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 \( t < n/3 \) 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 $t_s < t_a$?

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 $F$, 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 $t_a$ cor-
ruptions. 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 $t_a$ 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 $\Delta$. 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 $\text{cir} \over \mathbb{F}$, consisting of linear and non-linear (multiplication) gates [33], where $\text{cir}$ 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, even after receiving their outputs. 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:

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

where the probability is over the random coins of the honest parties and the adversary in the protocol. The property of almost-surely obtaining 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 $f(x) = a_0 + \ldots + a_dx^d$, where each $a_i \in \mathbb{F}$. An $(\ell, \ell)$-degree symmetric bivariate polynomial is of the form $F(x, y) = \sum_{i,j=0}^{i+j=\ell} r_{ij}x_iy_j$, 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 $\alpha_i, \alpha_j$ and $F(x, \alpha_i) = F(\alpha_i, y)$. We say that a $\ell$-degree $F_i(x)$, where $i \in \{1, \ldots, n\}$, lies on a $(\ell, \ell)$-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 $\ell$-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 $(\ell, \ell)$-degree symmetric bivariate polynomial $F^*(x, y)$.  

- Let $\mathcal{C} \subseteq \mathcal{P}$ 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 $\{F(x, \alpha_i)\}_{P_i \in \mathcal{C}}$ and $\{F'(x, \alpha_i)\}_{P_i \in \mathcal{C}}$ 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 $([u_1]_d, \ldots, [v_m]_d) = g([u_1]_d, \ldots, [v_m]_d)$, we mean that parties locally apply the function $g$ on their respective shares of $u_1, \ldots, v_m$ 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 $d + t + 1$ 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 $\Delta$ time for $P_R$ to recover $q(\cdot)$, since the points of the honest parties will be delivered within $\Delta$ time. The procedure 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 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 $S \in \mathcal{P}$ 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 $\Pi_{\text{Acast}}$ 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 $m^*$, then every honest party eventually outputs $m^*$. 

---

5
Synchronous Network: (a) Liveness: If S is honest, then all honest parties obtain an output within time $3\Delta$. (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 $m^*$ at time $T$, then every honest $P_i$ outputs $m^*$ by the end of time $T + 2\Delta$.

Communication Complexity: $O(n^2 \ell)$ bits are communicated, where S’s message is of size $\ell$ bits.

Terminologies for Using $\Pi_{ACast}$: We will say that $P_i$ Acasts $m$ to mean that $P_i$ acts as a sender S and invokes an instance of $\Pi_{ACast}$ 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 $\Pi_{ACast}$.

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 $\Pi$ be a protocol for $P$, where every $P_i$ has input $b_i \in \{0, 1\}$ and a possible output from $\{0, 1, \bot\}$. $\Pi$ 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 $\bot$.
- Consistency: All honest parties with an output, output the same value (which can be $\bot$).
- WeakConsistency: All honest parties with an output, output either a common $v \in \{0, 1\}$ or $\bot$.

$\Pi$ 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, $\Pi$ 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 $\Pi_{ABA}$ such that:

- Asynchronous Network: The protocol is a $t$-perfectly-secure ABA protocol. If the inputs of all honest parties are same, then $\Pi_{ABA}$ 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 $\Pi_{ABA}$ achieves guaranteed liveness and all honest parties obtain their output within time $T_{ABA} = k \cdot \Delta$ for some constant $k$. Else it achieves almost-surely liveness and requires $O(poly(n) \cdot \Delta)$ expected time to generate the output.
  - Communication Complexity: $\Pi_{ABA}$ incurs a communication of $O(poly(n) \log |F|)$ bits if the inputs of all honest parties are the same. Else, it incurs an expected communication of $O(poly(n) \log |F|)$ bits.
ΠABA 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. ΠABA 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 ΠSBA 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 ΠSBA, where $T_{SBA} = (12n - 3) \cdot \Delta$. Given ΠSBA, we “combine” it with ΠABA to get a best-of-both-worlds BA protocol ΠBA (Fig 1). In ΠBA, the parties first run ΠSBA 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 ΠABA to decide the final output. In protocol ΠBA (and all other protocols in this paper), we specify by red color the maximum time required by each step in a synchronous network.

![Protocol ΠBA](image)

Figure 1: The best-of-both-worlds BA protocol that combines ΠSBA and ΠABA. The above code is executed by $P_i$.

**Theorem 3.3.** Let $t < n/3$. Let ΠSBA 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 ΠABA be a randomized protocol satisfying the conditions as per Lemma 3.2. Then ΠBA achieves the following.

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

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

### 3.1 Protocol ΠSBA: Synchronous BA Protocol with Asynchronous Guarantees

[16] presents a blueprint for ΠSBA 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 $\Pi$ be a protocol, where a sender $S \in P$ has input $m \in \{0, 1\}^\ell$, and parties obtain a possible output. $\Pi$ 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 $\perp$).
- **Weak Consistency**: If $S$ is *corrupt*, then every honest party with an output, outputs a common $m^* \in \{0,1\}^\ell$ or $\perp$.

$\Pi$ is called a $t$-perfectly-secure broadcast protocol, if it has Liveness, Validity and Consistency.

**A Blueprint for $\Pi_{SBA}$** Let $\Pi_{BC}$ 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 $\Pi_{BC}$ with $T_{BC} = (12n - 3) \cdot \Delta$). Given $\Pi_{BC}$, we design a protocol $\Pi_{SBA}$ (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 $\Pi_{BC}$. At time $T_{BC}$ the parties check if $n - t$ instances have produced outputs different from $\perp$ (which should happen in a *synchronous* network) and output the majority of those values. Theorem 3.5 is proved in Appendix C.

**Protocol $\Pi_{SBA}$**

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

**Theorem 3.5.** Let $t < n/3$. Let $\Pi_{BC}$ 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 $\Pi_{SBA}$ achieves the following, with its communication complexity being $n$ times that of $\Pi_{BC}$.

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

**3.2 Protocol $\Pi_{BC}$: Synchronous Broadcast with Asynchronous Guarantees**

From Theorems 3.3 and 3.5, the design of $\Pi_{BA}$ boils down to $\Pi_{BC}$. We first note that Bracha’s Acast protocol $\Pi_{ACast}$ does not satisfy the requirements of $\Pi_{BC}$, 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 $\Pi_{BC}$ is based on $\Pi_{ACast}$ 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 $\Pi_{BGP}$ of [15], which requires a communication of $O(n^2\ell)$ bits, if the inputs of the parties are of size $\ell$ bits. In $\Pi_{BGP}$, at time $T_{BGP} = (12n - 6) \cdot \Delta$, 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 $\Pi_{BGP}$ is satisfied at local time $(12n - 6) \cdot \Delta$ (if the criteria is not satisfied, then parties can output $\perp$). We stress that we only need guaranteed liveness from $\Pi_{BGP}$ 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 $3\Delta$, 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 $3\Delta$; if there is no output at time $3\Delta$, then the input for Π_{BGP} is ⊥. Finally, at time $3\Delta + T_{BGP}$, parties output $m^*$, if it has been received from Acast of S and if it is the output of Π_{BGP} as well, else the parties output ⊥. Now consistency is achieved for a corrupt S in a synchronous network because if any honest party obtains an output $m^* \neq ⊥$, then at least one honest party must have received $m^*$ from S’s Acast by time $3\Delta$ and so by time $3\Delta + T_{BGP}$, all honest parties will receive $m^*$ from S’s Acast.

**Eventual Consistency and Validity in Asynchronous Network:** In Π_{BC}, the parties set a “time-out” of $3\Delta + T_{BGP}$, 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 ⊥ at the end of time-out, while others may output ⊥. For the security of the VSS protocol, we would require the latter subset of (honest) parties to eventually output the common non-⊥ value, if the parties continue participating in Π_{BC}. For this, each $P_i$ who outputs ⊥ at time $3\Delta + T_{BGP}$, “switches” its output to $m^*$, if $P_i$ eventually receives $m^*$ 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 $3\Delta + T_{BGP}$, while fallback-mode consists of the process of deciding the output beyond time $3\Delta + T_{BGP}$.

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-⊥ outputs due to the consistency property of Π_{Acast}.

---

**Protocol Π_{BC}**

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

- **(Fallback Mode)**
  - Every $P_i \in P$ who has the output ⊥ at time $3\Delta + T_{BGP}$, changes it to $m^*$, if $m^*$ 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 $O(n^2\ell)$ bits, where all honest parties have an
output at time $T_{BGP} = (12n - 6) \cdot \Delta$. Then $\Pi_{BC}$ achieves the following for any $t < n/3$, with a communication complexity of $O(n^2 \ell)$ bits where $T_{BC} = (12n - 3) \cdot \Delta$.

- 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 $m^* \not\perp$ at time $T$ through fallback-mode (hence $T > T_{BC}$), then every honest party outputs $m^*$ 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 $\perp$. (c) Fallback Validity: If $S$ is honest, then each honest $P_h$ with output $\perp$ 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 $m^* \not\perp$ or $\perp$. (e) Fallback Consistency: If $S$ is corrupt and some honest party has output $m^* \not\perp$ at local time $T$ where $T \geq T_{BC}$, then every honest party eventually outputs $m^*$.

Terminologies for $\Pi_{BC}$: When we say that $P_i$ broadcasts $m$, we mean that $P_i$ invokes $\Pi_{BC}$ 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 $T_{BC}$) during the instance of $\Pi_{BC}$.

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

In the VSS protocol $\Pi_{VSS}$, there exists a dealer $D$ with $L$ number of $t_s$-degree polynomials $q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)$ where $L \geq 1$ and each (honest) $P_i$ is supposed to “verifiably” receive output shares $\{q^{(i)}(\alpha_i)\}_{i=1,\ldots,L}$; hence the goal is to generate a $t_s$-sharing of $\{q^{(i)}(0)\}_{i=1,\ldots,L}$. 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 $\Pi_{VSS}$, 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 $\Pi_{WPS}$ 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 $\Pi_{WPS}$ 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 $\Pi_{AWPS}$: $D$ 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 0K messages, if the tests are “positive”. Based on the 0K 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 \((t_s, t_a)\)-degree symmetric bivariate polynomial \(Q^*(x, y)\), where \(Q^*(x, y) = Q(x, y)\) for an honest \(D\). This is because \(\mathcal{E}'\) has at least \(t_s + 1\) honest parties with pair-wise consistent polynomials, defining \(Q^*(x, y)\), 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 \(Q^*(x, y)\) by applying OEC on the common points on these polynomials received from the parties in \(\mathcal{F}'\). Every \(P_i\) then outputs \(Q^*(0, \alpha_i)\) as its share, which is same as \(q^*(\alpha_i)\), where \(q^*(\cdot) = Q^*(0, y)\). Note that \(\Pi_{AWPS}\) 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:** \(\Pi_{AWPS}\) 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 \((n, t_s)\)-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 \(n - 2t_s + t_s > t_s\) 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 \(\Pi_{SWPS}\) 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 0K messages and their respective versions of the disputed points (the 0K messages were not required for \(\Pi_{AWPS}\)). The parties then construct the consistency graph. Next \(D\) removes all the parties from consideration in its consistency graph, who have made public “incorrect” 0K 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 \(W\), such that the polynomial of every party in \(W\) is publicly confirmed to be pair-wise consistent with the polynomials of at least \(n - t\) parties within \(W\). If a \(W\) is found, then \(D\) checks for the presence of an \((n, t_s)\)-star, say \((\mathcal{E}, \mathcal{F})\) among \(W\) 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 \(W\) is of size at least \(n - t_s\) and every party in \(W\) has an edge with at least \(n - t_s\) parties within \(W\) 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 \(W\). Furthermore, the parties now additionally verify whether any pair of parties \(P_j, P_k\) from \(W\) have made public “conflicting” 0K messages during the third phase. That is, if there exists any \(P_j, P_k \in W\) who made public 0K messages with \(q_{jk}\) and \(q_{kj}\) respectively during the third phase such that \(q_{jk} \neq q_{kj}\), then \(W\) 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 0K message is incorrect. Since \(D\) also would have seen these public 0K messages during the third phase, it should have have discarded the corresponding corrupt party, before finding \(W\). Hence if \((\mathcal{W}, \mathcal{E}, \mathcal{F})\) is accepted at the end of fourth phase, then the polynomials of all honest parties in \(W\) are guaranteed to be pair-wise consistent and lie on a single \((t_s, t_s)\)-degree symmetric bivariate polynomial, say \(Q^*(x, y)\), where \(Q^*(x, y) = Q(x, y)\).
for an honest D. This will further guarantee that the polynomials of all honest parties in \( F \) also lie on \( Q^*(x, y) \), as \( F \subseteq W \).

If \( (W, E, F) \) is accepted, then each \( P_i \in W \) outputs the constant term of its univariate polynomial as its share. On the other hand, the parties outside W attempt to obtain their corresponding polynomials lying on \( Q^*(x, y) \) by applying OEC on the common points on these polynomials received from the parties in \( F \) 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 \( W \) and hence will have the share \( q(\alpha_i) \). On the other hand, if a \( W \) is accepted for a corrupt D, then all the honest parties in \( W \) (which are at least \( t_s + 1 \) in number) will have their shares lying on \( q^*(\cdot) = Q^*(0, y) \).

Moreover, even if an honest party \( P_i \) outside \( W \) is able to obtain its desired share, as \( q^*(\alpha_i) \) due to the OEC mechanism. However, for a corrupt D, all the honest parties outside \( W \) may not be able to obtain their desired share, as \( F \) is guaranteed to have only \( n - 2t_s \) honest parties and OEC may fail. It is precisely for this reason that \( \Pi_{SWPS} \) fails to qualify as a VSS.

\( \Pi_{SWPS} + \Pi_{AWPS} \Rightarrow \text{Best-of-Both-Worlds WPS} \) \( \Pi_{WPS} \)

In \( \Pi_{WPS} \) (Fig 4), the parties first run \( \Pi_{SWPS} \) assuming a synchronous network, where \( \Pi_{BC} \) 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 \( \Pi_{SWPS} \) would have been over by time \( \Delta, 2\Delta, 2\Delta + T_{BC} \) and \( 2\Delta + 2T_{BC} \) respectively and by time \( 2\Delta + 2T_{BC} \), the parties should have accepted \( (W, E, F) \). However, in an asynchronous network, parties may have different opinion regarding the acceptance of \( (W, E, F) \). Hence at time \( 2\Delta + 2T_{BC} \), they run protocol \( \Pi_{BA} \). If the parties conclude that a \( (W, E, F) \) is accepted, then the parties compute their WPS-shares as per \( \Pi_{SWPS} \). However, we need to ensure that for a corrupt D in a synchronous network, if the polynomials of the honest parties in \( W \) are not pair-wise consistent, then the corresponding conflicting NOK messages are received within time \( 2\Delta + T_{BC} \) (the time required for the third phase of \( \Pi_{SWPS} \) to be over), so that \( (W, E, F) \) is not accepted. This is ensured by enforcing even a corrupt D to send the respective polynomials of all the honest parties in \( W \) by time \( \Delta \), so that the pair-wise consistency test between every pair of honest parties in \( W \) is over by time \( 2\Delta \). 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 \( W \) are not delivered within time \( \Delta \), then the results of the pair-wise consistency tests also get delayed beyond time \( 2\Delta + T_{BC} \) (the time-out of the third phase of \( \Pi_{SWPS} \)), thus ensuring that \( W \) is not accepted.

If the parties conclude that no \( (W, E, F) \) is accepted at time \( 2\Delta + 2T_{BC} \), then either D is corrupt or the network is asynchronous and hence parties resort to \( \Pi_{AWPS} \). 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 \( \Pi_{SWPS} \) 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 \( \Pi_{BA} \)) conclude that \( (W, E, F) \) is accepted, if some honest party(ies) accepts \( (W, E, F) \) within the time-out \( 2\Delta + 2T_{BC} \). Even in this case, the polynomials of all honest parties in \( W \) lie on a single \((t_s, t_a)\)-degree symmetric bivariate polynomial \( Q^*(x, y) \) for a corrupt D. This is because there will be at least \( n - 2t_s - t_a > t_s \) honest parties in \( E \) with pair-wise consistent polynomials, defining \( Q^*(x, y) \), and the polynomial of every honest party in \( F \) will be pair-wise consistent with the polynomials of all honest parties in \( E \) and hence lie on \( Q^*(x, y) \). Now consider any honest \( P_i \in (W \setminus F) \). As part of \( \Pi_{SWPS} \), it is ensured that the polynomial of \( P_i \) is consistent with the polynomials of at least \( n - t_s \) parties among \( W \). Among these \( n - t_s \) parties, at least \( n - 2t_s - t_a > t_s \) will be honest parties from \( F \). Thus, the polynomial of \( P_i \) will also lie on \( Q^*(x, y) \).
Protocol \( \Pi_{\text{WPS}}(D, q(\cdot)) \)

- **Phase I (Time \( \Delta \)) — Sending Polynomials:**
  - \( D \) on having the inputs \( q(\cdot) \) chooses a random \((t_s, t_a)\)-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 \( \Delta \)) — 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 \( \Delta \) and then send \( q_{ij} = q_i(\alpha_j) \) to \( P_j \), for \( j = 1, \ldots, n \).

- **Phase III (Time \( T_{\text{BC}} \)) — 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 \( \Delta \). Then broadcast \( \text{OK}(i, j) \) if \( q_{ij} = q_i(\alpha_j) \) holds, else broadcast \( \text{NOK}(i, j, q_i(\alpha_j)) \) if \( q_{ij} \neq q_i(\alpha_j) \).

- **Local computation — Constructing Consistency Graph:** Each \( P_i \in \mathcal{P} \) does the following.
  - Each \( P_i \) computes \( G \) from \( q(\cdot) \) through regular-mode.

- **Phase IV (Time \( T_{\text{BC}} \)) — Checking for \((n, t_s)\)-star:** \( D \) does the following in its consistency graph \( G_D \) at time \( 2\Delta + 2T_{\text{BC}} \).
  - Remove edges incident with \( P_i \), if \( \text{NOK}(i, j, q_{ij}) \) is received from the broadcast of \( P_i \) through regular-mode and \( q_{ij} \neq Q(\alpha_i, \alpha_j) \). Set \( W = \{P_j : \deg(P_j) \geq n - t_s\} \), where \( \deg(P_j) \) denotes the degree of \( P_j \) in \( G_D \). Remove \( P_j \) from \( W \) if \( P_j \) is not incident with at least \( n - t_s \) parties in \( W \). Repeat this step till no more parties can be removed from \( W \).
  - Run algorithm \( \text{AlgStar} \) on \( G_D[W] \), where \( G_D[W] \) denotes the subgraph of \( G_D \) induced by the vertices in \( W \). If an \((n, t_s)\)-star \((E, F)\) is obtained, then broadcast \((W, E, F)\).

- **Local Computation — Verifying and Accepting \((W, E, F)\):** Each \( P_i \in \mathcal{P} \) does the following at time \( 2\Delta + 2T_{\text{BC}} \).
  - If \((W, E, F)\) is received from \( D \)’s broadcast through regular-mode, then accept it if following were true at time \( 2\Delta + T_{\text{BC}} \):
    - \( \exists P_j, P_k \in W \), such that \( \text{NOK}(j, k, q_{jk}) \) and \( \text{NOK}(k, j, q_{kj}) \) messages were received from the broadcast of \( P_j \) and \( P_k \) respectively through regular-mode, where \( q_{jk} \neq q_{kj} \).
    - In \( G_i \), \( \deg(P_j) \geq n - t_s \) for all \( P_j \in W \) and \( P_j \) has edges with at least \( n - t_s \) parties from \( G_i[W] \).
    - \((E, F)\) was a \((n, t_s)\)-star in \( G_i[W] \) and \( \forall P_j, P_k \in W \) where the edge \((P_j, P_k)\) is present in \( G_i \), the \( \text{OK}(k, j) \) and \( \text{OK}(k, j) \) messages were received from the broadcast of \( P_j \) and \( P_k \) respectively through regular-mode.

- **Phase V (Time \( T_{\text{BA}} \)) — Deciding Whether to Go for \((n, t_a)\)-star:** At time \( 2\Delta + 2T_{\text{BC}} \), each \( P_i \in \mathcal{P} \) participates in an instance of \( \Pi_{\text{BA}} \) with input \( b_i = 0 \) if \((W, E, F)\) was accepted, else with input \( b_i = 1 \) and waits for time \( T_{\text{BA}} \).

- **Local Computation — Computing WPS-share Through \( W \):** If the output of \( \Pi_{\text{BA}} \) is 0, then each \( P_i \in \mathcal{P} \) computes \( \text{WPS-Share} \) \( s_i \) (initially set to \( \bot \)) as follows.
  - If a \((W, E, F)\) is not yet received then wait till \((W, E, F)\) is received from \( D \)’s broadcast through fall-back mode.
  - If \( P_i \in W \), then output \( s_i = q_i(0) \).
  - \((a)\) Else, initialise a support set \( \mathcal{S}_i \) to \( \emptyset \). Upon receiving \( q_{ji} \) from \( P_j \in F \), include \( q_{ji} \) to \( \mathcal{S}_i \).

- **Phase VI (Time \( T_{\text{BC}} \)) — Broadcasting \((n, t_a)\)-star:** If the output of \( \Pi_{\text{BA}} \) is 1, then \( D \) does the following.
  - After every update in \( G_D \), run \( \text{AlgStar} \) on \( G_D \). If an \((n, t_a)\)-star \((E', F')\) is obtained, then broadcast \((E', F')\).

- **Local Computation — Computing WPS-share Through \((n, t_a)\)-star:** If the output of \( \Pi_{\text{BA}} \) is 1, then each \( P_i \) does the following to compute its \( \text{WPS-Share} \).
  - Waits till an \((n, t_a)\)-star \((E', F')\) is obtained from the broadcast of \( D \), either through regular or fall-back mode. Upon receiving, wait till \((E', F')\) becomes an \((n, t_a)\)-star in \( G_i \).
  - If \( P_i \in F' \), then output \( s_i = q_i(0) \). Else, compute \( q_i(x) \) using similar procedure as in step \((a)\).
\(\Pi_{\text{WPS}}\) for \(L\) Polynomials If \(\mathcal{D}\) 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 \(\text{OK}(i,j)\) 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 \(\text{NOK}\) 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 \(\Pi_{\text{BA}}\) is used to decide whether any \((\mathcal{W}, \mathcal{E}, \mathcal{F})\) is accepted. The modified protocol will incur a communication of \(O(n^2L \log |\mathcal{F}| + n^4 \log |\mathcal{F}|)\) bits and involves 1 instance of \(\Pi_{\text{BA}}\). 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 \(\mathcal{D}\) has \(L\) number of \(t_s\)-degree polynomials \(q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)\) as input for \(\Pi_{\text{WPS}}\) where \(L \geq 1\) and let \(T_{\text{WPS}} = 2\Delta + 2T_{\text{BC}} + T_{\text{BA}}\). Then protocol \(\Pi_{\text{WPS}}\) achieves the following properties.

- **Correctness:** If \(\mathcal{D}\) is honest then: (a) In a synchronous network, each (honest) \(P_i\) outputs \(\{q^{(\ell)}(\alpha_i)\}_{\ell=1}^{L}\) at time \(T_{\text{WPS}}\). (b) In an asynchronous network, almost-surely, each (honest) \(P_i\) eventually outputs \(\{q^{(\ell)}(\alpha_i)\}_{\ell=1}^{L}\).

- **Privacy:** If \(\mathcal{D}\) is honest, then the view of adversary remains independent of \(q^{(1)}(\cdot), \ldots, q^{(L)}(\cdot)\).

- **Weak-Commitment:** If \(\mathcal{D}\) is corrupt, then either no honest party obtains any output or there exist \(t_s\)-degree polynomials \(\{q^{(\ell)}(\cdot)\}_{\ell=1}^{L}\), such that (a) In an asynchronous network, almost-surely, each (honest) \(P_i\) eventually outputs \(\{q^{(\ell)}(\alpha_i)\}_{\ell=1}^{L}\). (b) In a synchronous network, at least \(t_s + 1\) honest parties \(P_i\) output \(\{q^{(\ell)}(\alpha_i)\}_{\ell=1}^{L}\) and if any honest \(P_j\) outputs \(s^{(1)}_j, \ldots, s^L_j \in \mathcal{F}\), then \(s^{(\ell)}_j = q^{(\ell)}(\alpha_j)\) holds for \(\ell = 1, \ldots, L\).

### 4.2 The VSS Protocol

Protocol \(\Pi_{\text{WPS}}\) fails to serve as a VSS because if \(\mathcal{D}\) is corrupt and the network is synchronous, then the (honest) parties outside \(\mathcal{W}\) may not obtain their desired shares. Protocol \(\Pi_{\text{VSS}}\) (see Fig 11) fixes this shortcoming. For ease of understanding, we present the protocol assuming \(\mathcal{D}\) has a single \(t_s\)-degree polynomial as input and later discuss the modifications needed when \(\mathcal{D}\) has \(L\) such polynomials. The protocol has two “layers” of communication involved. The first layer is similar to \(\Pi_{\text{WPS}}\) 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 \(Q^*(x, y)\). 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 \(Q^*(x, y)\).

In a more detail, to perform the pair-wise consistency check of the polynomials received from \(\mathcal{D}\), each \(P_j\) upon receiving \(q_j(x)\) from \(\mathcal{D}\), shares the polynomial \(q_j(x)\) by invoking an instance of \(\Pi_{\text{WPS}}\) as a dealer. Any party \(P_i\) who computes a WPS-Share in this instance of \(\Pi_{\text{WPS}}\) either broadcasts an \(\text{OK}\) or \(\text{NOK}\) message for \(P_j\), depending on whether the WPS-share lies on the polynomial which \(P_i\) has received from \(\mathcal{D}\). 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 \(Q^*(x, y)\) 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 \(\Pi_{\text{WPS}}\) 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 $SS_i$. While the WPS-shares corresponding to the honest parties in $SS_i$ will be the common points on $Q^s(x, \alpha_i)$, the same holds even for corrupt parties in $SS_i$. This is because in order to be included in $\mathcal{F}$, such parties are “forced” to share polynomials lying on $Q^s(x, y)$, in their respective instances of $\Pi_{WPS}$. Now using the WPS-shares corresponding to the parties in $SS_i$, party $P_i$ will be able to compute $Q^s(x, \alpha_i)$ and hence, its share.

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

- **Phase I (Time $\Delta$) — Sending Polynomials**: same as in the protocol $\Pi_{WPS}$.
- **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 $\Delta$ and then does the following.
  - Act as a dealer and invoke an instance $\Pi^{(i)}_{WPS}$ of $\Pi_{WPS}$ to share $q_i(x)$. For $j = 1, \ldots, n$, participate in the instance $\Pi^{(j)}_{WPS}$, 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 $\Delta$ and then does the following.
  - If a WPS-share $q_{ji}$ is computed in $\Pi^{(j)}_{WPS}$ then broadcast $OK(i, j)$ if $q_{ji} = q_i(\alpha_j)$ holds, else broadcast $NOK(i, j, q_i(\alpha_j))$.
- **Local Computation — Constructing Consistency Graph**: Each $P_i$ constructs the consistency graph $G_i$ as in $\Pi_{WPS}$.
- **Phase IV (Time $T_{BC}$) — Constructing $(n, t_s)$—star**: $D$ does the following in its consistency graph $G_D$ at time $\Delta + T_{WPS} + T_{BC}$.
  - Check for $(W, E, F)$ as done in $\Pi_{WPS}$ and if a $(W, E, F)$ is computed, then broadcast it.
- **Local Computation — Verifying and Accepting $(W, E, F)$**: Each $P_i \in \mathcal{P}$ does the following at time $\Delta + T_{WPS} + 2T_{BC}$.
  - Accept any $(W, E, F)$ received from $D$’s broadcast through regular-mode, if the conditions for accepting $(W, E, F)$ in $\Pi_{WPS}$ were true at time $\Delta + T_{WPS} + T_{BC}$.
- **Phase V (Time $T_{BA}$) — Deciding Whether to Go for $(n, t_a)$—star**: At time $\Delta + T_{WPS} + 2T_{BC}$, each party $P_i$ participates in an instance of $\Pi_{BA}$ with input $b_i = 0$ if a $(W, E, F)$ is accepted, else with input $b_i = 1$ and waits for time $T_{BA}$.
- **Local Computation — Computing VSS-Share Through $(W, E, F)$**: If $\Pi_{BA}$ outputs 0, then each $P_i$ does the following.
  - If a $(W, E, F)$ is not yet received, then wait till it is received from $D$’s broadcast through fall-back mode.
  - If $P_i \in W$, then output $q_i(0)$.
  - Else, initialize $SS_i$ to $\emptyset$. Include $P_j \in F$ to $SS_i$ if a WPS-share $q_{ji}$ is computed during $\Pi^{(j)}_{WPS}$. Wait till $|SS_i| \geq t_s + 1$. Then interpolate $\{(\alpha_j, q_{ji})\}_{P_j \in SS_i}$ 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 $\Pi_{BA}$ is 1, then $D$ runs $AlgStar$ after every update in its consistency graph $G_D$ and broadcasts $(E', F')$, if it finds a $(n, t_a)$—star $(E', F')$.
- **Local Computation — Computing VSS-Share Through $(n, t_a)$—star**: If $\Pi_{BA}$ outputs 1, then each $P_i$ does the following.
  - Participate in any instance of $\Pi_{BC}$ invoked by $D$ for broadcasting an $(n, t_a)$—star only after time $\Delta + T_{WPS} + 2T_{BC} + T_{BA}$. Wait till some $(E', F')$ is obtained from $D$’s broadcast (through any mode), which constitutes an $(n, t_a)$—star in $G_i$.
  - If $P_i \in E'$, then output $q_i(0)$. Else, include $P_j \in F'$ to $SS_i$ (initialized to $\emptyset$) if a WPS-share $q_{ji}$ is computed in $\Pi^{(j)}_{WPS}$. Wait till $|SS_i| \geq t_s + 1$. Then interpolate $\{(\alpha_j, q_{ji})\}_{P_j \in SS_i}$ 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 $\Pi_{WPS}$ handling $L$ polynomials, with each party broadcasting a single $OK/NOK$ message for every other party. The protocol requires
a communication of $O(n^3 \log |F| + n^5 \log |F|)$ bits and $n + 1$ instances of $\Pi_{BA}$. 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 $\Pi_{VSS}$ where $L \geq 1$ and let $T_{VSS} = \Delta + T_{WPS} + 2T_{BC} + T_{BA}$. Then protocol $\Pi_{VSS}$ achieves the following properties.

- **Correctness:** If $D$ is honest, then: (a) In a synchronous network, each (honest) $P_i$ outputs \{\(q^{(i)}(\alpha_i)\)\}_{t=1}^{L}$ at time $T_{VSS}$. (b) In an asynchronous network, almost-surely, each (honest) $P_i$ eventually outputs \{\(q^{(i)}(\alpha_i)\)\}_{t=1}^{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 \(q^{(i)}(\cdot)\) such that. (a) In an asynchronous network, almost-surely, each honest $P_i$ eventually outputs \{\(q^{(i)}(\alpha_i)\)\}_{t=1}^{L}$. (b) In a synchronous network, every honest $P_i$ eventually outputs \{\(q^{(i)}(\alpha_i)\)\}_{t=1}^{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 $T > T_{VSS}$, 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 $\Pi_{Beaver}$ 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 $\Pi_{TripTrans}$ “transforms” a set of $t_s$-shared triples \{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i=1}^{2d+1}$ into “co-related” $t_s$-shared triples \{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i=1}^{2d+1}$, such that: (a): There exist $d$-degree polynomials $X(\cdot), Y(\cdot)$ and $2d$-degree polynomial $Z(\cdot)$, such that $X(\alpha_i) = x^{(i)}, Y(\alpha_i) = y^{(i)} \text{ and } Z(\alpha_i) = z^{(i)}$ holds for $i = 1, \ldots, 2d + 1$. (b): $z^{(i)} = x^{(i)} \cdot y^{(i)} \iff z^{(i)} = x^{(i)} \cdot y^{(i)}$. Hence, $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$, iff all the input triples are multiplication-triples. (c): Adversary learns $(x^{(i)}, y^{(i)}, z^{(i)})$ iff it knows $(x^{(i)}, y^{(i)}, z^{(i)})$. The protocol requires $d$ invocations of $\Pi_{Beaver}$.

**Agreement on a Common Subset:** In $\Pi_{ACS}$, each $P_i$ wants to distribute points on $L$ number of $t_s$-degree polynomials through $\Pi_{VSS}$. As corrupt parties may not invoke its instances of $\Pi_{VSS}$, 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. $\Pi_{ACS}$ lets the parties agree on a common subset $CS$ 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 $CS$. The idea is to invoke $n$ instances of $\Pi_{BA}$, where the $j^{th}$ instance is to decide if $P_j \in CS$.

**Triple-Sharing Protocol:** Protocol $\Pi_{TripSh}$ 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
The best-of-both-worlds perfectly-secure MPC protocol and conjecture that it is
leaves the following interesting open problems:

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

6 The Circuit-Evaluation Protocol
Protocol $\Pi_{\text{CirEval}}$ for evaluating cir has four phases. In the first phase, the parties generate $t_s$-sharing of $c_M$ random multiplication-triples through $\Pi_{\text{TripGen}}$. They also invoke $\Pi_{\text{ACS}}$ 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 $\Pi_{\text{CirEval}}$ 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 $f : \mathbb{F}^n \to \mathbb{F}$ 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, $\Pi_{\text{CirEval}}$ incurs a communication of $O\left(\frac{n^5}{2^k+1}c_M \log |\mathbb{F}| + n^7 \log |\mathbb{F}| \right)$ bits, invokes $O(n^3)$ instances of $\Pi_{\text{BA}}$ 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 $\Pi_{\text{ABA}}$. (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


19
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 RSDec$(d, r, W)$ that takes as input a set $W$ of distinct points on a $d$-degree polynomial and tries to output a $d$-degree polynomial, by correcting at most $r$ incorrect points in $W$. Coding theory [39] says that RS-Dec can correct up to $r$ errors in $W$ and correctly interpolate back the original polynomial if and only if $|W| \geq d + 2r + 1$ holds. There are several efficient implementations of RSDec (for example, the algorithm of Berlekamp-Welch).

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

![Protocol OEC($d, t, P'$)]

<table>
<thead>
<tr>
<th>Setting:</th>
<th>There exists a subset of parties $P'$ containing at most $t$ corrupt parties, with each $P_i \in P'$ having a point $q(\alpha_i)$ on some $d$-degree polynomial $q(\cdot)$. Every (honest) party in $P'$ is supposed to send its respective point to $P_R$, who is designated to reconstruct $q(\cdot)$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Computation</td>
<td>For $r = 0, \ldots, t$, party $P_R$ does the following in iteration $r$:</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Let $W$ denote the set of parties in $P'$ from whom $P_R$ has received the points and let $I_r$ denote the points received from the parties in $W$, when $W$ contains exactly $d + t + 1 + r$ parties.</td>
<td></td>
</tr>
<tr>
<td>Wait until $W \geq d + t + 1 + r$. Execute RSDec$(d, r, I_r)$ 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.</td>
<td></td>
</tr>
<tr>
<td>If for at least $d + t + 1$ values $v_i \in I_r$ it holds that $q_r(\alpha_i) = v_i$, then output $q_r(\cdot)$. Otherwise, proceed to the next iteration.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: The online error-correction procedure.

**Lemma A.1** ([19]). Let $P' \subseteq P$ contain at most $t$ corrupt parties and let there exist some $d$-degree polynomial $q(\cdot)$ with every (honest) $P_i \in P'$ 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 $d < (|P'| - 2t)$, then in a a synchronous network, $P_R$ outputs $q(\cdot)$ by the end of $\Delta$ 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 $O(n \log |F|)$ 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 $I_r$, 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 $d < (|P'| - 2t)$. Let adversary control $\hat{r}$ parties in $P'$, 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 $|I_{d+t+1+t-\hat{r}_2}| \geq d + 2\hat{r}_1 + 1$ 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 \( \Delta \) time. \( \square \)

### A.2 Bracha’s Acast Protocol

Bracha’s Acast protocol [18] tolerating \( t < n/3 \) corruptions is presented in Fig 7.

<table>
<thead>
<tr>
<th>Protocol ( \Pi_{\text{Acast}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. If ( P_i = S ), then on input ( m ), send ((\text{init}, S, m)) to all the parties.</td>
</tr>
<tr>
<td>2. Upon receiving the message ((\text{init}, S, m)) from ( S ), send ((\text{echo}, S, m)) to all the parties.</td>
</tr>
<tr>
<td>3. Upon receiving ((\text{echo}, S, m^<em>)) from ( n - t ) parties, send ((\text{ready}, S, m^</em>)) to all the parties.</td>
</tr>
<tr>
<td>4. Upon receiving ((\text{ready}, S, m^<em>)) from ( t + 1 ) parties, send ((\text{ready}, S, m^</em>)) to all the parties.</td>
</tr>
<tr>
<td>5. Upon receiving ((\text{ready}, S, m^<em>)) from ( n - t ) parties, output ( m^</em> ).</td>
</tr>
</tbody>
</table>

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 \( \Pi_{\text{Acast}} \).

**Lemma 2.3.** Protocol \( \Pi_{\text{Acast}} \) 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 \( m^* \), then every honest party eventually outputs \( m^* \).

- **Synchronous Network:**
  - **Liveness:** If \( S \) is honest, then all honest parties obtain some output within time \( 3\Delta \).
  - **Validity:** If \( S \) is honest, then every honest party with an output, outputs \( m \).
  - **Consistency:** If \( S \) is corrupt and some honest party outputs \( m^* \) at time \( T \), then every honest party outputs \( m^* \) 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 \( \ell \) 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 \( m^* \). We next show that all honest parties eventually obtain the output \( m^* \). Since \( P_h \) obtains the output \( m^* \), it implies that it receives \( n - t \) ready messages for \( m^* \) 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 \( m^* \). As there are at least \( n - t \) honest parties, it follows that eventually \( n - t \) ready messages for \( m^* \) 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 $m^*$.

On contrary, let $P_{h'}$ be another honest party, different from $P_h$, who outputs $m^{**} \neq m^*$. This implies that $P_{h'}$ received ready messages for $m^{**}$ 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 $m^*$, some honest party must have received $n - t$ echo messages for $m^*$, at most $t$ of which could come from the corrupt parties. Similarly, since $P_{h'}$ receives $n - t$ ready messages for $m^{**}$, some honest party must have received $n - t$ echo messages for $m^{**}$. However, since $n - t > 2t$, it follows that in order that $n - t$ echo messages are produced for both $m^*$ as well as $m^{**}$, it must be the case that some honest party must have generated an echo message, both for $m^*$, as well as $m^{**}$ 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 $\Delta$. Consequently, every honest party will send an echo message for $m$ to all the parties, which will be delivered within time $2\Delta$. Hence every honest party will send a ready message for $m$ to all the parties, which will be delivered within time $3\Delta$. 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 $3\Delta$ and output $m$.

If $S$ is corrupt and some honest party $P_h$ outputs $m^*$ at time $T$, then it implies that $P_h$ has received ready messages for $m^*$ during step 5 of the protocol at time $T$ from a set $H$ 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 $m^*$ at time $T + \Delta$. Consequently, by the end of time $T + \Delta$, every honest party would have sent a ready message for $m^*$ to every other honest party, which will be delivered within time $T + 2\Delta$. Hence, every honest party will output $m^*$ 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.

\[Q.E.D.\]

## 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 \(\sigma\)”;
- For \(\sigma \in \{0,1\}\), the output \((\sigma,1)\) stands for “distinct majority for \(\sigma\)”;
- The output \((\Lambda,0)\) stands for “non-distinct majority”.

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

```
Protocol \(\Pi_{\text{Vote}}\)

- On having the input \(x_i\), Acast (input, \(P_i, x_i\)).
- Create a dynamic set \(X_i\) which is initialized to \(\emptyset\). Add \((P_j, x_j)\) to \(X_i\) if (input, \(P_j, x_j\)) is received from the Acast of \(P_j\).
- Wait until \(|X_i| = n - t\). Then assign \(X_i = \mathcal{X}_i\), set \(a_i\) to the majority bit among \(\{x_j \mid (P_j, x_j) \in X_i\}\).
- Create a dynamic set \(Y_i\) which is initialized to \(\emptyset\). Add \((P_j, x_j, a_j)\) to \(Y_i\) if (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 \(|Y_i| = n - t\). Then assign \(Y_i = \mathcal{Y}_i\), set \(b_j\) to the majority bit among \(\{a_j \mid (P_j, X_j, a_j) \in Y_i\}\) and Acast (re-vote, \(P_j, Y_j, b_j\)).
- Create a set \(Z_i\), which is initialized to \(\emptyset\). Add \((P_j, Y_j, b_j)\) to \(Z_i\) if (re-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 \(\Pi_{\text{Vote}}\) 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 \(\sigma\), 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 \(\Pi_{\text{Vote}}\) achieves in a synchronous network is that all honest parties will receive their output by the end of time \(9\Delta\). Intuitively, this is because the protocol involves three different “phases” of Acast, each of which will produce an output within time \(3\Delta\) 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 \(\sigma\). 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 $\Pi_{Vote}$, all honest parties obtain their output within time $9\Delta$. Moreover, the output will be $(\sigma, 2)$, if all the honest parties have the same input $\sigma$.

**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 $3\Delta$. 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 $3\Delta$. Hence each honest $P_i$ will Acast a $(\text{vote}, P_i, X_i, a_i)$ message latest at time $3\Delta + 1$ and every honest party receives this message from the corresponding Acast instance within time $6\Delta$. 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 $X_k$, latest by time $5\Delta$. This further implies that upon receiving the message $(\text{vote}, P_i, X_i, a_i)$ 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 $Y_k$ sets within time $6\Delta$.

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

If all the honest parties have the same input $\sigma$, 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 vote message only for $\sigma$. Consequently, every honest party will output $(\sigma, 2)$. □

**B.2 Coin-Flipping Protocol**

The coin-flipping protocol denoted by $\Pi_{CoinFlip}$ (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

\[\text{If a corrupt party } P_j \text{ sends a vote message for } 1 - \sigma, \text{ then it will never be accepted and no honest party } P_i \text{ will ever include } (P_j, X_j, 1 - \sigma) \text{ in its } Y_i \text{ set. This is because } 1 - \sigma \text{ will not be the majority among the inputs of the honest parties in } X_j.\]
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 \( \Pi_{CoinFlip} \) 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 \( \perp \), 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 \( O(poly(n) \log |\mathbb{F}|) \) 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 \( O(poly(n) \log |\mathbb{F}|) \) bits.

### B.3 Vote + Coin-Flipping ⇒ ABA

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

In the first instance of \( \Pi_{Vote} \), 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 \( \Pi_{Vote} \), the parties participate in an instance of \( \Pi_{CoinFlip} \). Next, the parties decide their respective inputs for the second instance of \( \Pi_{Vote} \) 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 \( \Pi_{Vote} \), then the party continues with the bit received from that \( \Pi_{Vote} \) instance for the second \( \Pi_{Vote} \) instance. Otherwise, the party switches to the output received from \( \Pi_{CoinFlip} \). The output from the second \( \Pi_{Vote} \) 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 \( \Pi_{Vote} \), 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 $\Pi_{\text{Vote}}$ protocol in all the iterations and the outputs from $\Pi_{\text{CoinFlip}}$ 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 $\Pi_{\text{CoinFlip}}$ 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 $\Pi_{\text{Vote}}$ 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 $\Pi_{\text{Vote}}$ is either the output bit of the first instance of $\Pi_{\text{Vote}}$ or the output of $\Pi_{\text{CoinFlip}}$, both of which are independent of each other. Hence if the output of $\Pi_{\text{CoinFlip}}$ 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 $\Pi_{\text{Vote}}$. If in any iteration $k$, it is guaranteed that all honest parties have the same inputs for the second instance of $\Pi_{\text{Vote}}$, then the parties will obtain a common output and with highest grade from the second instance of $\Pi_{\text{Vote}}$. 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 $\Pi_{\text{Vote}}$. 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 $\Pi_{\text{CoinFlip}}$. This follows from the fact that the corrupt parties may deviate from the protocol instructions during the instances of $\Pi_{\text{CoinFlip}}$. There can be at most $(n - t)$ local-conflicts which may occur ($t$ potentially corrupt parties getting in conflict with $n - t$ honest parties) overall during various “failed” instances of $\Pi_{\text{CoinFlip}}$ (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 $\Pi_{\text{CoinFlip}}$ 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 $\Pi_{\text{CoinFlip}}$, 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 $\Pi_{\text{CoinFlip}}$ (and hence, same input for the second instance of $\Pi_{\text{Vote}}$) 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 $\Pi_{\text{CoinFlip}}$ protocol: while the $\Pi_{\text{CoinFlip}}$ protocol of \([3]\) will ensure that at least 1 new local-conflict is identified, the $\Pi_{\text{CoinFlip}}$ protocol of \([8]\) will ensure that $\Theta(n)$ number of new local-conflicts are identified.

\(^7\)Since each failed instance of the $\Pi_{\text{Coinflip}}$ 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 $\Pi_{\text{Coinflip}}$ 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 $^8\Pi_{\text{Vote}}$.

\textbf{Protocol $\Pi_{\text{ABA}}$}

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

Figure 9: The ABA protocol from $\Pi_{\text{Vote}}$ and $\Pi_{\text{CoinFlip}}$. 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 $\Pi_{\text{CoinFlip}}$ 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 \texttt{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 $\Pi_{\text{Vote}}$ during the first iteration will take at most $9\Delta$ time to produce output, while the instance of $\Pi_{\text{CoinFlip}}$ will take at most $c \cdot \Delta$ time. Additionally, $\Delta$ time will be taken by each party to send a \texttt{ready} message for $b$ to every other party. Consequently, $T_{\text{ABA}}$ will be $(c + 19) \cdot \Delta$.

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

\section*{C Properties of the Best-of-Both-Worlds BA Protocol}

We start with the proof of the properties of the protocol $\Pi_{\text{BA}}$.

\footnote{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 $\Pi_{\text{SBA}}$ 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 $T_{\text{SBA}}$. Moreover, let $\Pi_{\text{ABA}}$ be a randomized protocol satisfying the conditions as per Lemma 3.2. Then $\Pi_{\text{BA}}$ achieves the following.

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

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

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

C.1 Properties of the Protocol $\Pi_{\text{SBA}}$

In this section, we prove the properties of the protocol $\Pi_{\text{SBA}}$.

Theorem 3.5. Let $t < n/3$. Let $\Pi_{\text{BC}}$ 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 $T_{\text{BC}}$. Then, $\Pi_{\text{SBA}}$ achieves the following with its communication complexity being $n$ times that of $\Pi_{\text{BC}}$.

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

Proof. The guaranteed liveness, both in the synchronous and asynchronous network trivially follows from the liveness of $\Pi_{\text{BC}}$, which ensures that all the $n$ instances of $\Pi_{\text{BC}}$ produce some output within (local) time $T_{\text{BC}}$, both in a synchronous as well as an asynchronous network. Hence, $T_{\text{SBA}} = T_{\text{BC}}$. We next prove the rest of the properties in a synchronous network.
In a synchronous network, at least \( n-t > 2t \) instances of \( \Pi_{BC} \) corresponding to the honest senders, result in an output different from \( \perp \) for all honest parties (follows from the validity property of \( \Pi_{BC} \) in the synchronous network). Moreover, from the consistency property of \( \Pi_{BC} \) in the synchronous network, all honest parties obtain the same output in all the \( n \) instances of \( \Pi_{BC} \) instances. Hence, all honest parties output the same value, different from \( \perp \) (obtained through the majority rule), proving consistency of \( \Pi_{SBA} \). Finally, if all honest parties have the same input bit \( b \), then there will be at least \( n-t \) instances of \( \Pi_{BC} \) 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 \( \Pi_{SBA} \).

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 \( \perp \). 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 \( \Pi_{BC} \) are invoked in the protocol.

\[ \square \]

C.2 Properties of the Protocol \( \Pi_{BC} \)

In this section, we prove the properties of the protocol \( \Pi_{BC} \).

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

- Synchronous network: (a) Liveness: at time \( T_{BC} = (12n - 3) \cdot \Delta \), 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 \( T_{BC} \).
- Fallback Consistency: If \( S \) is corrupt and some honest \( P_h \) outputs \( m^* \neq \perp \) at time \( T \) through fallback-mode (hence \( T > T_{BC} \)), then every honest party outputs \( m^* \) 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 \( \perp \). (c) Fallback Validity: If \( S \) is honest, then all honest \( P_h \) with output \( \perp \) at local time \( T_{BC} \), eventually output \( m \) through fallback-mode.
- Weak Consistency: If \( S \) is corrupt, then at local time \( T_{BC} \), each honest \( P_h \) has output \( m^* \neq \perp \) or \( \perp \). (e) Fallback Consistency: If \( S \) is corrupt and some honest party has output \( m^* \neq \perp \) at local time \( T \) where \( T \geq T_{BC} \), then every honest party eventually outputs \( m^* \).

\textbf{Proof.} The liveness (both for the synchronous as well asynchronous network) simply follows from the fact that every honest party outputs something (including \( \perp \)) at (local) time \( T_{BC} = 3\Delta + T_{BGP} \), where \( T_{BGP} = (12n - 6) \cdot \Delta \). 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 \( \Pi_{BGP} \) in the synchronous network.

If \( S \) is honest, then due to the liveness and validity properties of \( \Pi_{Acast} \) in the synchronous network, at time \( 3\Delta \), every honest party \( P_i \) receives \( m \) from the Acast of \( S \). Hence, every honest party participates with input \( m \) in the instance of \( \Pi_{BGP} \). From the guaranteed liveness and validity of \( \Pi_{BGP} \) in synchronous network, at time \( 3\Delta + T_{BGP} \), every honest party will have \( m \) as the output from \( \Pi_{BGP} \). 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 \( \Pi_{BGP} \) at time \( T_{BC} \), which follows from the consistency property of \( \Pi_{BGP} \) in synchronous network. If all honest honest parties have the output \( \perp \) for \( \Pi_{BC} \) at time \( T_{BC} \), then
consistency holds trivially. So consider the case when some honest party, say $P_i$, has the output $m^* \neq \bot$ for $\Pi_{BC}$ at time $T_{BC}$. This implies that the output of $\Pi_{BGP}$ is $m^*$ for every honest party. Moreover, it also implies that at time $3\Delta$, at least one honest party, say $P_h$, has received $m^*$ from the Acast of $S$. Otherwise, all honest parties would participate with input $\bot$ in the instance of $\Pi_{BGP}$ and from the validity of $\Pi_{BGP}$ in the synchronous network, every honest party would compute $\bot$ as the output during $\Pi_{BGP}$, which is a contradiction. Since $P_h$ has $m^*$ from $S$’s Acast at time $3\Delta$, it follows from the consistency property of $\Pi_{ACast}$ in the synchronous network that all honest parties will have $m^*$ from $S$’s Acast by time $5\Delta$. Moreover, $5\Delta < (12n - 3) \cdot \Delta$ holds. Consequently, at time $(12n - 3) \cdot \Delta$, all honest parties will have $m^*$ from $S$’s Acast and $m^*$ as the output of $\Pi_{BGP}$ and hence, output $m^*$ for $\Pi_{BC}$.

For fallback consistency, we have to consider a corrupt $S$. Let $P_h$ be an honest party who outputs $m^* \neq \bot$ at time $T$ through fallback-mode, where $T > T_{BC}$. 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 $\Pi_{BC}$, 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 $m^*$. Since $P_h$ has obtained the output $m^*$, it means that at time $T$, it has received $m^*$ from the Acast of $S$. It then follows from the consistency of $\Pi_{ACast}$ in the synchronous network that every honest party will also receive $m^*$ from the Acast of $S$, latest by time $T + 2\Delta$ and output $m^*$.

We next prove the properties of the protocol $\Pi_{BC}$ in an asynchronous network, for which we depend upon the properties of $\Pi_{ACast}$ in the asynchronous network. The weak-validity property follows from the validity property of $\Pi_{ACast}$ 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 $\Pi_{ACast}$ 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 $\Pi_{ACast}$ in the asynchronous network that party $P_i$ will eventually obtain $m$ from the Acast of $S$ through the fallback-mode of $\Pi_{BC}$. 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 $m^*$ at time $T$ where $T \geq T_{BC}$. This implies that $P_j$ has obtained $m^*$ from the Acast of $S$. Now, consider an arbitrary honest $P_i$. From the liveness and weak consistency properties of $\Pi_{BC}$ in asynchronous network, it follows that $P_i$ outputs either $m^*$ or $\bot$ at local time $T_{BC}$. If $P_i$ has output $\bot$, then from the consistency property of $\Pi_{ACast}$ in the asynchronous network, it follows that $P_i$ will also eventually obtain $m^*$ from the Acast of $S$ through the fallback-mode of $\Pi_{BC}$. Consequently, party $P_i$ eventually changes its output from $\bot$ to $m^*$.

The communication complexity follows from the communication complexity of $\Pi_{BGP}$ and $\Pi_{ACast}$.

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

D.1  Proof of the Properties of the Protocol $\Pi_{\text{WPS}}$

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

---

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

- **Phase I** (Time $\Delta$) — 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 $Q^{(i)}(0, y) = q^{(i)}(\cdot)$ holds for $\ell = 1, \ldots, L$ and sends $q^{(i)}(\cdot) = Q^{(i)}(x, \alpha_i)$ to each party $P_i \in P$.
- **Phase II** (Time $\Delta$) — Exchanging Common Values:
  - Each $P_i$ upon receiving $t_s$-degree polynomials $\{q^{(i)}(\cdot)\}_{\ell=1}^{L}$ from $D$, does the following.
  - Wait till the local time becomes a multiple of $\Delta$ and then for $\ell = 1, \ldots, L$, send $q^{(i)}(\cdot) = q^{(i)}(\alpha_j)$ to each $P_j \in P$.
- **Phase III** (Time $T_{\text{BC}}$) — Publicly Declaring the Results of Pair-Wise Consistency Test:
  - Each $P_i \in P$ does the following.
  - Upon receiving $\{q^{(i)}_{ij}\}_{\ell=1}^{L}$ from $P_j$, wait till the local time becomes a multiple of $\Delta$. Then broadcast $\text{OK}(i, j)$ if $q^{(i)}_{ij} = q^{(i)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$. Else, set $\text{Conflict}_{ij} = \{\ell | q^{(i)}_{ij} \neq q^{(i)}(\alpha_j)\}$.
  - and broadcast $\text{NOK}(\ell_{\min}, i, j, q^{(i)}_{ij}(\alpha_j))$ where $\ell_{\min}$ is the minimum value in $\text{Conflict}_{ij}$.
- **Local computation — Constructing Consistency Graph:** Each $P_i \in P$ does the following.
  - Construct a consistency graph $G_i$ over $P$, where the edge $(P_j, P_k)$ is included in $G_i$, if $\text{OK}(j, k)$ and $\text{OK}(k, j)$ is received from the broadcast of $P_j$ and $P_k$ respectively, either through the regular-mode or fall-back mode.
- **Phase IV** (Time $T_{\text{BC}}$) — Checking for $(n, t_s)$—star:
  - $D$ does the following in its consistency graph $G_D$ at time $2\Delta + T_{\text{BC}}$.
  - Remove edges incident with $P_i$ from $G_D$, if $\text{NOK}(\ell, i, j, q^{(i)}_{ij})$ is received from the broadcast of $P_i$ through regular-mode such that $q^{(i)}_{ij} \neq Q^{(i)}(\alpha_j, \alpha_i)$.
  - Compute $\mathcal{W} = \{P_i : \deg(P_i) \geq n - t_s\}$, where $\deg(P_i)$ 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 $\text{AlgStar}$ on $G_D|\mathcal{W}$, where $G_D|\mathcal{W}$ 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 $P_i \in P$ does the following at time $2\Delta + 2T_{\text{BC}}$.
  - 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 $2\Delta + T_{\text{BC}}$:
    - $\forall P_j, P_k \in \mathcal{W}, \exists \ell \in \{1, \ldots, L\}$, such that $\text{NOK}(\ell, j, k, q^{(i)}_{jk})$ and $\text{NOK}(\ell, k, j, q^{(i)}_{kj})$ messages were received from the broadcast of $P_j$ and $P_k$ respectively through regular-mode, where $q^{(i)}_{jk} \neq q^{(i)}_{kj}$.
    - In $G_i$, $\deg(P_j) \geq n - t_s \forall P_j \in \mathcal{W}$ and $P_j$ had edges with at least $n - t_s$ parties from $G^i_D|\mathcal{W}$.
    - $(\mathcal{E}, \mathcal{F})$ was an $(n, t_s)$—star in $G^i_D|\mathcal{W}$ and $\forall P_j, P_k \in \mathcal{W}$ where the edge $(P_j, P_k)$ was present in $G_i$, the $\text{OK}(j, k)$ and $\text{OK}(k, j)$ messages were received from the broadcast of $P_j$ and $P_k$ respectively through regular-mode.
- **Phase V** (Time $T_{\text{BA}}$) — Deciding Whether to Go for $(n, t_s)$—star:
  - At time $2\Delta + 2T_{\text{BC}}$, each $P_i \in P$ participates in an instance of $\Pi_{\text{BA}}$ 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 $T_{\text{BA}}$.
- **Local Computation — Computing WPS-shares Through $W$:** If the output of $\Pi_{\text{BA}}$ is 0, then each $P_i \in P$ computes $WPS-Shares s^{(1)}_i, \ldots, s^{(L)}_i$ (initially all 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...
parties. Furthermore, no information about the polynomials of the honest parties is revealed during the pair-wise consistency check. 

\begin{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 \( \text{OK}((\ell, i, j, q^{(\ell)}_{ij})) \) 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.
\end{proof}

\begin{lemma}[Correctness in Synchronous Network]
In protocol \( \Pi_{WPS} \), 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 \( T_{WPS} = 2\Delta + 2T_{BC} + T_{BA} \).
\end{lemma}

\begin{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 \( q^{(\ell)}_{ij}(x) = Q^{(\ell)}(x, \alpha_j) \) from \( D \) within time \( \Delta \), for \( \ell = 1, \ldots, L \). Hence during phase II, every honest \( P_j \) sends \( \{q^{(\ell)}_{ik}\}_{\ell=1,\ldots,L} \) to every \( P_k \), which takes at most \( \Delta \) time to be delivered. Hence, by time \( 2\Delta \), every honest \( P_j \) receives \( \{q^{(\ell)}_{kj}\}_{\ell=1,\ldots,L} \) from every honest \( P_k \), such that \( q^{(\ell)}_{kj} = q^{(\ell)}_{ij}(\alpha_k) \) holds. Consequently, during phase III, every honest \( P_j \) broadcasts \( \text{OK}(j, k) \) corresponding to every honest \( P_k \), and vice versa. From the validity property of \( \Pi_{BC} \) in the synchronous network, it follows that every honest \( P_i \) receives \( \text{OK}(j, k) \) and \( \text{OK}(k, j) \) from the broadcast of every honest \( P_j \) and every honest \( P_k \) through regular-mode, at time \( 2\Delta + T_{BC} \). 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
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 \( \Pi_{BA} \) at local time \( 2\Delta + 2T_{BC} \). Hence almost-surely, the instance of \( \Pi_{BA} \) eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of \( \Pi_{BA} \) is 0:** This means that at least one honest party, say \( P_h \), has accepted \((\mathcal{W}, \mathcal{E}, \mathcal{F})\) at local \( 2\Delta + 2T_{BC} \), received from the broadcast of D. Hence, by the weak validity and fallback validity of \( \Pi_{BC} \) 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 \( \Pi_{BC} \), the consistency graphs of all honest parties will eventually have all the edges which were present in \( G_h \) at time \( 2\Delta + 2T_{BC} \). 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 \( \Pi_{BA} \) is 1:** Since D is honest, every pair of honest parties \( P_j, P_k \) eventually broadcast \( \text{OK}(j, k) \) and \( \text{OK}(k, j) \) messages, as the pair-wise consistency check between them will eventually be successful. From the weak validity and fallback validity of \( \Pi_{BC} \), these messages are eventually delivered to every honest party. Also from the weak consistency and fallback consistency of \( \Pi_{BC} \), any \( \text{OK} \) 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 \( \Pi_{BC} \), 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 \( \Pi_{BC} \). 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 \( Q^{(\ell)}(x, \alpha_i) \).

\[ \Box \]

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

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

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

**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 \( (W, \mathcal{E}, \mathcal{F}) \) from the broadcast of \( D \) through regular-mode and accepted it at time \( 2\Delta + 2T_{BC} \). From the protocol steps, the following must be true for \( P_h \) at time \( 2\Delta + T_{BC} \):

- \( \exists P_j, P_k \in W, \exists \ell \in \{1, \ldots, L\}, \) such that \( \text{NOK}(\ell, j, k, q^{(\ell)}_{jk}) \) and \( \text{NOK}(\ell, k, j, q^{(\ell)}_{jk}) \) messages had been received from the broadcast of \( P_j \) and \( P_k \) respectively through regular-mode, where \( q^{(\ell)}_{jk} \neq q^{(\ell)}_{kj} \).
- In \( P_h \)'s consistency graph \( G_h \), \( \text{deg}(P_j) \geq n - t_s \) for all \( P_j \in W \) and \( P_j \) has edges with at least \( n - t_s \) parties from \( W \).
- \( (\mathcal{E}, \mathcal{F}) \) was an \((n, t_s)\)-star in the induced subgraph \( G_h[W] \), such that \( \forall P_j, P_k \in W \) where the edge \((P_j, P_k)\) is present in \( G_h \), messages \( \text{OK}(j, k) \) and \( \text{OK}(k, j) \) 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 \( P_j \in W \) be an honest party, who received its \( t_s \)-degree univariate polynomials \( \{q^{(\ell)}_j(x)\}_{\ell=1,\ldots,L} \) at time \( \Delta + \delta \), where \( \delta > 0 \). Moreover, let \( P_k \in W \) be an honest party different from \( P_j \) (note that there are at least \( n - 2t_s \) honest parties in \( W \)). As stated above, at time \( 2\Delta + T_{BC} \), party \( P_h \) has received the message \( \text{OK}(k, j) \) from the broadcast of \( P_k \) through regular-mode. From the protocol steps, \( P_k \) waits till its local time becomes a multiple of \( \Delta \), before it sends the points on its polynomial to other parties. Hence, \( P_j \) must have started sending the points after time \( r \cdot \Delta \), where \( r \geq 2 \). Since the network is synchronous, the points \( \{q^{(\ell)}_{jk} = q^{(\ell)}_j(\alpha_k)\}_{\ell=1,\ldots,L} \) must have been received by \( P_k \) by time \( (r + 1) \cdot \Delta \). Moreover, from the protocol steps, even if \( P_k \) receives these points at time \( T \), where \( r \cdot \Delta < T < (r + 1) \cdot \Delta \), it waits till time \( (r + 1) \cdot \Delta \), before broadcasting the \( \text{OK}(k, j) \) message. Since \( P_k \) is honest, from the validity property of \( \Pi_{BC} \) in synchronous network, it will take exactly \( T_{BC} \) time for the message \( \text{OK}(k, j) \) to be received through regular-mode, once it is broadcasted. This implies that \( P_k \) will receive the message \( \text{OK}(k, j) \) at time \( (r + 1) \cdot \Delta + T_{BC} \), where \( (r + 1) \cdot \Delta + T_{BC} \). However, this is a contradiction, since the \( \text{OK}(k, j) \) message has been received by \( P_h \) at time \( 2\Delta + T_{BC} \).

To prove the second part of the lemma, we will show that for \( \ell = 1, \ldots, L \), the univariate polynomials \( q^{(\ell)}_j(x) \) of all the honest parties in \( W \) are pair-wise consistent. Since there are at least \( n - 2t_s > t_s \) honest parties in \( W \), from Lemma 2.1 it follows that the univariate polynomials \( q^{(\ell)}_j(x) \)
of all the honest parties in \( W \) lie on a unique \((t_s,t_s)\)-degree symmetric bivariate polynomial, say \( Q^{i(t)}(x,y) \). Consider an arbitrary pair of honest parties \( P_j, P_k \in W \). From the first part of the lemma, both \( P_j \) and \( P_k \) must have received their respective univariate polynomials \( q_j^{(t)}(x) \) and \( q_k^{(t)}(x) \) by time \( \Delta \). This further implies that \( P_j \) and \( P_k \) must have received the points \( q_{kj}^{(t)} = q_k^{(t)}(\alpha_j) \) and \( q_{jk}^{(t)} = q_j^{(t)}(\alpha_k) \) respectively by time \( 2\Delta \). 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, \( \text{Conflict}_{jk} = \text{Conflict}_{kj} \). If \( \text{Conflict}_{jk} \neq \emptyset \), then \( P_j \) and \( P_k \) would broadcast \( \text{OK}(\ell,j,k,q_{jk}^{(t)}) \) and \( \text{OK}(\ell,k,j,q_{kj}^{(t)}) \) messages respectively at time \( 2\Delta \), where \( \ell \) is the minimum value in \( \text{Conflict}_{jk} \) and \( \text{Conflict}_{kj} \). From the validity property of \( \Pi_{BC} \) in the synchronous network, \( P_h \) will receive these messages through regular-mode at time \( 2\Delta + T_{BC} \). Consequently, \( P_h \) will not accept \((W,E,F)\), which is a contradiction.

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

\( \Box \)

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

- There are at least \( t_s + 1 \) honest parties \( P_i \) who output \( \{q^{i(t)}(\alpha_t)\}_{t=1,\ldots,L} \).
- If any honest \( P_j \) outputs \( s_j^{(1)}, \ldots, s_j^{(L)} \in F \), then \( s_j^{(t)} = q^{i(t)}(\alpha_t) \) holds for \( t = 1, \ldots, L \).

**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 \( \ell \) elements of \( F \). Now, there are two possible cases.

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

- **At time $2T_{\text{BC}} + 2\Delta$, no honest party has accepted any $(W, \mathcal{E}, \mathcal{F})$:** This implies that all honest parties participate in $\Pi_{\text{BA}}$ with input 1 and by the validity and guaranteed liveness of $\Pi_{\text{BA}}$, all honest parties obtain the output 1 in $\Pi_{\text{BA}}$. Let $P_h$ be the first honest party who outputs $\ell$ values from $\mathcal{F}$. 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 $\Pi_{\text{BC}}$ 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 OEC($\star, \star$) 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 $\Pi_{\text{BC}}$ in synchronous network. Since $|E'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, 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 A.1, lie on a unique $(t_s, t_a)$-degree symmetric bivariate polynomial, say $Q^*(x, y)$. 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 $Q^*(x, y)$. Let $q^{(\ell)}(\cdot) \overset{def}{=} Q^*(0, y)$. We show that every honest $P_i$ outputs $q^{(\ell)}(\alpha_i)$. For this it is enough to show that each honest $P_i$ gets $q_i^{(\ell)}(x) = Q^*(x, \alpha_i)$, as $P_i$ outputs $q_i^{(\ell)}(0)$, which will be the same as $q^{(\ell)}(\alpha_i)$. Consider an arbitrary honest party $P_i$. There are two possible cases.

- $P_i \in \mathcal{F}'$: In this case, $P_i$ already has $Q^*(x, \alpha_i)$, 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, $Q^*(x, \alpha_i)$ 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 $Q^*(x, \alpha_i)$ received from the parties in $\mathcal{F}'$, party $P_i$ will compute $Q^*(x, \alpha_i)$, for $\ell = 1, \ldots, L$. \hfill \Box

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

**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 $\ell$ elements of $\mathcal{F}$. We note that every honest party participates with some input in the instance of $\Pi_{\text{BA}}$ at local time $2\Delta + 2T_{\text{BC}}$. Hence, almost-surely, the instance of
\(\PiBA\) eventually terminates for every honest party, with some common output. Now there are two possible cases:

- **The output of \(\PiBA\) is 0:** This means that at local time \(2\Delta + 2T_{BC}\), 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 \(\PiBC\) 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 \(q_i^{(t)}(x)\) are pair-wise consistent and define a \((t_s, t_s)\)-degree symmetric bivariate polynomial, say \(Q^{(t)}(x, y)\), for \(\ell = 1, \ldots, L\). Similarly, the univariate polynomial \(q_j^{(t)}(x)\) of every honest party \(P_j \in \mathcal{F}\) will be pair-wise consistent with the univariate polynomials \(q_i^{(t)}(x)\) of all the honest parties in \(\mathcal{E}\) and lie on \(Q^{(t)}(x, y)\). For \(\ell = 1, \ldots, L\), let \(q^{(t)}(\cdot) \overset{\text{def}}{=} Q^{(t)}(0, y)\). We claim that every honest \(P_i\) will eventually get \(Q^{(t)}(x, \alpha_i)\), for \(\ell = 1, \ldots, L\). This implies that eventually every honest \(P_i\) outputs \(s_i^{(t)} = Q^{(t)}(0, \alpha_i) = q^{(t)}(\alpha_i)\). 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 \(q_i^{(t)}(x)\), received from \(D\) and since \(P_i \in \mathcal{F}\), the condition \(q_i^{(t)}(x) = Q^{(t)}(x, \alpha_i)\) holds.

- \(P_i \in \mathcal{W}\) and \(P_i \notin \mathcal{F}\): In this case, \(P_i\) already has \(q_i^{(t)}(x)\), 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 \(q_i^{(t)}(x)\) is pair-wise consistent with the polynomials \(q_j^{(t)}(x)\) at least \(n - t_s\) parties \(P_j \in \mathcal{W}\) (since \(P_i\) has edges with at least \(n - t_s\) parties \(P_j \in \mathcal{W}\)). Now among these \(n - t_s\) parties, at least \(n - 2t_s\) will be from \(\mathcal{F}\), of which at least \(n - 2t_s - t_a > t_s\) parties will be honest. Hence, \(q_i^{(t)}(x)\) is pair-wise consistent with the \(q_j^{(t)}(x)\) polynomials of at least \(t_s + 1\) honest parties \(P_j \in \mathcal{F}\). Now since the \(q_j^{(t)}(x)\) polynomials of all the honest parties in \(\mathcal{F}\) lie on \(Q^{(t)}(x, y)\), it implies that \(q_i^{(t)}(x) = Q^{(t)}(x, \alpha_i)\) 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 \(Q^{(t)}(x, \alpha_i)\) 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 \(Q^{(t)}(x, \alpha_i)\) polynomial received from the parties in \(\mathcal{F}\), party \(P_i\) will eventually obtain \(Q^{(t)}(x, \alpha_i)\).

- **The output of \(\PiBA\) is 1:** Let \(P_h\) be the first honest party, who outputs \(\ell\) values from \(\mathcal{F}\). 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 \(\PiBC\) 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 \(\PiBC\) 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 \(q_i^{(t)}(x)\) are pair-wise consistent and hence from Lemma 2.1, lie on a unique degree-\((t_s, t_s)\) symmetric bivariate polynomial, say \(Q^{(t)}(x, y)\), for \(\ell = 1, \ldots, L\). Similarly, since the univariate polynomial \(q_j^{(t)}(x)\) of every honest party \(P_j \in \mathcal{F}'\) is pair-wise consistent with the univariate polynomials \(q_i^{(t)}(x)\) of the honest parties in \(\mathcal{E}'\), it implies that the univariate polynomial \(q_j^{(t)}(x)\) of all the honest parties in \(\mathcal{F}'\) also lie on \(Q^{(t)}(x, y)\), for \(\ell = 1, \ldots, L\). Let \(q^{(t)}(\cdot) \overset{\text{def}}{=} Q^{(t)}(0, y)\). We show that every honest \(P_i\) eventually outputs \(\{q^{(t)}(\alpha_i)\}_{\ell=1,...,L}\). For this it is enough to show that
each honest $P_i$ eventually gets $q_i^{(t)}(x) = Q^{(t)}(x, \alpha_i)$, as $P_i$ outputs $q_i^{(t)}(0)$, which will be same as $q^{(t)}(\alpha_i)$. Consider an arbitrary honest $P_i$. There are two possible cases.  
- $P_i \in \mathcal{F}':$ In this case, $P_i$ already has $Q^{(t)}(x, \alpha_i)$, 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 $Q^{(t)}(x, \alpha_i)$ is a $t_s$-degree polynomial and $t_a < |\mathcal{F}'| - 2t_a$, from Lemma A.1, it follows that by applying the OEC procedure on the common points on the polynomial $Q^{(t)}(x, \alpha_i)$ received from the parties in $\mathcal{F}'$, party $P_i$ will eventually obtain $Q^{(t)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.  

$\blacklozenge$

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

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 $\Pi_{BC}$ (see Theorem 3.6).  

$\blacklozenge$

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

D.2 Proof of the Properties of the Protocol $\Pi_{VSS}$

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

**Protocol $\Pi_{VSS}(D, \{q_i^{(t)}(\cdot)\}_{t=1, \ldots, L})$**

- **Phase I (Time $\Delta$) — Sending Polynomials:**  
  - $D$ on having the input $q_1^{(1)}(\cdot), \ldots, q_L^{(1)}(\cdot)$ chooses random $(t_s, t_s)$-degree symmetric bivariate polynomials $Q_1^{(1)}(x, y), \ldots, Q_{L}^{(1)}(x, y)$ such that $Q^{(1)}(0, y) = q^{(1)}(\cdot)$, for $\ell = 1, \ldots, L$ and sends $q_i^{(1)}(x) = Q^{(1)}(x, \alpha_i)$ 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 $\{q_i^{(t)}(x)\}_{t=1, \ldots, L}$ from $D$, waits till the current local time becomes a multiple of $\Delta$ and then does the following.  
  - Act as a dealer and invoke $L$ instances $\Pi_{WPS}^{(i, t)}$ of $\Pi_{WPS}$ to share $q_i^{(t)}(x)$, for $\ell = 1, \ldots, L$. For $j = 1, \ldots, n$, participate in the instances $\Pi_{WPS}^{(j, \ell)}$, 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 $\Delta$ and does the following.  
    - If WPS-shares $q_{i, 1}^{(1)}$, $\ldots$, $q_{i, L}^{(1)}$ are computed during the instances $\Pi_{WPS}^{(1, 1)}, \ldots, \Pi_{WPS}^{(1, L)}$ respectively, then broadcast OK($i, j$) if $q_{i, \ell}^{(t)} = q_i^{(t)}(\alpha_j)$ holds for $\ell = 1, \ldots, L$. Else, set $\text{Conflict}_{ij} = \{\ell | q_{i, \ell}^{(t)} \neq q_i^{(t)}(\alpha_j)\}$ and broadcast NOK($\ell_{\min}, i, j, q_{i, \ell_{\min}}^{(t)}(\alpha_j)$) where $\ell_{\min}$ is the minimum value in $\text{Conflict}_{ij}$.
  - **Local Computation — Constructing Consistency Graph:** Each $P_i$ constructs the consistency graph $G_i$ as in $\Pi_{WPS}$.
- **Phase IV (Time $T_{BC}$) — Constructing $(n, t_s)$-star:** $D$ does the following in its consistency graph $G_D$ at time $\Delta + T_{WPS} + T_{BC}$. 

39
- Remove edges incident with $P_i$ from $G_D$, if $\text{NOK}(\ell, i, j, q_{ij}^{(\ell)})$ is received from the broadcast of $P_i$ through regular-mode such that $q_{ij}^{(\ell)} \neq Q^{(\ell)}(\alpha_j, \alpha_i)$.
- Compute $W = \{P_i : \deg(P_i) \geq n - t_s\}$, where $\deg(P_i)$ denotes the degree of $P_i$ in $G_D$. Remove party $P_i$ from $W$ if in $G_D$, party $P_i$ is not incident with at least $n - t_s$ parties in $W$. Repeat this step till no more parties can be removed from $W$.
- Run algorithm AlgStar on $G_D[W]$, where $G_D[W]$ denotes the subgraph of $G_D$ induced by the vertices in $W$. If an $(n, t_s)$-star $(E, F)$ is obtained, then broadcast $(W, E, F)$.

- **Local Computation — Verifying and Accepting** $(W, E, F)$: Each $P_i \in P$ does the following at time $\Delta + T_{\text{WPS}} + 2T_{\text{BC}}$.
  - If a $(W, E, F)$ is received from the broadcast of $D$ through regular-mode, then accept it if following were true at time $\Delta + T_{\text{WPS}} + T_{\text{BC}}$:
    - $\forall P_j, P_k \in W, \forall \ell \in \{1, \ldots, L\}$, such that $\text{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\text{NOK}(\ell, j, k, q_{kj}^{(\ell)})$ 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 $G_i$, $\deg(P_j) \geq n - t_s$ for all $P_j \in W$ and $P_j$ had edges with at least $n - t_s$ parties from $G_i[W]$.
    - $(E, F)$ was an $(n, t_s)$-star in $G_i[W]$ and $\forall P_j, P_k \in W$ where the edge $(P_j, P_k)$ was present in $G_i$, the $\text{OK}(j, k)$ and $\text{OK}(k, j)$ messages were received from the broadcast of $P_j$ and $P_k$ respectively through regular-mode.

- **Phase V** $(Time\ T_{\text{BA}})$ — Deciding Whether to Go for $(n, t_s)$-star: At time $\Delta + T_{\text{WPS}} + 2T_{\text{BC}}$, each party $P_i$ participates in an instance of $\Pi_{\text{BA}}$ with input $b_i = 0$ if a $(W, E, F)$ is accepted, else with input $b_i = 1$ and waits for time $T_{\text{BA}}$.

- **Local Computation — Computing VSS-Shares Through** $(W, E, F)$: If $\Pi_{\text{BA}}$ outputs 0, then each $P_i$ does the following.
  - If a $(W, E, F)$ has not been received from $D$’s broadcast, then wait till it is received from $D$’s broadcast through fall-back mode.
  - If $P_i \in W$, then output $\{q_i^{(\ell)}(0)\}_{\ell=1, \ldots, L}$.
  - Else, initialize $SS_i^{(\ell)}$ to $\emptyset$, for $\ell = 1, \ldots, L$. Include $P_j \in F$ to $SS_i^{(\ell)}$ if a WPS-share $q_{ji}^{(\ell)}$ is computed during the instance $\Pi_{\text{WPS}}^{(j, \ell)}$. Wait till $|SS_i^{(\ell)}| \geq t_s + 1, \forall \ell = 1, \ldots, L$. Then interpolate $\{(\alpha_j, q_{ji}^{(\ell)})\}_{P_j \in SS_i^{(\ell)}}$ to get a $t_s$-degree polynomial $q_i^{(\ell)}(x)$ and output $\{q_i^{(\ell)}(0)\}_{\ell=1, \ldots, L}$.

- **Phase VI** $(Time\ T_{\text{BC}})$ — Broadcasting $(n, t_s)$-star: If the output of $\Pi_{\text{BA}}$ is 1, then $D$ does the following.
  - After every update in $G_D$, run AlgStar on $G_D$. If an $(n, t_s)$-star $(E', F')$ is obtained, then broadcast $(E', F')$.

- **Local Computation — Computing VSS-Shares Through** $(n, t_s)$-star: If $\Pi_{\text{BA}}$ outputs 1, then each $P_i$ does the following.
  - Start participating in any instance of $\Pi_{\text{BC}}$ invoked by $D$ for broadcasting an $(n, t_s)$-star only after time $\Delta + T_{\text{WPS}} + 2T_{\text{BC}} + T_{\text{BA}}$. Wait till some $(E', F')$ is obtained from the broadcast of $D$ (through any mode), which constitutes an $(n, t_s)$-star in $G_i$.
  - If $P_i \in F'$, then output $\{q_i^{(\ell)}(0)\}_{\ell=1, \ldots, L}$. Else, initialize $SS_i^{(\ell)}$ to $\emptyset$, for $\ell = 1, \ldots, L$. Include $P_j \in F'$ to $SS_i^{(\ell)}$ if a WPS-share $q_{ji}^{(\ell)}$ is computed during the instance $\Pi_{\text{WPS}}^{(j, \ell)}$. Wait till $|SS_i^{(\ell)}| \geq t_s + 1, \forall \ell = 1, \ldots, L$. Then interpolate $\{(\alpha_j, q_{ji}^{(\ell)})\}_{P_j \in SS_i^{(\ell)}}$ 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 $\Pi_{\text{VSS}}$, if $D$ is honest, then irrespective of the network type, the view of the adversary remains independent of $\{q_i^{(\ell)}(\cdot)\}_{\ell=1, \ldots, L}$.

**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 $q^{(\ell)}_i(\alpha_j)$ 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 $\Pi_{WPS}$, the view of the adversary remains independent of $q^{(\ell)}_i(x)$ during $\Pi^{(\ell)}_{WPS}$, 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 $\text{NOK}(\ell, i, j, q^{(\ell)}_{ij})$ 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 $\Pi_{VSS}$, if $D$ is honest and network is synchronous, then each honest $P_i$ outputs $\{q^{(\ell)}(\alpha_i)\}_{\ell=1,\ldots,L}$ at time $T_{VSS} = \Delta + T_{WPS} + 2T_{BC} + T_{BA}$.

Proof. Let $D$ be honest and network be synchronous with up to $t_s$ corruptions. During phase I, all honest parties receive $q^{(\ell)}_i(x) = Q^{(\ell)}(x,\alpha_i)$, for $\ell = 1, \ldots, L$ from $D$ within time $\Delta$. Consequently during phase II, each honest $P_i$ shares the polynomials $\{q^{(\ell)}_i(x)\}_{\ell=1,\ldots,L}$. From the correctness of $\Pi_{WPS}$ in synchronous network, corresponding to each honest $P_j$, every honest $P_i$ computes the WPS-share $q^{(\ell)}_{ij} = q^{(\ell)}_i(\alpha_j)$, for $\ell = 1, \ldots, L$, at time $\Delta + T_{WPS}$. Consequently, during phase III, every honest party broadcasts an $\text{OK}$ message for every other honest party. From the validity property of $\Pi_{BC}$ in synchronous network, these $\text{OK}$ messages are received by every honest party through regular-mode at time $\Delta + T_{WPS} + T_{BC}$. 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 $\text{NOK}(\ell, i, j, q^{(\ell)}_{ij})$ message from the broadcast of any corrupt $P_j$ through regular-mode at time $\Delta + T_{WPS} + T_{BC}$ where $q^{(\ell)}_{ij} \neq Q^{(\ell)}(\alpha_j,\alpha_i)$, 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 $G_D[\mathcal{W}]$ at time $\Delta + T_{WPS} + T_{BC}$ and $D$ will find an $(n, t_s)$-star $(\mathcal{E}, \mathcal{F})$ in $G_D[\mathcal{W}]$ and broadcast $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ during phase IV. By the validity of $\Pi_{BC}$ in synchronous network, all honest parties will receive $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ through regular-mode at time $\Delta + T_{WPS} + 2T_{BC}$. Moreover, all honest parties will accept an $(\mathcal{W}, \mathcal{E}, \mathcal{F})$ and participate with input 0 in the instance of $\Pi_{BA}$. By the validity and guaranteed liveness of $\Pi_{BA}$, the output of $\Pi_{BA}$ will be 0 for every honest party at time $\Delta + T_{WPS} + 2T_{BC} + T_{BA}$. Now consider an arbitrary honest party $P_i$. Since $P_i \in \mathcal{W}$, $P_i$ outputs $s^{(\ell)}_i = q^{(\ell)}_i(0) = Q^{(\ell)}(0,\alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. □

Lemma D.10 (Correctness in Asynchronous Network). In protocol $\Pi_{VSS}$, 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 $\text{OK}(i, j)$ message, corresponding to every honest $P_j$. This is because both $P_i$ and $P_j$ eventually receive $q^{(\ell)}_i(x) = Q^{(\ell)}(x,\alpha_i)$ and $q^{(\ell)}_j(x) = Q^{(\ell)}(x,\alpha_j)$ respectively from $D$, for $\ell = 1, \ldots, L$. Moreover, $P_j$ shares $q^{(\ell)}_j(\cdot)$ during $\Pi^{(\ell)}_{WPS}$ and from the correctness of $\Pi_{WPS}$ in asynchronous network, party $P_i$ eventually computes the WPS-share $q^{(\ell)}_{ij} = q^{(\ell)}_i(\alpha_j)$ during $\Pi^{(\ell)}_{WPS}$. Moreover, $q^{(\ell)}_{ij} = q^{(\ell)}_i$ holds, for $\ell = 1, \ldots, L$. Note that every honest party participates with some input in the instance of $\Pi_{BA}$ at local time $\Delta + T_{WPS} + 2T_{BC}$. Hence almost-surely, the instance of $\Pi_{BA}$ eventually terminates for every honest party, with some common output. Now there are two possible cases:
The output of $\Pi_{BA}$ is 0: This means that at least one honest party, say $P_h$, has received $(W, E, F)$ from the broadcast of $D$ and accepted this. Hence, by the weak validity and fallback validity of $\Pi_{BC}$, all honest parties will eventually receive $(W, E, F)$ 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 $s^{(\ell)}_i = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary honest party $P_i$. There are two possible cases.

- $P_i \in W$: In this case, $P_i$ already has $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$, received from $D$.
- $P_i \notin W$: In this case, we first note that there will be at least $t_h + 1$ parties, who are eventually included in $SS^{(\ell)}_i$, for $\ell = 1, \ldots, L$. This follows from the fact that there are at least $t_h + 1$ honest parties $P_j$ in $F$. And corresponding to every honest $P_j \in F$, party $P_i$ will eventually compute the WPS-share $q^{(\ell)}_{ji}$ in the instance $\Pi_{WPS}^{(j, \ell)}$ (follows from the correctness of $\Pi_{WPS}$ in asynchronous network). We next claim that corresponding to every $P_j \in SS^{(\ell)}_i$, the value $q^{(\ell)}_{ji}$ computed by $P_i$ is the same as $Q^{(\ell)}(\alpha_j, \alpha_i)$.

The claim is obviously true for every honest $P_j \in SS^{(\ell)}_i$, so consider a corrupt $P_j \in SS^{(\ell)}_i$. We first note that the polynomial $q^{(\ell)}(x)$ shared by $P_j$ during $\Pi_{WPS}^{(j, \ell)}$ is the same as $Q^{(\ell)}(x, \alpha_k)$ and where the WPS-share $q^{(\ell)}_{jk}$ computed by $P_j$ during $\Pi_{WPS}^{(j, \ell)}$ is the same as $Q^{(\ell)}(x, \alpha_k)$ (follows from the weak commitment of $\Pi_{WPS}$ in synchronous network). Since $|\mathcal{H}| > t_h$, it implies that at least $t_h + 1$ honest parties $P_k$ have verified that $q^{(\ell)}_{jj} = Q^{(\ell)}(\alpha_j, \alpha_k)$ holds. This further implies that $q^{(\ell)}_j(x) = Q^{(\ell)}(x, \alpha_j)$. Since $P_i$ has computed WPS-share $q^{(\ell)}_{ji}$ during $\Pi_{WPS}^{(j, \ell)}$, from the weak commitment of $\Pi_{WPS}$ in synchronous network, it follows that $q^{(\ell)}_{ji} = q^{(\ell)}_{j}(\alpha_i) = Q^{(\ell)}(\alpha_i, \alpha_j) = Q^{(\ell)}(\alpha_j, \alpha_i)$, for $\ell = 1, \ldots, L$. The last equality follows since each $Q^{(\ell)}(x, y)$ is a symmetric bivariate polynomial.

The output of $\Pi_{BA}$ is 1: As mentioned earlier, since $D$ is honest, every pair of honest parties eventually broadcast $\emptyset$-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 $\Pi_{BC}$ in asynchronous network, these messages are eventually delivered to every honest party. Also from the weak consistency and fallback consistency of $\Pi_{BC}$ in asynchronous network, any $\emptyset$-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 $(E', F')$ in its consistency graph and broadcast it. From the weak validity and fallback validity of $\Pi_{BC}$ in asynchronous network, this star will be eventually received by every honest party. Moreover, $(E', F')$ 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 $s^{(\ell)}_i = Q^{(\ell)}(0, \alpha_i) = q^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. To prove the claim, consider an arbitrary honest party $P_i$. There are two possible cases.

- $P_i \in F'$: In this case, $P_i$ already has $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$, received from $D$.
- $P_i \notin F'$: In this case, we note that there will be will be at least $t_h + 1$ parties, who are eventually included in $SS^{(\ell)}_i$, for $\ell = 1, \ldots, L$, such that corresponding to every $P_j \in SS^{(\ell)}_i$, the value $q^{(\ell)}_{ji}$ computed by $P_i$ is the same as $Q^{(\ell)}(\alpha_j, \alpha_i)$. The proof for this will be similar.
Before we proceed to prove the strong commitment property in the synchronous network, we prove a helping lemma.

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

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

**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 $(W, E, F)$ from the broadcast of $D$ through regular-mode and accepted it at time $\Delta + T_{\text{WPS}} + 2T_{\text{BC}}$. From the protocol steps, the following must be true for $P_h$ at time $\Delta + T_{\text{WPS}} + T_{\text{BC}}$:

- $\exists P_j, P_k \in W$ and $\exists \ell \in \{1, \ldots, L\}$, such that $\text{NOK}(\ell, j, k, q_{jk}^{(\ell)})$ and $\text{NOK}(\ell, k, j, q_{kj}^{(\ell)})$ 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 $G_h$, $\text{deg}(P_j) \geq n - t_s$ for all $P_j \in W$ and $P_j$ has edges with at least $n - t_s$ parties from $W$.
- $(E, F)$ was an $(n, t_s)$-star in the induced subgraph $G_h[W]$, such that $\forall P_j, P_k \in W$ where the edge $(P_j, P_k)$ is present in $G_h$, the $\text{OK}(j, k)$ and $\text{OK}(k, j)$ 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 $P_j \in W$ be an honest party, who received its $t_s$-degree univariate polynomials $\{q_{ij}^{(\ell)}(x)\}_{\ell=1,\ldots,L}$ at time $\Delta + \delta$, where $\delta > 0$. Moreover, let $P_k \in W$ 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 $W$). As stated above, at time $\Delta + T_{\text{WPS}} + T_{\text{BC}}$, party $P_h$ has received the message $\text{OK}(k, j)$ from the broadcast of $P_k$ through regular-mode. From the protocol steps, $P_j$ waits till its local time becomes a multiple of $\Delta$, before it shares its polynomial $q_{ij}^{(\ell)}(\cdot)$ in the instance $\Pi_{\text{WPS}}^{(j, \ell)}$. Hence, $P_j$ must have started sharing $q_{ij}^{(\ell)}(\cdot)$ through $\Pi_{\text{WPS}}^{(j, \ell)}$ at time $c \cdot \Delta$, where $c \geq 2$. Since the network is synchronous, from the correctness of $\Pi_{\text{WPS}}$ in synchronous network, party $P_k$ will compute its WPS-share $q_{jk}^{(\ell)}$ in $\Pi_{\text{WPS}}^{(j, \ell)}$ at time $c \cdot \Delta + T_{\text{WPS}}$. Hence the result of the pair-wise consistency test with $P_j$ will be available to $P_k$ at time $c \cdot \Delta + T_{\text{WPS}}$ and so $P_k$ starts broadcasting $\text{OK}(k, j)$ message only at time $c \cdot \Delta + T_{\text{WPS}}$. Since $P_k$ is honest, from the validity property of $\Pi_{\text{BC}}$ in synchronous network, it will take exactly $T_{\text{BC}}$ time for the message $\text{OK}(k, j)$ to be received through regular-mode, once it is broadcasted. This implies that $P_h$ will receive the message $\text{OK}(k, j)$ at time $c \cdot \Delta + T_{\text{WPS}} + T_{\text{BC}}$, where $c \geq 2$. However, this is a contradiction, since the $\text{OK}(k, j)$ message has been received by $P_h$ at time $\Delta + T_{\text{WPS}} + T_{\text{BC}}$.

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

To prove the third part of the lemma, we note that since $P_h$ has received $(W, E, F)$ from the broadcast of $D$ through regular-mode at time $Δ + T_{WPS} + 2T_{BC}$, it implies that $D$ must have started broadcasting $(W, E, F)$ latest at time $Δ + T_{WPS} + T_{BC}$, which 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 $(W, E, F)$ from the broadcast of $D$ through regular-mode at time $Δ + T_{WPS} + 2T_{BC}$. Since at time $Δ + T_{WPS} + T_{BC}$, party $P_h$ has received the OK$(j, k)$ and OK$(k, j)$ messages through regular-mode from the broadcast of every $P_j, P_k \in W$ 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 $Δ + T_{WPS}$. 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 $Δ + T_{WPS} + T_{BC}$. Hence $(E, F)$ will constitute an $(n, t_s)$-star in the induced subgraph $G[V]$ of every honest party $P_i$’s consistency-graph at time $Δ + T_{WPS} + T_{BC}$ and consequently, every honest party accepts $(W, E, F)$.

\[ \square \]

**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 $\{q_i^{(ℓ)}(·)\}_{ℓ=1,\ldots,L}$, such that each honest $P_i$ eventually outputs $\{q_i^{(ℓ)}(α_i)\}_{ℓ=1,\ldots,L}$, where the following hold.

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

**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 $(W, E, F)$ from the broadcast of $D$ through regular-mode and accepted $(W, E, F)$ at time $Δ + T_{WPS} + 2T_{BC}$:** In this case, from Lemma D.11, the polynomials $q_i^{(ℓ)}(x)$ of all honest parties in $W$ are guaranteed to lie on a unique $(t_s, t_s)$-degree symmetric bivariate polynomial, say $Q^{(ℓ)}(x, y)$, for $ℓ = 1, \ldots, L$. As per the protocol steps, $P_h$ has also verified that $F ⊆ W$ (by checking that $(E, F)$ constitutes an $(n, t_s)$-star in the induced subgraph $G[H[V]]$) and hence the polynomials $q_i^{(ℓ)}(x)$ of all honest parties in $F$ also lie on $Q^{(ℓ)}(x, y)$, for $ℓ = 1, \ldots, L$. Moreover, from Lemma D.11, all honest parties accept $(W, E, F)$ at time $Δ + T_{WPS} + 2T_{BC}$. 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 $T_{VSS} = Δ + T_{WPS} + 2T_{BC} + T_{BA}$. Let $q^{(ℓ)}(·) = Q^{(ℓ)}(0, y)$ and consider an arbitrary honest party $P_i$. We wish to show that $P_i$ has $q_i^{(ℓ)}(x) = Q^{(ℓ)}(x, α_i)$ at time $T_{VSS}$, which will imply that $P_i$ outputs the VSS-share $s_i^{(ℓ)} = q_i^{(ℓ)}(0)$ at time $T_{VSS}$, which will be the same as $q^{(ℓ)}(α_i)$, for $ℓ = 1, \ldots, L$. There are two possible cases.
- \( P_i \in \mathcal{W} \): In this case, \( P_i \) has already received \( q_{ij}^{(\ell)}(x) \) from \( D \) within time \( \Delta \), for \( \ell = 1, \ldots, L \). This follows from Lemma D.11.

- \( P_i \notin \mathcal{W} \): In this case, we claim that at time \( T_{\text{VSS}} \), there will be will be at least \( t_s + 1 \) parties from \( \mathcal{F} \), who are included in \( SS_i^{(\ell)} \), for \( \ell = 1, \ldots, L \), such that corresponding to every \( P_j \in SS_i^{(\ell)} \), party \( P_i \) will have the value \( q_{ji}^{(\ell)}(x) \), which will be the same as \( \mathcal{Q}_{\ell}^{(\ell)}(\alpha_j, \alpha_i) \). Namely, there are at least \( t_s + 1 \) honest parties in \( \mathcal{F} \), who will be included in \( SS_i^{(\ell)} \) and the claim will be trivially true for those parties, due to the correctness property of \( \Pi_{\text{wps}} \) in synchronous network. On the other hand, if any corrupt \( P_j \in \mathcal{F} \) is included in \( SS_i^{(\ell)} \), then the polynomial shared by \( P_j \) during \( \Pi_{\text{wps}}^{(j, \ell)} \) 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 \( \mathcal{Q}_{\ell}^{(\ell)}(x, \alpha_j) \). Moreover, from the weak commitment of \( \Pi_{\text{wps}} \) in synchronous network, the WPS-share \( q_{ji}^{(\ell)} \) computed by \( P_i \) during \( \Pi_{\text{wps}}^{(j, \ell)} \) will be the same as \( \mathcal{Q}_{\ell}^{(\ell)}(\alpha_i, \alpha_j) \), which will be the same as \( \mathcal{Q}_{\ell}^{(\ell)}(\alpha_j, \alpha_i) \), since \( \mathcal{Q}_{\ell}^{(\ell)}(x, y) \) is a symmetric bivariate polynomial. Hence, \( P_i \) will interpolate \( q_{ij}^{(\ell)}(x) \), 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 \( \Delta + T_{\text{wps}} + 2T_{\text{BC}} \): This implies that all honest parties participate in \( \Pi_{\text{BA}} \) with input 1 and by the validity and guaranteed liveness of \( \Pi_{\text{BA}} \) in synchronous network, all honest parties obtain the output 1 from \( \Pi_{\text{BA}} \) at time \( \Delta + T_{\text{wps}} + 2T_{\text{BC}} + T_{\text{BA}} \). 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 \( P_i \)'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 \( \mathcal{OK}(*)\) messages, from the broadcast of respective parties in \( \mathcal{E}' \) and \( \mathcal{F}' \). From the protocol steps, \( T > T_{\text{VSS}} \), since the honest parties participate in the instance of \( \Pi_{\text{BC}} \) through which \( D \) has broadcasted \((\mathcal{E}', \mathcal{F}')\) only after time \( T_{\text{VSS}} \). By the consistency and fallback consistency properties of \( \Pi_{\text{BC}} \) 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 \( \mathcal{OK} \) 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 \((q_{ij}^{(\ell)}(x), q_{ij}^{(\ell)}(x))\) of every pair of honest parties \( P_i, P_h \in \mathcal{E}' \) will be pair-wise consistent and hence lie on a unique \((t_s, t_a)\)-degree symmetric bivariate polynomial, say \( \mathcal{Q}_{\ell}^{(\ell)}(x, y) \). Similarly, the univariate polynomial \( q_{ij}^{(\ell)}(x) \) of every honest party \( P_i \) in \( \mathcal{F}' \) is pair-wise consistent with the univariate polynomials \( q_{ij}^{(\ell)}(x) \) of all the honest parties in \( \mathcal{E}' \) and hence lie on \( \mathcal{Q}_{\ell}^{(\ell)}(x, y) \) as well, for \( \ell = 1, \ldots, L \). Let \( q_{ij}^{(\ell)}(\cdot) \equiv q_{ij}^{(\ell)}(0, y) \). We show that every honest \( P_i \) outputs \( q_{ij}^{(\ell)}(\alpha_i) \), for \( \ell = 1, \ldots, L \), by time \( T + 2\Delta \). For this it is enough to show that each honest \( P_i \) has \( q_{ij}^{(\ell)}(x) = \mathcal{Q}_{\ell}^{(\ell)}(x, \alpha_i) \) by time \( T + 2\Delta \), as \( P_i \) outputs \( q_{ij}^{(\ell)}(0) \), which will be the same as \( \mathcal{Q}_{\ell}^{(\ell)}(\alpha_i) \), 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 \( \mathcal{Q}_{\ell}^{(\ell)}(x, \alpha_i) \), 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 \( SS_i^{(\ell)} \), such that corresponding to every \( P_j \in SS_i^{(\ell)} \), party \( P_i \) will have the value \( q_{ij}^{(\ell)}(x) \), which will be the same as \( \mathcal{Q}_{\ell}^{(\ell)}(\alpha_j, \alpha_i) \), 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 $\Pi_{\text{BA}}$ is 0. Namely every honest $P_j \in \mathcal{F}'$ will be included in $\mathcal{SS}_i^{(t)}$. This is because $P_j$ starts broadcasting 0K messages for other parties in $\mathcal{E}'$ only after sharing its polynomial in the instance $\Pi_{\text{WPS}}^{(j,\ell)}$. Hence, by time $T + 2\Delta$, the WPS-share $q_{ji}^{(t)}$ from the instance $\Pi_{\text{WPS}}^{(j,\ell)}$ 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 $\mathcal{SS}_i^{(t)}$, 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).

\[ \square \]

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

**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 $\Pi_{\text{BA}}$ at local time $\Delta + T_{\text{WPS}} + 2T_{\text{BC}}$. Hence, almost-surely, the instance of $\Pi_{\text{BA}}$ eventually terminates for every honest party, with some common output. Now there are two possible cases:

**The output of $\Pi_{\text{BA}}$ 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 $\Delta + T_{\text{WPS}} + 2T_{\text{BC}}$. Hence, by the weak consistency and fallback consistency of $\Pi_{\text{BC}}$ 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 $Q^{(\ell)}(x,y)$, 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 $Q^{(\ell)}(x,y)$ as well. Let $q_i^{(\ell)}(\cdot) \overset{\text{def}}{=} Q^{(\ell)}(0,y)$. We claim that every honest $P_i$ will eventually have $Q^{(\ell)}(x,\alpha_i)$, for $\ell = 1, \ldots, L$. This will imply that eventually every honest $P_i$ outputs the VSS-share $s_i^{(\ell)} = Q^{(\ell)}(0,\alpha_i) = q_i^{(\ell)}(\alpha_i)$, 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 $q_i^{(\ell)}(x) = Q^{(\ell)}(x,\alpha_i)$ holds.

- **$P_i \in \mathcal{W}$ and $P_i \not\in \mathcal{F}$:** 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 $q_i^{(\ell)}(x)$ 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 $Q^{(\ell)}(x,y)$, it implies that $q_i^{(\ell)}(x) = Q^{(\ell)}(x,\alpha_i)$ holds.

- **$P_i \not\in \mathcal{W}$:** In this case, similar to the proof of Lemma D.10, one can show that $P_i$ eventually includes at least $t_a + 1$ parties from $\mathcal{F}$ in $\mathcal{SS}_i^{(t)}$, for $\ell = 1, \ldots, L$ and the value computed by $P_i$ corresponding to any $P_j \in \mathcal{SS}_i^{(t)}$ will be the same as $Q^{(\ell)}(\alpha_j,\alpha_i)$. Hence, $P_i$ will eventually interpolate $Q^{(\ell)}(x,\alpha_i)$, for $\ell = 1, \ldots, L$. 

46
- The output of $\Pi_{BA}$ is 1: Let $P_h$ be the first honest party, who obtains an output in $\Pi_{VSS}$. This means that $P_h$ has received an $(n, t_a)$-star $(E', F')$ from the broadcast of $D$, such that $(E', F')$ constitutes an $(n, t_a)$-star in $P_h$’s consistency graph. By the weak consistency and fallback consistency properties of $\Pi_{BC}$ in asynchronous network, all honest parties eventually receive $(E', F')$ from the broadcast of $D$. Moreover, since the consistency graphs are constructed based on broadcasted OK messages and since $(E', F')$ 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 $\Pi_{BC}$ in asynchronous network, the pair $(E', F')$ 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 $|E'| \geq n - 2t_a > 2t_s + (t_s - t_a) > 2t_s$, it follows that $E'$ 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 $(t_s, t_a)$-degree symmetric bivariate polynomial, say $Q^{(\ell)}(x, y)$, for $\ell = 1, \ldots, L$. Similarly, since the univariate polynomials $q_j^{(\ell)}(x)$ of every honest party $P_j$ in $F'$ is pair-wise consistent with the univariate polynomials $q_j^{(\ell)}(x)$ of all the honest parties $P_j$ in $E'$, it implies that the polynomials $q_j^{(\ell)}(x)$ of all the honest parties $P_j$ in $F'$ also lie on $Q^{(\ell)}(x, y)$ as well, for $\ell = 1, \ldots, L$. Let $q^{(\ell)}(\cdot) \overset{\text{def}}{=} Q^{(\ell)}(0, 0)$. We show that every honest $P_i$ eventually outputs $q_i^{(\ell)}(\alpha_i)$, for $\ell = 1, \ldots, L$. For this it is enough to show that each honest $P_i$ eventually gets $q_i^{(\ell)}(x) = Q^{(\ell)}(x, \alpha_i)$, as $P_i$ outputs $q_i^{(\ell)}(0)$, which will be the same as $q_i^{(\ell)}(\alpha_i)$. Consider an arbitrary honest party $P_i$. There are two possible cases.

- $P_i \in F'$: In this case, $P_i$ already has received $Q^{(\ell)}(x, \alpha_i)$ from $D$, for $\ell = 1, \ldots, L$.
- $P_i \notin F'$: Again in this case, one can show that $P_i$ eventually includes at least $t_s + 1$ parties from $F'$ in $SS_i^{(\ell)}$, for $\ell = 1, \ldots, L$ and the value computed by $P_i$ corresponding to any $P_j \in SS_i^{(\ell)}$ will be the same as $Q^{(\ell)}(\alpha_j, \alpha_i)$, for $\ell = 1, \ldots, L$. Hence, $P_i$ will eventually interpolate $Q^{(\ell)}(x, \alpha_i)$, for $\ell = 1, \ldots, L$.

□

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

Proof. The proof follows from Lemma D.7 and the fact that each party acts as a dealer and invokes an instance of $\Pi_{WPS}$ to share $L$ polynomials. Hence, the total communication cost due to $\Pi_{WPS}$ in $\Pi_{VSS}$ will be $O(n \cdot (n^2 L \log |F| + n^4 \log |F|)) = O(n^3 L \log |F| + n^5 \log |F|)$ bits, along with $n$ instances of $\Pi_{BA}$. Additionally, there is an instance of $\Pi_{BA}$ invoked in $\Pi_{VSS}$ to agree on whether some $(W, E, F)$ 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 $\Pi_{BA}$ and $\Pi_{VSS}$ 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 \( \Pi_{\text{Beaver}} \) for computing \( x \cdot y \) is presented in Fig 12.

<table>
<thead>
<tr>
<th>Protocol ( \Pi_{\text{Beaver}}({[x], [y]}, {(a), [b], [c]}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Masking Input Values (Local Computation)</strong> — parties locally compute ( [e] = [x] - [a] ) and ( [d] = [y] - [b] ).</td>
</tr>
<tr>
<td>- <strong>Publicly Reconstructing Masked Inputs (Time ( \Delta ))</strong> — each ( P_i \in \mathcal{P} ) does the following:</td>
</tr>
<tr>
<td>- Send the share of ( e ) and ( d ) to every party in ( \mathcal{P} ) and wait for ( \Delta ) time.</td>
</tr>
<tr>
<td>- Apply the ( \text{OEC}(t_s, t_s, \mathcal{P}) ) procedure on the received shares of ( e ) and ( d ) to compute ( e ) and ( d ).</td>
</tr>
<tr>
<td>- <strong>Output Computation (Local Computation)</strong> — parties locally compute ( [z] = d \cdot e + e \cdot [b] + d \cdot [a] + [c] ) and output ([z]).</td>
</tr>
</tbody>
</table>

**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 \( \Pi_{\text{Beaver}} \) achieves the following properties in the presence of up to \( t_s \) corruptions.

- **Correctness:** If the network is synchronous, then after time \( \Delta \), 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 \( \text{OEC} \) in the synchronous network (see Lemma A.1), at time \( \Delta \), every honest \( P_i \) will have \( d \) and \( e \) and hence the parties hold a \( t_s \)-sharing of \( z \) at time \( \Delta \). On the other hand, if the network is asynchronous, then from the correctness property of \( \text{OEC} \) 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. \( \square \)

E.1.2 The Triple-Transformation Protocol \( \Pi_{\text{TripTrans}} \)

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

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

The idea behind \(\Pi_{\text{TripTrans}}\) is as follows: the polynomials \(X(\cdot)\) and \(Y(\cdot)\) 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 \(X(\cdot)\) and \(Y(\cdot)\) 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 \(X(\cdot)\) and \(Y(\cdot)\) 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 \(Z(\cdot)\) 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** \(\Pi_{\text{TripTrans}}(d, \{[x^{(i)}], [y^{(i)}], [z^{(i)}]\})_{i=1,...,2d+1}\)

- **Defining \(X(\cdot)\) and \(Y(\cdot)\) Polynomials (Local Computation)** — The parties locally do the following:
  - For \(i = 1, \ldots, d + 1\), set 
    \([x^{(i)}] = [x^{(i)}], \ [y^{(i)}] = [y^{(i)}]\) and \([z^{(i)}] = [z^{(i)}]\).
  - Let \(X(\cdot)\) and \(Y(\cdot)\) be the unique \(d\)-degree polynomials passing through the points
    \([\{\alpha_i, x^{(i)}\}]_{i=1,\ldots, d+1}\) and \([\{\alpha_i, y^{(i)}\}]_{i=1,\ldots, d+1}\) respectively.
  - For \(i = d + 2, \ldots, 2d + 1\), compute \([x^{(i)}] = [X(\alpha_i)]\) from \([\{x^{(i)}\}]_{i=1,\ldots, d+1}\) by applying the corresponding Lagrange’s linear function.
  - For \(i = d + 2, \ldots, 2d + 1\), compute \([y^{(i)}] = [Y(\alpha_i)]\) from \([\{y^{(i)}\}]_{i=1,\ldots, d+1}\) by applying the corresponding Lagrange’s linear function.

- **Computing Points on the \(Z(\cdot)\) Polynomial (Time \(\Delta\))** — The parties do the following:
  - For \(i = d + 2, \ldots, 2d + 1\), participate in the instance \(\Pi_{\text{Beaver}}(\{([x^{(i)}], [y^{(i)})], ([x^{(i)}], [y^{(i)}], [z^{(i)}])\})\) of \(\Pi_{\text{Beaver}}\).
  - Let \([z^{(i)}]\) be the output obtained from this instance.
  - Output \([x^{(i)}], [y^{(i)}], [z^{(i)}]\) for \(i = 1, \ldots, 2d+1\).

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 \([x^{(i)}], [y^{(i)}], [z^{(i)}]\) be a set of \(t_s\)-shared triples. Then protocol \(\Pi_{\text{TripTrans}}\) 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 \(\Delta\), the parties have \(t_s\)-shared triples \([x^{(i)}], [y^{(i)}], [z^{(i)}]\) for \(i = 1, \ldots, 2d+1\), while in an asynchronous network, the parties eventually have \(t_s\)-shared triples \([x^{(i)}], [y^{(i)}], [z^{(i)}]\) for \(i = 1, \ldots, 2d+1\).*
  - *There exist \(d\)-degree polynomials \(X(\cdot)\), \(Y(\cdot)\) and \(2d\)-degree polynomial \(Z(\cdot)\), such that \(X(\alpha_i) = x^{(i)}, Y(\alpha_i) = y^{(i)}\) and \(Z(\alpha_i) = z^{(i)}\) holds for \(i = 1, \ldots, 2d + 1\).*
  - *(\(x^{(i)}, y^{(i)}, z^{(i)}\) 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 \((x^{(i)}, y^{(i)}, z^{(i)})\) 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 \(O(dn^2 \log |F|)\) 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 \([x^{(i)}], [y^{(i)}]\) for \(i = 1, \ldots, 2d+1\) and \(t_s\)-sharings \([z^{(i)}]\) for \(i = 1, \ldots, d+1\). If the network is synchronous, then from the correctness property of \(\Pi_{\text{Beaver}}\) in the synchronous network (see Lemma E.1), it follows that after time \(\Delta\), all honest parties will have their respective output in all the \(d\) instances of \(\Pi_{\text{Beaver}}\). Hence after time \(\Delta\), the parties have \(t_s\)-sharings \([z^{(i)}]\) for \(i = d + 2, \ldots, 2d + 1\). On the other hand, if the network is asynchronous, then from the correctness property of \(\Pi_{\text{Beaver}}\) in the asynchronous network (see Lemma E.1), it follows that all the \(d\) instances of \(\Pi_{\text{Beaver}}\) eventually produce an output for each honest party. Hence the parties eventually compute the \(t_s\)-sharings \([z^{(i)}]\) for \(i = d + 2, \ldots, 2d + 1\) and hence eventually obtain their output.
Next consider any \(i \in \{1, \ldots, d + 1\}\). Since \((x^{(i)}, y^{(i)}, z^{(i)}) = (x^{(i)}, y^{(i)}, z^{(i)})\), it follows that \((x^{(i)}, y^{(i)}, z^{(i)})\) 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 \([z^{(i)}]\) is the output of the instance \(\Pi_{\text{Beaver}}([x^{(i)}], [y^{(i)}]), ([x^{(i)}], [y^{(i)}]), [z^{(i)}]_d\), it follows from the correctness property of \(\Pi_{\text{Beaver}}\) that \(z^{(i)} = x^{(i)} \cdot y^{(i)}\) 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 \(X(\cdot)\) and \(Y(\cdot)\) defined in the protocols are \(d\)-degree polynomials, as they are defined through \(d + 1\) distinct points \(\{(\alpha_i, x^{(i)})\}_{i=1, \ldots, d+1}\) and \(\{(\alpha_i, y^{(i)})\}_{i=1, \ldots, d+1}\) respectively. On the other hand, \(Z(\cdot)\) is a \(2d\)-degree polynomial, as it is defined through the \(2d + 1\) distinct points \(\{(\alpha_i, z^{(i)})\}_{i=1, \ldots, 2d+1}\).

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 \((x^{(i)}, y^{(i)}, z^{(i)})\) is also random from the point of view of the adversary, since \((x^{(i)}, y^{(i)}, z^{(i)}) = (x^{(i)}, y^{(i)}, z^{(i)})\). 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 \(\Pi_{\text{Beaver}}\) (see Lemma E.1), it follows that no additional information is learnt about \((x^{(i)}, y^{(i)}, z^{(i)})\).

The communication complexity follows from the fact that there are \(d\) instances of \(\Pi_{\text{Beaver}}\) invoked in the protocol. \(\square\)

### E.1.3 Agreement on a Common Subset

In the protocol \(\Pi_{\text{ACS}}\), every party \(P_i\) will have a set of \(L\) number of \(t_s\)-degree polynomials, such that \(\Pi_V\) would like to act as a dealer and distribute points on these polynomials by invoking an instance of \(\Pi_{\text{VSS}}\). As corrupt parties may not invoke their instances of \(\Pi_{\text{VSS}}\), 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 \(\Pi_{\text{ACS}}\) lets the parties agree on a common subset \(CS\) 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 \(CS\).

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 \(\Pi_{\text{VSS}}\) to distribute points on its polynomial. After the time-out \(T_{\text{VSS}}\) of \(\Pi_{\text{VSS}}\) (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 \(\Pi_{\text{VSS}}\) in which they have received an output and then start participating in \(n\) instances of \(\Pi_{\text{BA}}\) where the \(j^{th}\) instance is to decide whether \(P_j\) should be included in \(CS\). The input criteria for these \(\Pi_{\text{BA}}\) instances is the following: if a party has received an output in the \(\Pi_{\text{VSS}}\) instance with \(P_j\) as the dealer, then the party starts participating with input 1 in the corresponding \(\Pi_{\text{BA}}\) instance. Now once 1 is obtained as the output from \(n - t_s\) instances of \(\Pi_{\text{BA}}\), then the parties start participating with input 0 in any of the remaining \(\Pi_{\text{BA}}\) instances for which the parties may have not provided any input yet. Finally, once output is obtained from all the \(n\) instances of \(\Pi_{\text{BA}}\), party \(P_j\) is included in \(CS\) iff the output of the corresponding \(\Pi_{\text{BA}}\) instance is 1. Since the parties wait for time \(T_{\text{VSS}}\) before starting the \(\Pi_{\text{BA}}\) instances, it is ensured that all honest dealers are included in \(CS\) in a synchronous network.

<table>
<thead>
<tr>
<th>Protocol (\Pi_{\text{ACS}})</th>
</tr>
</thead>
</table>

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

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

- **Synchronous Network**: The following is achieved in the presence of up to $t_s$ corruptions.
  - **Correctness**: at time $T_{\text{ACS}} = T_{\text{VSS}} + 2T_{\text{BA}}$ 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 $f_j^*(\cdot)$, such that every honest $P_i$ has $f_j^*(\alpha_i)$.

- **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 $n - t_a$, 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 $f_j^*(\cdot)$, such that every honest $P_i$ eventually has $f_j^*(\alpha_i)$.

- **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 $O(n^6 \log |\mathbb{F}|)$ bits and invokes $O(n^2)$ instances of $\Pi_{\text{BA}}$.

**Proof.** The privacy property simply follows from the privacy property of $\Pi_{\text{VSS}}$, while communication complexity follows from the communication complexity of $\Pi_{\text{VSS}}$ (see Theorem 4.2) and the fact that $O(n)$ instances of $\Pi_{\text{VSS}}$ 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 $T_{\text{VSS}}$ during $\Pi_{\text{VSS}}^{(j)}$, which follows from the correctness of $\Pi_{\text{VSS}}$ in synchronous network (see Theorem 4.2). Consequently, at time $T_{\text{VSS}}$, 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 $\Pi_{\text{BA}}^{(j)}$ at time $T_{\text{VSS}}$. From the validity and guaranteed liveness of $\Pi_{\text{BA}}$ in the synchronous network, it follows that at time $T_{\text{VSS}} + T_{\text{BA}}$, every honest $P_i$ obtains the output 1 during the instance $\Pi_{\text{BA}}^{(j)}$, corresponding to every $P_j \in \mathcal{H}$. Consequently, at time $T_{\text{VSS}} + T_{\text{BA}}$, every honest party will start participating in the remaining $\Pi_{\text{BA}}$ instances for which no input has been provided yet (if there are any) and from the guaranteed liveness and consistency of $\Pi_{\text{BA}}$ in the synchronous network, these $\Pi_{\text{BA}}$ instances will produce common outputs for every honest party at time $T_{\text{ACS}} = T_{\text{VSS}} + 2T_{\text{BA}}$. Hence, at time $T_{\text{ACS}}$, 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 \( \Pi^{(j)}_{BA} \) produces the output 1. This further implies that at least one honest \( P_i \) must have obtained some output from the instance \( \Pi^{(j)}_{VSS} \) by time \( T_{VSS} + T_{BA} \) (implying that \( P_j \in \mathcal{C}_i \)) and participated with input 1 in the instance \( \Pi^{(j)}_{BA} \). This is because if at time \( T_{VSS} + T_{BA} \), 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 \( \Pi^{(j)}_{BA} \) from time \( T_{VSS} + T_{BA} \). Then, from the validity of \( \Pi_{BA} \) in the synchronous network, every honest party would obtain the output 0 in the instance \( \Pi^{(j)}_{BA} \) and hence \( P_j \) will not be present in \( \mathcal{CS} \), which is a contradiction. Now if \( P_i \) has obtained some output from \( \Pi^{(j)}_{VSS} \) at time \( T_{VSS} + T_{BA} \), then from the strong-commitment of \( \Pi_{VSS} \), it follows that \( P_j \) has some \( t_s \)-degree polynomial, say \( f_{j}^*(\cdot) \), such that every honest party \( P_i \) obtains \( f_{j}^*(\alpha_i) \) by time \( T_{VSS} + T_{BA} + 2\Delta \). Since \( 2\Delta < T_{BA} \), it follows that at time \( T_{ACS} \), every honest \( P_i \) has \( f_{j}^*(\alpha_i) \), 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 \( \Pi_{BA} \) 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 \( \Pi^{(j)}_{VSS} \), which follows from the correctness of \( \Pi_{VSS} \) in asynchronous network (see Theorem 4.2). So even if the corrupt parties \( P_j \) do not invoke their respective \( \Pi^{(j)}_{VSS} \) instances, there will be at least \( n - t_s \) instances of \( \Pi_{BA} \) in which all honest parties eventually participate with input 1. Consequently, almost-surely these \( \Pi_{BA} \) instances eventually produce the output 1. Hence, all honest parties eventually participate with some input in the remaining \( \Pi_{BA} \) instances, which almost-surely produce some output for every honest party eventually. From the security of \( \Pi_{BA} \) 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 \( \Pi^{(j)}_{BA} \), which further implies that at least one honest \( P_i \) participated with input 1 in \( \Pi^{(j)}_{BA} \) after receiving some output in the instance \( \Pi^{(j)}_{VSS} \). Now if \( P_j \) is honest, then the correctness of \( \Pi_{VSS} \) in asynchronous network (see Theorem 4.2) guarantees that every honest party \( P_i \) eventually obtains the output \( f_j(\alpha_i) \) during \( \Pi^{(j)}_{VSS} \). On the other hand, if \( P_j \) is corrupt, then strong commitment of \( \Pi_{VSS} \) in asynchronous network (see Theorem 4.2) guarantees that there exists some \( t_s \)-degree polynomial, say \( f_{j}^*(\cdot) \), such that every honest party \( P_i \) eventually obtains the output \( f_{j}^*(\alpha_i) \) in the instance \( \Pi^{(j)}_{VSS} \).

Protocol \( \Pi_{ACS} \) for Multiple Polynomials: Protocol \( \Pi_{ACS} \) 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 \( \ell \) polynomials in its corresponding instance of \( \Pi_{VSS} \). The rest of the protocol steps remain the same. The protocol will incur a communication of \( O(n^4L\log |\mathbb{F}| + n^6\log |\mathbb{F}|) \) bits and invokes \( O(n^2) \) instances of \( \Pi_{BA} \).

E.1.4 Triple-Sharing Protocol

Protocol \( \Pi_{TripSh} \) 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 \( \Pi_{TripSh} \) 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 \( \PiVSS \). 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 \( \PiTripTrans \) and “transform” these shared triples into “co-related” shared triples, constituting distinct points on the triplet of polynomials \( (X(\cdot), Y(\cdot), Z(\cdot)) \), which are guaranteed to be obtained from \( \PiTripTrans \). Then, to check if all the triples shared by D are multiplication-triples, it is sufficient to verify if \( Z(\cdot) = X(\cdot) \cdot Y(\cdot) \) holds. To verify the latter, we incorporate a mechanism which enables the parties to publicly learn if \( Z(\alpha_j) = X(\alpha_j) \cdot Y(\alpha_j) \) holds under \( P_j \)’s “supervision” in such a way that if \( P_j \) is honest, then the supervised verification of the triplet \( (X(\alpha_j), Y(\alpha_j), Z(\alpha_j)) \) 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 \( Z(\cdot) = X(\cdot) \cdot Y(\cdot) \) holds. This is because \( Z(\cdot) \) is a \( 2t_s \)-degree polynomial. Upon confirming that \( Z(\cdot) = X(\cdot) \cdot Y(\cdot) \) 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 \( X(\cdot), Y(\cdot) \) and \( Z(\cdot) \), 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 \( \PiACS \), 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 \( \PiACS \) is invoked in parallel with D’s invocation of \( \PiVSS \). Through the instance of \( \PiACS \), the parties agree upon a set \( W \) of at least \( n - t_s \) supervisors, whose shared verification-triples are used to verify the points on the polynomials \( X(\cdot), Y(\cdot) \) and \( Z(\cdot) \). Namely, if \( P_j \in W \) 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 \( Z(\alpha_j) - X(\alpha_j) \cdot Y(\alpha_j) = 0 \) holds. For this, the parties recompute \( X(\alpha_j) \cdot Y(\alpha_j) \) in a shared fashion using Beaver’s method, by deploying the shared verification-triple \( (u^{(j)}, v^{(j)}, w^{(j)}) \). If \( Z(\alpha_j) - X(\alpha_j) \cdot Y(\alpha_j) \) 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 \( (X(\alpha_j), Y(\alpha_j), Z(\alpha_j)) \) is a multiplication-triple and discard D if the triplet 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 \( W \) (follows from the correctness of \( \PiACS \) in a synchronous network) and hence, there will be at least \( n - t_s > 2t_s + 1 \) honest supervisors in \( W \). On the other hand, even in an asynchronous network, there will be at least \( n - t_s - t_a > 2t_s \) honest supervisors in \( W \). Hence if a corrupt D is not discarded, then it is guaranteed that D has shared multiplication-triples.

**Protocol \( \PiTripSh \)**

- **Phase I — Sharing Triples and Verification-Triples (Time \( T_{\PiACS} \)):**
  - 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 \( \PiVSS \) with input \( \{f_{x^{(j)}}(\cdot), f_{y^{(j)}}(\cdot), f_{z^{(j)}}(\cdot)\}_{j=1,\ldots,2t_s+1} \) and the parties in \( P \) participate in this instance.
Lemma E.4. Protocol $\Pi_{\text{TripSh}}$ achieves the following properties.

- **Correctness:** If $D$ is honest, then the following hold:
  - If the network is synchronous, then after time $T_{\text{TripSh}} = T_{\text{ACS}} + 4\Delta$, 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 \( O(n^6 \log |F|) \) bits and involves \( O(n^2) \) instances of \( \Pi_{BA} \).

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 \( \Pi_{VSS} \) 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 \( \Pi_{VSS} \) (see Theorem 4.2). Since the instance of \( \Pi_{ACS} \) is invoked in parallel with the instance of \( \Pi_{VSS} \) invoked by D, at time \( T_{ACS} \), all honest parties will have a common subset \( \mathcal{W} \) from the instance of \( \Pi_{ACS} \) with every honest \( P_j \) being present in the \( \mathcal{W} \). This follows from the correctness property of \( \Pi_{ACS} \) in synchronous network (see Lemma E.3). At time \( T_{ACS} + \Delta \), the multiplication-triples shared by D will be transformed and parties will have \( t_s \)-shared multiplication-triples \( \{(x^{(j)}, y^{(j)}, z^{(j)})\}_{j=1, \ldots, 2t_s+1} \) and there will exist \( t_s \)-degree polynomials \( X(\cdot), Y(\cdot) \) and \( 2t_s \)-degree polynomial \( Z(\cdot) \) where \( Z(\cdot) = X(\cdot) \cdot Y(\cdot) \) holds. This follows from the correctness property of \( \Pi_{TripTrans} \) in a synchronous network (see Lemma E.2).

Next, corresponding to every honest \( P_j \in \mathcal{W} \), the value \( \zeta^{(j)} \) will be the same as \( x^{(j)} \cdot y^{(j)} \), which follows from the correctness of \( \Pi_{Beaver} \) and the fact that the corresponding verification-triple \( (u^{(j)}, v^{(j)}, w^{(j)}) \) will be a multiplication-triple. Hence, \( \gamma^{(j)} = z^{(j)} - \zeta^{(j)} \) will be 0 and so each honest \( P_j \) will set flag\(_{z^{(j)}} \) to 0, without suspecting and reconstructing the triple \( (x^{(j)}, y^{(j)}, z^{(j)}) \). Moreover, in this case, no additional information about \( x^{(j)}, y^{(j)}, z^{(j)} \) is revealed, which follows from the privacy property of \( \Pi_{Beaver} \) 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 \( P_j \) will reset flag\(_{z^{(j)}} \) to 0 after reconstructing the corresponding suspected-triple \( (x^{(j)}, y^{(j)}, z^{(j)}) \), since it will be a multiplication-triple. The process of computing \( \gamma^{(j)} \) and the difference \( \gamma^{(j)} \) will take \( 2\Delta \) time and additionally \( \Delta \) time might be required to publicly reconstruct suspected-triples corresponding to corrupt \( P_j \in \mathcal{W} \). Hence, at time \( T_{ACS} + 4\Delta \), each honest \( P_j \) sets flag\(_{z^{(j)}} = 1 \) 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 \( (X(\beta), Y(\beta), Z(\beta)) \). Since at most \( t_s \) triples \( (x^{(j)}, y^{(j)}, z^{(j)}) \) 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 \( X(\cdot), Y(\cdot) \) and \( Z(\cdot) \) polynomials.

This further implies that \( (X(\beta), Y(\beta), Z(\beta)) \) will be random from the point of view of the adversary, since \( X(\cdot), Y(\cdot) \) are \( t_s \)-degree polynomials and \( Z(\cdot) \) 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 \( \Pi_{VSS}, \Pi_{ACS}, \Pi_{Beaver} \) 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 \( \Pi_{VSS} \) or not, the instance of \( \Pi_{ACS} \) 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 \( \Pi_{ACS} \) in
This implies that at least one honest strong-commitment holds trivially. So consider the case when some honest party obtains an output. Let T be the time at which $P_h$ has the output for the instance of $\Pi_{VSS}$ invoked by D. Note that $T \geq T_{VSS}$ and T could be greater than $T_{ACS}$, as a corrupt D may delay the start of the instances of $\Pi_{VSS}$. From the strong-Commitment property of $\Pi_{VSS}$ 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 $\Pi_{VSS}$ invoked by D. Hence at time $T + 2\Delta$, there are $2t_s + 1$ triples which are $t_s$-shared by D.

If $T \leq T_{ACS} - 2\Delta$, 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 $\Pi_{TripTrans}$ will produce its output at time $T_{ACS} + \Delta$. The follow up instances of $\Pi_{Beaver}$ to recompute the products will take $\Delta$ time, followed by $\Delta$ time for publicly reconstructing the difference values $\gamma^{(j)}$. Additionally, the parties may take $\Delta$ time to publicly reconstruct any suspected triples. Hence in this case, all honest parties will have their respective output shares at time $T_{ACS} + 4\Delta$.

On the other hand, if $T > T_{ACS} - 2\Delta$, then each honest party obtains its output from the instance of $\Pi_{TripTrans}$, either at time $T + \Delta$ or at time $T + 3\Delta$. Then, each honest party obtains its output from the instances of $\Pi_{Beaver}$, 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 $2\Delta$ time within which the honest parties obtain their output in the protocol, due to a possible difference of $2\Delta$ time in getting the output in the instances of $\Pi_{VSS}$ invoked by the corrupt D.

If the triples shared by D during the instances of $\Pi_{VSS}$ 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 $Z(\cdot) \neq X(\cdot) \cdot Y(\cdot)$, where $X(\cdot), Y(\cdot)$ are the $t_s$-degree polynomials and $Z(\cdot)$ is the $2t_s$-degree polynomial, which are guaranteed to exist from the protocol $\Pi_{TripTrans}$. Let $P_j$ be an honest party, such that $Z(\alpha_j) \neq X(\alpha_j) \cdot Y(\alpha_j)$. This further implies that the transformed triple $(x^{(j)}, y^{(j)}, z^{(j)})$ 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 $Z(\alpha_j) = X(\alpha_j) \cdot Y(\alpha_j)$ holds corresponding to every honest $P_j$, then it implies that $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$ holds (due to the degrees of the respective polynomials), which is a contradiction.

We next show that each honest $P_j$ will set $\text{flag}_{i}^{(j)} = 1$ and hence $\text{flag}_{i} = 1$. For this, we note that $P_j \in \mathcal{W}$. This follows from the correctness property of $\Pi_{ACS}$, which guarantees that all honest parties (and not just $P_j$) will be present in $\mathcal{W}$. Since the verification-triple $(w^{(j)}, v^{(j)}, w^{(j)})$ shared by $P_j$ will be a multiplication-triple, from the correctness property of $\Pi_{Beaver}$ in synchronous network, it follows that $\gamma^{(j)} = x^{(j)} \cdot y^{(j)}$ holds. But since $z^{(j)} \neq x^{(j)} \cdot y^{(j)}$, it follows that $\gamma^{(j)} = z^{(j)} - \gamma^{(j)} \neq 0$. Consequently, the parties will publicly reconstruct the suspected-triple $(x^{(j)}, y^{(j)}, z^{(j)})$ and find that it is not a multiplication-triple. Hence each honest $P_j$ will set $\text{flag}_{i}^{(j)} = 1$ and hence $\text{flag}_{i} = 1$. 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 $\Pi_{VSS}$ and correctness properties of $\Pi_{ACS}$ and $\Pi_{Beaver}$ 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 $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, through their respective verification-triples.

56
The communication complexity follows from communication complexity of $\Pi_{\text{ACS}}, \Pi_{\text{VSS}}, \Pi_{\text{TripTrans}}$ and $\Pi_{\text{Beaver}}$.

**Protocol $\Pi_{\text{TripSh}}$ for Sharing $L$ Multiplication-Triples:** Protocol $\Pi_{\text{TripSh}}$ 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 $\Pi_{\text{VSS}}$, while each party $P_j$ will need to select $L$ verification-triples during the instance of $\Pi_{\text{ACS}}$. Moreover, there will be $L$ instances of $\Pi_{\text{TripTrans}}$ to transform $D$’s shared triples, resulting in $L$ triplets of shared polynomials $(X(\cdot), Y(\cdot), Z(\cdot))$, each of which is independently verified by performing supervised verification. To avoid repetition, we do not provide the formal details. The modified $\Pi_{\text{TripSh}}$ protocol will incur a communication of $O(n^4 L \log |F| + n^6 \log |F|)$ bits and invokes $O(n^2)$ instances of $\Pi_{\text{BA}}$.

**E.1.5 The Triple-Extraction Protocol**

Protocol $\Pi_{\text{TripExt}}$ (Fig 16) takes as input a publicly-known subset $CS$ of $2d + 1$ parties, where $d \geq t_s$ and where it will be ensured that each party $P_j \in CS$ 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 $\Pi_{\text{TripTrans}}$ 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 $(X(\cdot), Y(\cdot), Z(\cdot))$ be the triplet of shared polynomials which is guaranteed to exist after $\Pi_{\text{TripTrans}}$. From the privacy property of $\Pi_{\text{TripTrans}}$, 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 $\Pi_{\text{TripExt}}(CS, \{(x^{(j)}), (y^{(j)}), (z^{(j)})\}_{j \in CS})$**

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

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

**Lemma E.5.** Let $CS$ be a set of $2d + 1$ parties where $d \geq t_s$, such that each party $P_j \in CS$ 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 $\Pi_{\text{TripExt}}$ achieves the following properties.
- **Correctness**: If the network is synchronous, then after time $\Delta$ the parties have $t_s$-shared multiplication-triples $\{a_j, b_j, c_j\}_{j=1,\ldots,d+1-t_s}$. If the network is asynchronous, then the parties eventually have $t_s$-shared multiplication-triples $\{a_j, b_j, c_j\}_{j=1,\ldots,d+1-t_s}$.

- **Privacy**: The triples $\{a_j, b_j, c_j\}_{j=1,\ldots,d+1-t_s}$ will be random from the point of view of the adversary.

- **Communication Complexity**: The protocol incurs a communication of $O(dn^2 \log |\mathcal{F}|)$ bits.

*Proof.* If the network is synchronous, then from the correctness property of $\Pi_{\text{TripTrans}}$ in the synchronous network (see Lemma E.2), it follows that after time $\Delta$, the honest parties have $t_s$-shared triples $\{x_j, y_j, z_j\}_{P_j \in \mathcal{CS}}$. 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 $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$ holds, where $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ are the $d, d$ and $2d$-degree polynomials respectively, which are guaranteed to exist from $\Pi_{\text{TripTrans}}$, such that $X(\alpha_j) = x_j, Y(\alpha_j) = y_j$ and $Z(\alpha_j) = z_j$ holds for every $j$, such that $P_j \in \mathcal{CS}$. It now follows that the honest parties output the $t_s$-shared triples $\{(a_j, b_j, c_j)\}_{j=1,\ldots,d+1-t_s}$ after time $\Delta$, where $X(\beta_j) = a_j, Y(\beta_j) = b_j$ and $Z(\beta_j) = c_j$. Moreover, the triples will be multiplication-triples, because $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$ 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 $\Pi_{\text{TripTrans}}$ 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 $\{x_j, y_j, z_j\}_{P_j \in \mathcal{CS}}$, which follows from the privacy property of $\Pi_{\text{TripTrans}}$. This implies that adversary will know at most $t_s$ distinct points on the polynomials $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, leaving $d+1-t_s$ degrees of freedom on the these polynomials. This further implies that the multiplication-triples $\{(X(\beta_j), Y(\beta_j), Z(\beta_j))\}_{j=1,\ldots,d+1-t_s}$, which are the same as $\{a_j, b_j, c_j\}_{j=1,\ldots,d+1-t_s}$, 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 $\{x_j, y_j, z_j\}_{P_j \in \mathcal{CS}}$ which are unknown to the adversary and the output multiplication-triples $\{a_j, b_j, c_j\}_{j=1,\ldots,d+1-t_s}$. 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 $\Pi_{\text{TripTrans}}$. \hfill $\square$

### E.2 The Triple-Generation Protocol

The triple-generation protocol $\Pi_{\text{TripGen}}$ is presented in Fig 17. In the protocol, each party acts as a dealer and invokes an instance of $\Pi_{\text{TripSh}}$, so that $L = \frac{cM}{2d+1-t_s}$ random multiplication-triples are shared on its behalf. As corrupt dealers may not invoke their instances of $\Pi_{\text{TripSh}}$ (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 $\Pi_{\text{BA}}$ (similar to the protocol $\Pi_{\text{ACS}}$). 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 $\Pi_{\text{TripExt}}$ 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 $2d+1$.

<table>
<thead>
<tr>
<th>Protocol $\Pi_{\text{TripGen}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L \overset{\text{def}}{=} \frac{cM}{2d+1-t_s}$</td>
</tr>
</tbody>
</table>

58
Phase 1 — Sharing Random Multiplication-Triple (Time $T_{TripSh}$): Each $P_i \in P$ does the following.
- Act as a dealer $D$ and invoke an instance $\Pi_{TripSh}^{(j)}$ of $\Pi_{TripSh}$ so that $L$ random multiplication-triples are shared on $P_i$’s behalf.
- For $j = 1, \ldots, n$, participate in the instance $\Pi_{TripSh}^{(j)}$ invoked by $P_j$ and wait for time $T_{TripSh}$.
- Initialize a set $C_i = \emptyset$ and include $P_j$ in $C_i$, if an output is obtained from $\Pi_{TripSh}^{(j)}$.

Phase II — Agreement on a Common Subset of Triple-Providers (Maximum time $2T_{BA}$): Each $P_i \in P$ does the following.
- For $j = 1, \ldots, n$, participate in an instance of $\Pi_{BA}^{(j)}$ of $\Pi_{BA}$ with input 1, if $P_j \in C_i$.
- Once $n - t_s$ instances of $\Pi_{BA}$ have produced an output 1, then participant with input 0 in all the $\Pi_{BA}$ instances $\Pi_{BA}^{(j)}$ such that $P_j \notin C_i$.
- Once all the $n$ instances of $\Pi_{BA}$ have produced a binary output, then set $CS$ to be the set of first $n - t_s$ parties $P_j$, such that 1 is obtained as the output in the instance $\Pi_{BA}^{(j)}$.

Phase III — Extracting Random Multiplication-Triples (Time $\Delta$): The parties do the following.
- For every $P_j \in CS$, 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 $\Pi_{TripExt}(CS, \{[x^{(j,\ell)}], [y^{(j,\ell)}], [z^{(j,\ell)}]\}_{P_j \in CS})$ of $\Pi_{TripExt}$ to obtain the output $\{[a^{(j,\ell)}], [b^{(j,\ell)}], [c^{(j,\ell)}]\}_{j = \frac{n-s-1}{2} + 1 - t_s, \ell=1,\ldots,L}$.
- Output the shared triples $\{[a^{(j,\ell)}], [b^{(j,\ell)}], [c^{(j,\ell)}]\}_{j = \frac{n-s-1}{2} + 1 - t_s, \ell=1,\ldots,L}$.

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

Theorem E.6. Protocol $\Pi_{TripGen}$ achieves the following properties.
- Correctness:
  - In a synchronous network, by time $T_{TripGen} = T_{TripSh} + 2T_{BA} + \Delta$, the honest parties output a $t_s$-sharing of $CM$ multiplication-triples.
  - In an asynchronous network, almost-surely, the honest parties eventually output a $t_s$-sharing of $CM$ multiplication-triples.
- Privacy: The view of the adversary remains independent of the output multiplication-triples.
- Communication Complexity: The protocol incurs a communication cost of $\mathcal{O}(\frac{n^5}{\log |\mathcal{F}|} + n^7 \log |\mathcal{F}|)$ bits and invokes $\mathcal{O}(n^3)$ instances of $\Pi_{BA}$. For $t_a = \Omega(n)$, the protocol incurs a communication cost of $\mathcal{O}(n^4CM \log |\mathcal{F}| + n^7 \log |\mathcal{F}|)$, along with $\mathcal{O}(n^3)$ instances of $\Pi_{BA}$.

Proof. Let the network be synchronous with up to $t_s$ corruptions. Let $H$ be the set of parties, where $|H| \geq n - t_s$. Corresponding to each $P_j \in H$, 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 $\Pi_{TripSh}$ in a synchronous network (see Lemma E.4). Consequently, the set $C_i$ will be of size at least $n - t_s$ for every honest $P_i$. At time $T_{TripSh}$, corresponding to each $P_j \in H$, each honest $P_i$ participates with input 1 in the instance $\Pi_{BA}^{(j)}$. It then follows from the validity and guaranteed liveness of $\Pi_{BA}$ in the synchronous network that corresponding to every $P_j \in H$, every honest $P_i$ obtains the output 1 in the instance $\Pi_{BA}^{(j)}$ at time $T_{TripSh} + T_{BA}$. Consequently, after time $T_{TripSh} + T_{BA}$, every honest party will start participating in the remaining $\Pi_{BA}$ instances for which no input has been provided yet (if there are any) and from the guaranteed liveness and consistency of $\Pi_{BA}$ in the synchronous network, these $\Pi_{BA}$ instances will produce common outputs for every honest party at time $T_{TripSh} + 2T_{BA}$. Consequently, by time $T_{TripSh} + 2T_{BA}$, every honest party has a common $CS$ of size $n - t_s$.

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

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

From the privacy property of $\Pi_{TripSh}$, it follows that the multiplication-triples which are $t_s$-shared on behalf of the honest parties $P_j \in CS$ will be random from the point of view of the adversary. It then follows from the privacy property of $\Pi_{TripExt}$ (Lemma E.4) that the $t_s$-shared multiplication-triples generated from each instance of $\Pi_{TripExt}$ 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 $\Pi_{TripSh}$ and $\Pi_{TripExt}$, and because $\frac{n-t_s-1}{2} + 1 - t_s \geq \frac{t_s}{2} + 1$.

\[\square\]

## F Properties of the Circuit-Evaluation Protocol

Protocol $\Pi_{CirEval}$ for securely evaluating the circuit $\text{cir}$ is presented in Fig 18.

```
Protocol $\Pi_{CirEval}$
```

- **Preprocessing and Input-Sharing (Time $T_{TripGen}$)** — The parties do the following:
  - Each $P_i \in P$ 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 $\Pi_{ACS}$ with input $f_{x^{(i)}}(\cdot)$. Let $CS$ be the common subset of parties obtained from $\Pi_{ACS}$, where $|CS| \geq n - t_s$. Corresponding to every $P_j \notin CS$, set $x^{(j)} = 0$ and set $[x^{(j)}]$ to a default $t_s$-sharing of 0.
  - In parallel, participate in an instance of $\Pi_{TripGen}$. Let $\{(a^{(j)}), [b^{(j)}], [c^{(j)}]\}_{j=1,...,c_M}$ be the $t_s$-shared multiplication-triples obtained from $\Pi_{TripGen}$.

- **Circuit Evaluation (Time $D_M \cdot Δ$)** — 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 $\ldots$
Consider a proof. Let $D$ be an arithmetic circuit $\text{cir}$ whose multiplicative depth is represented by an arithmetic circuit $\Pi_n$. Theorem 6.1. ACS property of $\Pi_n$ with $P$ and where $|CS| \geq CS$. As $CS$ outside running the corresponding instances of $\Pi_n$ evaluated locally, while all the independent copies will take $\Delta$ time for every party to reconstruct $y$. From the properties of $OEC(t_s, t_s, P)$, it will take $\Delta$ time for every party to reconstruct $y$. Hence, during the termination phase, all honest parties will send a ready message for $y$. Since there are at least $2t_s + 1$ honest parties, every 

**Figure 18:** A best-of-both-worlds perfectly-secure protocol for securely evaluating the arithmetic circuit $\text{cir}$. 

**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 $\text{cir}$ over $\mathbb{F}$ consisting of $c_M$ number of multiplication gates, and whose multiplicative depth is $D_M$. Then, $\Pi_{\text{CirEval}}$ incurs a communication cost of $O\left(\frac{n^3}{2 - 1}c_M \log |\mathbb{F}| + n^7 \log |\mathbb{F}|\right)$ bits (which is $O(n^4c_M \log |\mathbb{F}| + n^7 \log |\mathbb{F}|)$ bits if $t_a = \Omega(n)$), invokes $O(n^3)$ instances of $\Pi_{\text{BA}}$ 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 $P_j \notin CS$, such that $|CS| \geq n - t_s$ and every honest $P_j \in CS$; here $k$ is the constant from Lemma 3.2, as determined by the underlying (existing) perfectly-secure ABA protocol $\Pi_{\text{ABA}}$. (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 $P_j \notin CS$ and where $|CS| \geq n - t_s$.

- **Privacy:** The view of the adversary will be independent of the inputs of the honest parties in $CS$.

**Proof.** Consider a synchronous network with up to $t_s$ corruptions. From the correctness property of $\Pi_{\text{TripGen}}$ in the synchronous network (see Theorem E.6), at time $T_{\text{TripGen}}$, the (honest) parties have $c_M$ number of $t_s$-shared multiplication-triples from the instance of $\Pi_{\text{TripGen}}$. From the correctness property of $\Pi_{\text{ACS}}$ in the synchronous network (see Lemma E.3), at time $T_{\text{ACS}}$, the (honest) parties have a common subset $CS$ from the instance of $\Pi_{\text{ACS}}$, where all honest parties will be present in $CS$ and where $|CS| \geq n - t_s$. Moreover, corresponding to every $P_j \in CS$, 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 $CS$ will be known publicly, the parties take a default $t_s$-sharing of 0 on the behalf of the parties outside $CS$ by considering $x^{(j)} = 0$. Since $T_{\text{ACS}} < T_{\text{TripGen}}$, it follows that at time $T_{\text{TripGen}}$, 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 $\Pi_{\text{Beaver}}$ in parallel, where each such instance requires $\Delta$ time. From the correctness property of $\Pi_{\text{Beaver}}$ 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 $\Delta$ time for every party to reconstruct $y$. Hence, during the termination phase, all honest parties will send a 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 $T_{\text{TripGen}} + (D_M + 2) \cdot \Delta$. By substituting the values of $T_{\text{TripGen}}, T_{\text{TripSh}}, T_{\text{VSS}}, T_{\text{WPS}}, T_{\text{BC}}, T_{\text{BA}}, T_{\text{SBA}}$ and $T_{\text{ABA}}$ and by noting that all instances of $\Pi_{\text{BC}}$ in $\Pi_{\text{CirEval}}$ 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 $\Pi_{\text{ABA}}$.

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 $\Pi_{\text{TripGen}}, \Pi_{\text{ACS}}, \Pi_{\text{Beaver}}$ and $\Pi_{\text{RecPriv}}$ 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 $\Pi_{\text{ACS}}$, corresponding to every honest $P_j \in CS$, the input $x^{(j)}$ will be random from the point of view of the adversary. Moreover, from the privacy property of $\Pi_{\text{TripGen}}$, the multiplication-triples generated through $\Pi_{\text{TripGen}}$ 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 $\Pi_{\text{Beaver}}$.

The communication complexity of the protocol follows from the communication complexity of $\Pi_{\text{TripGen}}, \Pi_{\text{ACS}}$ and $\Pi_{\text{Beaver}}$. \hfill \QED