, all these triples will be multiplication-triples, since all the input triples $\{x^{(j)}, y^{(j)}, z^{(j)}\}_{P_j \in \mathcal{CS}}$ are guaranteed to be multiplication-triples. This further implies that the condition $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds, where $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$ are the d, d and $2d$-degree polynomials respectively, which are guaranteed to exist from $\Pi_{\mathsf{TripTrans}}$, such that $\mathsf{X}(\alpha_j) = \mathbf{x}^{(j)}$, $\mathsf{Y}(\alpha_j) = \mathbf{y}^{(j)}$ and $\mathsf{Z}(\alpha_j) = \mathbf{z}^{(j)}$ holds for every j, such that $P_j \in \mathcal{CS}$. It now follows that the honest parties output the t_s-shared triples $\{(\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)})\}_{j=1,\ldots,d+1-t_s}$ after time Δ, where $\mathsf{X}(\beta_j) = \mathbf{a}^{(j)}$, $\mathsf{Y}(\beta_j) = \mathbf{b}^{(j)}$ and $\mathsf{Z}(\beta_j) = \mathbf{c}^{(j)}$. Moreover, the triples will be multiplication-triples, because $\mathsf{Z}(\cdot) = \mathsf{X}(\cdot) \cdot \mathsf{Y}(\cdot)$ holds.

If the network is *asynchronous*, then the proof of *correctness* property will be similar as above, except that we now depend upon the *correctness* property of $\Pi_{\mathsf{TripTrans}}$ in the *asynchronous* network.

For *privacy*, we note that there will be at most t_s *corrupt* parties in $\mathcal{CS}$ and hence adversary will know at most t_s multiplication-triples in the set $\{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)}\}_{P_j \in \mathcal{CS}}$, which follows from the *privacy* property of $\Pi_{\mathsf{TripTrans}}$. This implies that adversary will know at most t_s distinct points on the polynomials $\mathsf{X}(\cdot), \mathsf{Y}(\cdot)$ and $\mathsf{Z}(\cdot)$, leaving $d + 1 - t_s$ degrees of freedom on the these polynomials. This further implies that the multiplication-triples $\{(\mathsf{X}(\beta_j), \mathsf{Y}(\beta_j), \mathsf{Z}(\beta_j))\}_{j=1,\ldots,d+1-t_s}$, which are the same as $\{\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}\}_{j=1,\ldots,d+1-t_s}$, will be random from the point of view of the adversary. Namely, there will be a one-to-one correspondence between the $d+1-t_s$ multiplication-triples in the set $\{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{z}^{(j)}\}_{P_j \in \mathcal{CS}}$ which are *unknown* to the adversary and the output multiplication-triples $\{\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}\}_{j=1,\ldots,d+1-t_s}$. Hence adversary's view will be consistent with every candidate value of the output $d + 1 - t_s$ multiplication-triples.

The *communication complexity* simply follows from the fact that the protocol requires one instance of $\Pi_{\mathsf{TripTrans}}$. □

E.2 The Triple-Generation Protocol

The triple-generation protocol Π_{TripGen} is presented in Fig 17. In the protocol, each party acts as a dealer and invokes an instance of Π_{TripSh}, so that $L = \frac{c_M}{(\frac{n-t_s-1}{2}+1-t_s)}$ random multiplication-triples are shared on its behalf. As *corrupt* dealers may not invoke their instances of Π_{TripSh} (even in a *synchronous* network), the parties agree on a *common* subset $\mathcal{CS}$ of $n - t_s$ parties, who have shared multiplication-triples, by executing instances of Π_{BA} (similar to the protocol Π_{ACS}). The multiplication-triples shared on the behalf of up to t_s *corrupt* triple-providers in $\mathcal{CS}$ will be known to adversary, while the multiplication-triples shared on the behalf of the *honest* triple-providers in $\mathcal{CS}$ will be random for the adversary. Since the exact identity of the *honest* triple-providers in $\mathcal{CS}$ is not known, the parties execute L instances of Π_{TripExt} to securely extract shared multiplication-triples, which will be random for the adversary. In the protocol, we assume for the sake of simplicity that $n - t_s$ is of the form $2d + 1$.

Protocol Π_{TripGen}

Let $L \overset{def}{=} \frac{c_M}{(\frac{n-t_s-1}{2}+1-t_s)}$.

- **Phase 1 — Sharing Random Multiplication-Triple (Time R_{TripSh})**: Each $P_i \in \mathcal{P}$ does the following.
  - Act as a dealer D and invoke an instance $\Pi^{(i)}_{\mathsf{TripSh}}$ of Π_{TripSh} so that L random multiplication-triples are shared on P_i's behalf.
  - For $j = 1, \ldots, n$, participate in the instance $\Pi^{(j)}_{\mathsf{TripSh}}$ invoked by P_j and wait for time T_{TripSh}.
  - Initialize a set $\mathcal{C}_i = \emptyset$ and include P_j in $\mathcal{C}_i$, if an output is obtained from $\Pi^{(j)}_{\mathsf{TripSh}}$.
- **Phase II — Agreement on a Common Subset of Triple-Providers (Maximum time $2T_{\mathsf{BA}}$)**: Each $P_i \in \mathcal{P}$ does the following.
  - For $j = 1, \ldots, n$, participate in an instance of $\Pi^{(j)}_{\mathsf{BA}}$ of Π_{BA} with input 1, if $P_j \in \mathcal{C}_i$.
  - Once $n - t_s$ instances of Π_{BA} have produced an output 1, then participate with input 0 in all the Π_{BA} instances $\Pi^{(j)}_{\mathsf{BA}}$ such that $P_j \notin \mathcal{C}_i$.
  - Once all the n instances of Π_{BA} have produced a binary output, then set $\mathcal{CS}$ to be the set of *first* $n - t_s$ parties P_j, such that 1 is obtained as the output in the instance $\Pi^{(j)}_{\mathsf{BA}}$.
- **Phase III — Extracting Random Multiplication-Triples (Time Δ)**: The parties do the following.
  - For every $P_j \in \mathcal{CS}$, let $\{[x^{(j,\ell)}], [y^{(j,\ell)}], [z^{(j,\ell)}]\}_{\ell=1,\ldots,L}$ be the t_s-shared multiplication-triples, shared on P_j's behalf.
  - For $\ell = 1, \ldots, L$, the parties participate in an instance $\Pi_{\mathsf{TripExt}}(\mathcal{CS}, \{[x^{(j,\ell)}], [y^{(j,\ell)}], [z^{(j,\ell)}]\}_{P_j \in \mathcal{CS}})$ of Π_{TripExt} to obtain the output $\{[\mathbf{a}^{(j,\ell)}], [\mathbf{b}^{(j,\ell)}], [\mathbf{c}^{(j,\ell)}]\}_{j=\frac{n-t_s-1}{2}+1-t_s}$.
  - Output the shared triples $\{[\mathbf{a}^{(j,\ell)}], [\mathbf{b}^{(j,\ell)}], [\mathbf{c}^{(j,\ell)}]\}_{j=\frac{n-t_s-1}{2}+1-t_s, \ell=1,\ldots,L}$.

Figure 17: The best-of-both-worlds protocol for generating shared random multiplication-triples.

Theorem E.6. *Protocol Π_{TripGen} achieves the following properties.*

- **Correctness**:
  - *In a synchronous network, by time $T_{\mathsf{TripGen}} = T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}} + \Delta$, the honest parties output a t_s-sharing of c_M multiplication-triples.*
  - *In an asynchronous network, almost-surely, the honest parties eventually output a t_s-sharing of c_M multiplication-triples.*
- **Privacy**: *The view of the adversary remains independent of the output multiplication-triples.*
- **Communication Complexity**: *The protocol incurs a communication cost of $\mathcal{O}(\frac{n^5}{\frac{t_a}{2}+1} c_M \log|\mathbb{F}| + n^7 \log|\mathbb{F}|)$ bits and invokes $\mathcal{O}(n^3)$ instances of Π_{BA}. For $t_a = \Omega(n)$, the protocol incurs a communication cost of $\mathcal{O}(n^4 c_M \log|\mathbb{F}| + n^7 \log|\mathbb{F}|)$, along with $\mathcal{O}(n^3)$ instances of Π_{BA}.*

Proof. Let the network be *synchronous* with up to t_s corruptions. Let $\mathcal{H}$ be the set of parties, where $|\mathcal{H}| \geq n - t_s$. Corresponding to each $P_j \in \mathcal{H}$, at time T_{TripSh}, L multiplication-triples $\{x^{(j,\ell)}, y^{(j,\ell)}, z^{(j,\ell)}\}_{\ell=1,\ldots,L}$ will be t_s-shared on behalf of P_j, which follows from the *correctness* of Π_{TripSh} in *synchronous* network (see Lemma E.4). Consequently, the set $\mathcal{C}_i$ will be of size at least $n - t_s$ for every honest P_i. After time T_{TripSh}, corresponding to each $P_j \in \mathcal{H}$, each honest P_i participates with input 1 in the instance $\Pi^{(j)}_{\mathsf{BA}}$. It then follows from the *validity* and *guaranteed liveness* of Π_{BA} in the *synchronous* network that corresponding to every $P_j \in \mathcal{H}$, every honest P_i obtains the output 1 in the instance $\Pi^{(j)}_{\mathsf{BA}}$ at time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$. Consequently, after time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$, every honest party will start participating in the remaining Π_{BA} instances for which no input has been provided yet (if there are any) and from the *guaranteed liveness* and *consistency* of Π_{BA} in the *synchronous* network, these Π_{BA} instances will produce common outputs for every honest party at time $T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}}$. Consequently, by time $T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}}$, every honest party has a common $\mathcal{CS}$ of size $n - t_s$.

Consider an *arbitrary* party $P_j \in \mathcal{CS}$. If P_j is *honest*, then as shown above, L multiplication-triples will be shared on behalf of P_j at time T_{TripSh}. Next consider a *corrupt* $P_j \in \mathcal{CS}$. Since $P_j \in \mathcal{CS}$, it follows that the instance $\Pi_{\mathsf{BA}}^{(j)}$ produces the output 1. This further implies that at least one *honest* P_i must have obtained some output from the instance $\Pi_{\mathsf{TripSh}}^{(j)}$ by time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$ (implying that $P_j \in \mathcal{C}_i$) and participated with input 1 in the instance $\Pi_{\mathsf{BA}}^{(j)}$. This is because if P_j *does not* belong to the $\mathcal{C}_i$ set of *any* honest P_i at time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$, then it implies that *all* honest parties participate with input 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ after time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$. And then from the *validity* of Π_{BA} in the *synchronous* network, every honest party would obtain the output 0 in the instance $\Pi_{\mathsf{BA}}^{(j)}$ and hence P_j will *not* be present in $\mathcal{CS}$, which is a contradiction. Now if P_i has obtained some output in $\Pi_{\mathsf{TripSh}}^{(j)}$ by time $T_{\mathsf{TripSh}} + T_{\mathsf{BA}}$, then from the *strong-commitment* of Π_{TripSh} in *synchronous* network (see Lemma E.4), it follows that there exist L multiplication-triples, say $\{(x^{(j,\ell)}, y^{(j,\ell)}, z^{(j,\ell)})\}_{\ell=1,\ldots,L}$, which will be t_s-shared among the parties on behalf of P_j by time $(T_{\mathsf{TripSh}} + T_{\mathsf{BA}} + 2\Delta) < (T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}})$; the latter follows because $2\Delta < T_{\mathsf{BA}}$.

From the above discussion, it follows that there will be L multiplication-triples, which will be t_s-shared on behalf of *each* $P_j \in \mathcal{CS}$ by time $T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}}$. Hence each instance of Π_{TripExt} will output $\frac{n-t_s-1}{2} + 1 - t_s$ number of t_s-shared multiplication-triples by time $T_{\mathsf{TripGen}} = T_{\mathsf{TripSh}} + 2T_{\mathsf{BA}} + \Delta$. This follows from the *correctness* property of Π_{TripExt} in *synchronous* network by substituting $|\mathcal{CS}| = n - t_s$ and $d = \frac{n-t_s-1}{2}$ in Lemma E.5. Since there are $L = \frac{c_M}{(\frac{n-t_s-1}{2}+1-t_s)}$ instances of Π_{TripExt}, it follows that at time T_{TripGen}, the parties have $L \cdot \left(\frac{n-t_s-1}{2}+1-t_s\right) = c_M$ number of t_s-shared multiplication-triples. This completes the proof of the *correctness* property in the *synchronous* network.

The proof of the *correctness* property in the *asynchronous* network is similar as above, except that we now use the *correctness* and *strong-commitment* properties of Π_{TripSh} in *asynchronous* network, the *correctness* property of Π_{TripExt} in *asynchronous* network and properties of Π_{BA} in *asynchronous* network.

From the *privacy* property of Π_{TripSh}, it follows that the multiplication-triples which are t_s-shared on behalf of the *honest* parties $P_j \in \mathcal{CS}$ will be random from the point of view of the adversary. It then follows from the *privacy* property of Π_{TripExt} (Lemma E.4) that the t_s-shared multiplication-triples generated from each instance of Π_{TripExt} will be random from the point of view of the adversary. This proves the *privacy* property.

The *communication complexity* follows from the communication complexity of Π_{TripSh} and Π_{TripExt}, and because $\frac{n-t_s-1}{2} + 1 - t_s \geq \frac{t_a}{2} + 1$. □

F Properties of the Circuit-Evaluation Protocol

Protocol Π_{CirEval} for securely evaluating the circuit cir is presented in Fig 18.

Protocol Π_{CirEval}

- **Preprocessing and Input-Sharing (Time T_{TripGen})** — The parties do the following:
  - Each $P_i \in \mathcal{P}$ on having the input $x^{(i)}$ for f, select a random t_s-degree polynomial $f_{x^{(i)}}(\cdot)$ where $f_{x^{(i)}}(0) = x^{(i)}$ and participates in an instance of Π_{ACS} with input $f_{x^{(i)}}(\cdot)$. Let $\mathcal{CS}$ be the common subset of parties obtained from Π_{ACS}, where $|\mathcal{CS}| \geq n - t_s$. Corresponding to every $P_j \notin \mathcal{CS}$, set $x^{(j)} = 0$ and set $[x^{(j)}]$ to a default t_s-sharing of 0.
  - In parallel, participate in an instance of Π_{TripGen}. Let $\{[\mathbf{a}^{(j)}], [\mathbf{b}^{(j)}], [\mathbf{c}^{(j)}]\}_{j=1,\ldots,c_M}$ be the t_s-shared multiplication-triples obtained from Π_{TripGen}.
- **Circuit Evaluation (Time $D_M \cdot \Delta$)** — Let $G_1, \ldots, G_m$ be a publicly-known topological ordering of the gates of cir. For $k = 1, \ldots, m$, the parties do the following for gate G_k:
  - *If G_k is an addition gate*: the parties locally compute $[w] = [u] + [v]$, where u and v are gate-inputs and w is the gate-output.
  - *If G_k is a multiplication-with-a-constant gate with constant c*: the parties locally compute

$[v] = c \cdot [u]$, where u is the gate-input and v is the gate-output.

  – *If G_k is an addition-with-a-constant gate with constant c*: the parties locally compute $[v] = c + [u]$, where u is the gate-input. and v is the gate-output.
  – *If G_k is a multiplication gate*: Let G_k be the ℓ^{th} multiplication gate in cir where $\ell \in \{1, \ldots, c_M\}$ and let $([a^{(\ell)}], [b^{(\ell)}], [c^{(\ell)}])$ be the ℓ^{th} shared multiplication-triple, generated from Π_{TripGen}. Moreover, let $[u]$ and $[v]$ be the shared gate-inputs of G_k. Then the parties participate in an instance $\Pi_{\mathsf{Beaver}}(([u], [v]), ([a^{(\ell)}], [b^{(\ell)}], [c^{(\ell)}]))$ of Π_{Beaver} and obtain $[w]$.

– **Output Computation (Time Δ)** — Let $[y]$ be the t_s-shared circuit-output. The parties exchange their respective shares of y and apply the $\mathsf{OEC}(t_s, t_s, \mathcal{P})$ procedure on the received shares to reconstruct y.

– **Termination (Time Δ)**: Each P_i does the following.
  – If y has been obtained during output computation, then send $(\texttt{ready}, y)$ message to all the parties.
  – If the message $(\texttt{ready}, y)$ is received from at least $t_s + 1$ distinct parties, then send $(\texttt{ready}, y)$ message to all the parties if not sent earlier.
  – If the message $(\texttt{ready}, y)$ is received from at least $2t_s + 1$ distinct parties, then terminate all sub-protocols, output y and terminate.

Figure 18: A best-of-both-worlds perfectly-secure protocol for securely evaluating the arithmetic circuit cir.

Theorem 6.1. *Let $t_a < t_s$, such that $3t_s + t_a < n$. Moreover, let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function represented by an arithmetic circuit cir over $\mathbb{F}$ consisting of c_M number of multiplication gates, and whose multiplicative depth is D_M. Then, Π_{CirEval} incurs a communication cost of $\mathcal{O}(\frac{n^5}{\frac{t_a}{2}+1} c_M \log |\mathbb{F}| + n^7 \log |\mathbb{F}|)$ bits (which is $\mathcal{O}(n^4 c_M \log |\mathbb{F}| + n^7 \log |\mathbb{F}|)$ bits if $t_a = \Omega(n)$), invokes $\mathcal{O}(n^3)$ instances of Π_{BA} and achieves the following.*

– *Correctness:* **(a)** *In a synchronous network, all honest parties output $y = f(x^{(1)}, \ldots, x^{(n)})$ at time $(120n + D_M + 6k - 20) \cdot \Delta$, where $x^{(j)} = 0$ for every $P_j \notin \mathcal{CS}$, such that $|\mathcal{CS}| \geq n - t_s$ and every honest $P_j \in \mathcal{CS}$; here k is the constant from Lemma 3.2, as determined by the underlying (existing) perfectly-secure ABA protocol Π_{ABA}.* **(b)** *In an asynchronous network, almost-surely, the honest parties eventually output $y = f(x^{(1)}, \ldots, x^{(n)})$ where $x^{(j)} = 0$ for every $P_j \notin \mathcal{CS}$ and where $|\mathcal{CS}| \geq n - t_s$.*

– *Privacy: The view of the adversary will be independent of the inputs of the honest parties in $\mathcal{CS}$.*

Proof. Consider a *synchronous* network with up to t_s corruptions. From the *correctness* property of Π_{TripGen} in the *synchronous* network (see Theorem E.6), at time T_{TripGen}, the (honest) parties have c_M number of t_s-shared multiplication-triples from the instance of Π_{TripGen}. From the *correctness* property of Π_{ACS} in the *synchronous* network (see Lemma E.3), at time T_{ACS}, the (honest) parties have a common subset $\mathcal{CS}$ from the instance of Π_{ACS}, where *all honest* parties will be present in $\mathcal{CS}$ and where $|\mathcal{CS}| \geq n - t_s$. Moreover, corresponding to *every* $P_j \in \mathcal{CS}$, there will be some $x^{(j)}$ available with P_j (which will be the same as P_j's input for f for an *honest* P_j), such that $x^{(j)}$ will be t_s-shared. As $\mathcal{CS}$ will be known *publicly*, the parties take a default t_s-sharing of 0 on the behalf of the parties outside $\mathcal{CS}$ by considering $x^{(j)} = 0$. Since $T_{\mathsf{ACS}} < T_{\mathsf{TripGen}}$, it follows that at time T_{TripGen}, the parties will hold t_s-sharing of c_M multiplication-triples and t_s-sharing of $x^{(1)}, \ldots, x^{(n)}$.

The circuit-evaluation will take $D_M \cdot \Delta$ time. This follows from the fact that linear gates are evaluated locally, while all the *independent* multiplication gates can be evaluated in parallel by running the corresponding instances of Π_{Beaver} in *parallel*, where each such instance requires Δ time. From the *correctness* property of Π_{Beaver} in the *synchronous* network (see Lemma E.1), the multiplication-gates will be evaluated correctly and hence, during the output-computation phase, the parties will hold a t_s-sharing of y, where $y = f(x^{(1)}, \ldots, x^{(n)})$. From the properties of OEC, it will take Δ time for every party to reconstruct y. Hence, during the termination phase, *all* honest parties will send a $\texttt{ready}$ message for y. Since there are at least $2t_s + 1$ honest parties, every honest party will then terminate with output y at time $T_{\mathsf{TripGen}} + (D_M + 2) \cdot \Delta$. By substituting the values of $T_{\mathsf{TripGen}}, T_{\mathsf{TripSh}}, T_{\mathsf{ACS}}, T_{\mathsf{VSS}}, T_{\mathsf{WPS}}, T_{\mathsf{BC}}, T_{\mathsf{BA}}, T_{\mathsf{SBA}}$ and T_{ABA} and by noting that all instances of Π_{BC} in Π_{CirEval} are invoked with $t = t_s$, we get that the parties terminate the protocol at time $(120n + D_M + 6k - 20) \cdot \Delta$, where k is the constant from Lemma 3.2, as determined by the underlying (existing) perfectly-secure ABA protocol Π_{ABA}.

The proof of the *correctness* property in an *asynchronous* network is similar as above, except that we now use the security properties of the building blocks $\Pi_{\mathsf{TripGen}}, \Pi_{\mathsf{ACS}}, \Pi_{\mathsf{Beaver}}$ and Π_{RecPriv} in the *asynchronous* network. During the termination phase, at most t_a *corrupt* parties can send `ready` messages for $y' \neq y$ and there will be at least $2t_s + 1$ honest parties, who eventually send `ready` messages for y. Moreover, if some honest party P_h terminates with output y, then every honest party eventually terminates the protocol with output y. This is because P_h must have received `ready` messages for y from at least $t_s + 1$ *honest* parties before termination, which are eventually delivered to *every* honest party. Consequently, irrespective of which stage of the protocol an honest party is in, every honest party (including P_h) eventually sends a `ready` message for y which are eventually delivered. As there are at least $2t_s + 1$ honest parties, this implies that every honest party eventually terminates with output y.

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

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