Perfectly-secure Network-agnostic MPC with Optimal Resiliency

Shravani Patil* Arpita Patra†

Abstract

We study network-agnostic secure multiparty computation with perfect security. Traditionally MPC is studied assuming the underlying network is either synchronous or asynchronous. In a network-agnostic setting, the parties are unaware of whether the underlying network is synchronous or asynchronous.

The feasibility of perfectly-secure MPC in synchronous and asynchronous networks has been settled a long ago. The landmark work of [Ben-Or, Goldwasser, and Wigderson, STOC'88] shows that $n > 3t_s$ is necessary and sufficient for any MPC protocol with n-parties over synchronous network tolerating t_s active corruptions. In yet another foundational work, [Ben-Or, Canetti, and Goldreich, STOC'93] show that the bound for asynchronous network is $n > 4t_a$, where t_a denotes the number of active corruptions. However, the same question remains unresolved for network-agnostic setting till date. In this work, we resolve this long-standing question.

> We show that perfectly-secure network-agnostic n-party MPC tolerating t_s active corruptions when the network is synchronous and t_a active corruptions when the network is asynchronous is possible if and only if $n > 2\max(t_s, t_a) + \max(2t_a, t_s)$.

When $t_a \geq t_s$, our bound reduces to $n > 4t_a$, whose tightness follows from the known feasibility results for asynchronous MPC. When $t_s > t_a$, our result gives rise to a new bound of $n > 2t_s + \max(2t_a, t_s)$. Notably, the previous network-agnostic MPC in this setting [Appan, Chandramouli, and Choudhury, PODC'22] only shows sufficiency for a loose bound of $n > 3t_s + t_a$.[1]

*Indian Institute of Science, Bangalore, India. shravanip@iisc.ac.in
†Indian Institute of Science, Bangalore, India. arpita@iisc.ac.in

[1] When $t_s > 2t_a$, our result shows tightness of $n > 3t_s$, whereas the existing work shows sufficiency for $n > 3t_s + t_a$.

Contents

1 Introduction

Secure multiparty computation (MPC) protocols enable n mutually distrusting parties to collaboratively compute a function on their inputs while ensuring the privacy of these inputs. Mutual distrust is typically modeled as an adversary that can control and coordinate the behavior of a subset of the parties. Further, depending on the resilience of MPC protocols to the prevailing network conditions, they can be classified as *synchronous* and *asynchronous*. The synchronous model has the property that the network has a known bounded delay. That is, the messages communicated between the honest parties are guaranteed to be delivered within a finite time delay, which is known publicly. In contrast, in the asynchronous network model, the messages between honest parties may be delivered after any finite delay. That is, there is no time bound to deliver the message; however, it is guaranteed that the messages between honest parties will be delivered eventually.

Traditionally, the design of MPC protocols has a monolithic view of the network. The protocols are designed assuming either a purely synchronous or purely asynchronous network; thus, the parties are aware of the network conditions. Deviating from this traditional approach of modeling the network, a line of research focuses on the scenario where parties are unaware of the network type [13, 15, 21, 3]. The requirements of both synchronous and asynchronous networks must be captured by a single protocol while ensuring security. Protocols designed in this setting are often referred to as *network-agnostic* protocols. While the prior two models had been at the center of study for more than three decades, the latter model is gaining a lot of traction recently due to its theoretical challenges and practical importance. We study network-agnostic MPC with perfect security. Perfect security, considered to be the most basic security, provides the strongest guarantee against a computationally unbounded adversary while ensuring zero error probability.

The feasibility questions for perfectly-secure MPC for synchronous and asynchronous settings have been settled a long ago. The landmark works of [27, 9] show that perfectly-secure MPC in the synchronous setting tolerating t_s active corruption is possible if and only $t_s < n/3$. Similarly, it is known that perfect security in the asynchronous setting can be achieved as long as the number of corrupt parties is $t_a < n/4$ [8, 11, 2]. The feasibility question of perfectly-secure network-agnostic MPC is still alluding. [3] shows sufficiency of such a protocol with $n > 3t_s + t_a$ tolerating t_s active corruptions when the network is synchronous and t_a active corruptions when the network is asynchronous. So far, it is not known if the bound is tight.

Our Main Result

In this work, we completely settle the feasibility of perfectly-secure network-agnostic MPC. We prove the following theorem.

Theorem 1.1 (Main Result)**.** *There exists a perfectly-secure, network-agnostic MPC protocol that is secure against an adversary corrupting up to t_s parties in a synchronous network and up to t_a parties in the asynchronous network if and only if $n > 2 \cdot \max(t_s, t_a) + \max(2t_a, t_s)$.*

Note that when $t_s \leq t_a$, our result gives a bound of $n > 4t_a$, which is the known lower bound for asynchronous MPC protocols. Also, as observed by the prior works, any known MPC protocol designed for the asynchronous network with this bound will be trivially secure in the synchronous network, thus serving as the network-agnostic protocol. For the other case, when $t_s > t_a$, we further have two cases to consider. First, when $2t_a \geq t_s$, we obtain a bound of $n > 2t_s + 2t_a$. Whereas when $2t_a < t_s$, we have that $n > 3t_s$ is necessary and sufficient. Thus, we show that the threshold

$n > 3t_s + t_a$ used in the prior works on perfectly-secure network-agnostic protocols is not tight for this setting.

Main Technical Result

Our main result is obtained via two key components– the necessity and the sufficiency.

Theorem 1.2 (Necessity). *For any n, if $2 \cdot \max(t_s, t_a) + \max(2t_a, t_s) \geq n$, then there is no perfectly-secure n-party MPC protocol that is secure against an adversary corrupting t_s parties in the synchronous network and t_a parties in the asynchronous network.*

Due to reasons discussed earlier, in the rest of the discussion, we focus our attention on the case when $t_a < t_s$. Within this case, when $2t_a \leq t_s$, the impossibility of $n \leq 3t_s$ is inherited from the impossibility of the synchronous setting. So, the most interesting case is that of $n \leq 2t_s + 2t_a$ when $2t_a > t_s$, which we prove. For simplicity, we assume $n = 2t_s + 2t_a$ and show that no network-agnostic perfectly-secure protocol with $2t_s + 2t_a$ parties can compute a specific function f (described below) securely when the network is asynchronous. For this, we first reduce the n party network-agnostic protocol to a 4 party protocol with parties P_1, P_2, P_3, P_4 such that P_1, P_2 emulate disjoint sets of t_s parties each, and each of P_3, P_4 emulate t_a parties in the underlying protocol. Next, we identify a function f as follows

$$f(x_1, x_2, \perp, \perp) \rightarrow (x_1 \wedge x_2, x_1 \wedge x_2, \perp, \perp)$$

Since the n party network-agnostic protocol is secure against an adversary corrupting t_s parties in the synchronous network and t_a parties in the asynchronous network, we have that the 4 party protocol should be secure if P_1 or P_2 is corrupt when run in a synchronous network, or one of P_3, P_4 is corrupt in an asynchronous network. We conclude our proof by demonstrating that it is impossible for the output receiving parties P_1, P_2 to have a unanimous output when the protocol is instantiated in the asynchronous network where either P_3 or P_4 is corrupt.

Our second contribution lies in providing a matching upper bound. In our view, the most technically involved contributions here are the weak secret sharing and verifiable triple sharing protocols. Weak secret sharing is a primitive with the following properties: (i) privacy: after the sharing phase, the adversary cannot learn anything about the secret of an honest dealer; (ii) commitment: the secret is completely determined by the shares of the honest parties after the sharing phase completes, however, all the honest parties may not necessarily have their shares; and (iii) correctness: if the dealer is honest, then at the end of the sharing phase, all the honest parties hold their shares corresponding to the dealer's secret. On the other hand, verifiable triple sharing allows a dealer to share multiplication triples such that all the parties can verify their correctness. In the network-agnostic setting, since the parties are unaware of the network conditions, the protocols must tolerate the worst case corruption threshold. Hence, the protocols typically operate with the sharing threshold of t_s ($> t_a$). Our construction of both the primitives, weak secret sharing as well as verifiable triple sharing crucially relies on utilizing the additional $t_s - t_a$ degree of freedom which is inherently available when the protocol operating with threshold t_s is instantiated in the asynchronous network where only at most t_a parties can be corrupt. Carefully leveraging this degree of freedom while being unaware of the exact network type constitutes our work's primary technical contribution, allowing us to obtain a protocol matching the lower bound. A verifiable secret sharing is built using the weak secret sharing to ensure that all the honest parties have shares even for a corrupt dealer. The verifiable secret sharing then serves as a building block in the triple secret sharing which acts as the primary tool for generating random multiplication triples, the main ingredient for MPC.

Theorem 1.3 (Sufficiency)**.** *Let n, t_s, t_a be such that $n > 2t_s + \max(2t_a, t_s)$. There exists a perfectly-secure, network-agnostic MPC protocol for any function secure against an adversary that can corrupt up to t_s parties in the synchronous network and up to t_a parties in the asynchronous network.*

Out weak secret sharing relies on finding a clique of size $n - t_s$ and hence requires exponential time. As discussed in Section 2, our weak secret sharing deals with a lot of challenges, despite using a clique finding algorithm. Achieving polynomial time protocols in the optimal threshold is an interesting open problem.

1.1 Related Work

We review some other related works below. Network-agnostic computation has been considered in other settings such as the general adversarial structure [4, 5], statistical security [5], computational security [13, 15, 21]. Further, it has been studied for state machine replication [14], secure message transmission [22], and consensus [3]. Importantly, [3] gives network-agnostic protocols for consensus and broadcast with perfect security and a threshold of $t_s, t_a < n/3$ which is known to be optimal for both the synchronous and asynchronous networks.

2 Technical Overview

In this section, we provide a technical overview of our work. We first describe the weak and verifiable secret sharing schemes in Section 2.1. Verifiable secret sharing protocol allows a designated party, the dealer, to perform a degree-t_s Shamir-sharing ((often, abbreviated as t_s-sharing) of its secrets. It is built on top of weak secret sharing via similar techniques used in earlier works [29, 28, 25]. In our MPC protocol, each party shares multiplication triples using the above protocol. Following this, parties must verify the correctness of these degree-t_s shared triples. This is captured by our verifiable triple sharing protocol, which requires additional techniques that are described in Section 2.2. Finally, in Section 2.3, we conclude with the high-level ideas of how these primitives are utilized to construct the network-agnostic MPC protocol.

2.1 Weak and Verifiable Secret Sharing

We start with an approach similar to the prior network-agnostic work of [3] and construct a primitive which we refer to as weak secret sharing (WSS) which proceeds in two phases, *sharing* and *reconstruction*. This primitive is weaker than verifiable secret sharing (VSS) in terms of the guarantees it offers and allows a dealer to share a secret with the following properties:

- Privacy: When the dealer is honest, the adversary cannot learn any information regarding the dealer's secret at the end of the sharing phase.
- Commitment: If the dealer is corrupt, at the end of the sharing phase, either no honest party holds its share or a subset of the honest parties hold their shares such that these shares completely determine the dealer's secret. Moreover, all the parties which hold a share are ensured to have shares corresponding to a common secret.

- Correctness: If the dealer is honest, all the honest parties hold shares consistent with the dealer's secret at the end of the sharing phase.

Although [3] provides a protocol for weak secret sharing, which they call weak polynomial sharing, they assume a threshold of $n > 3t_s + t_a$. We discuss the high level approach of [3], which follows from previous work in this setting [15] and the challenges to extend it to the optimal-resiliency setting. To construct a network-agnostic weak secret sharing protocol, the idea is to run a protocol designed for weak secret sharing in a synchronous network followed by that for an asynchronous network with some intermediate steps to ensure correctness. In more detail, the protocol design relies on observing the properties guaranteed by the synchronous protocol, and either deciding on an output or deciding to run the asynchronous protocol subsequently. Typically, the sharing occurs via a bivariate polynomial, where the dealer sends a univariate polynomial as its share to each party. This is followed by parties checking the pairwise consistency of their univariate polynomial by exchanging one point with each party and broadcasting the result of this check. Subsequently, parties ensure that the dealer has indeed committed to a polynomial (and hence a value) by checking the existence of a clique of sufficiently large size. To ensure this in polynomial time, their protocol uses the (n, t)-Star algorithm [18] whose properties are described below and in Section 3. We give a high-level relevant description of their protocol below, bypassing the finer details such as specific time steps and wait periods to ensure correctness.

1. **(Sending polynomial shares)** The dealer chooses a symmetric bivariate polynomial $S(x, y)$ with degree t_s in each variable and the constant term embedding its secret. The dealer then sends to each P_i, its share $S(x, i)$.
2. **(Pairwise consistency check)** Let the polynomial received by P_i be $q_i(x)$. Each P_i sends to every P_j a point $q_i(j)$.
3. **(Broadcasting the results of consistency check)** Let q_{ji} be received by P_i from P_j. P_i broadcasts $\mathsf{OK}(i, j)$ if $q_{ji} = q_i(j)$ holds, and $\mathsf{NOK}(i, j, q_i(j))$ otherwise.
4. **(Constructing the consistency graph)** Each party constructs a graph G with vertices as $\{1, \ldots, n\}$ such that an edge (i, j) is included in G if and only if $\mathsf{OK}(i, j)$ and $\mathsf{OK}(j, i)$ is received from the broadcast of P_i, P_j respectively.
5. **(Finding (n, t_s)-Star)** The dealer updates its consistency graph as follows:
   - Remove all the edges incident with P_i if $\mathsf{NOK}(i, j, q_{ij})$ was received from P_i such that $q_{ij} \neq S(i, j)$.
   - From the set of vertices, remove those with degree smaller than $n - t_s$. Perform this step iteratively till no more vertices can be eliminated.

   Let the graph induced after these modifications be G_D, and the set of vertices be W. Following this, the dealer runs the (n, t_s)-Star algorithm and broadcasts it if found.
6. **(Deciding on (n, t_s) or (n, t_a)-Star)** Parties run a Byzantine agreement protocol to decide on whether an (n, t_s)-Star was found or to proceed and identify an (n, t_a)-Star.
7. **(Finding (n, t_a)-Star)** If the latter is decided, then the dealer runs the (n, t_a)-Star algorithm and broadcasts it if found.
8. **(Computing the Output)** Finally, parties decide on the output based on the outcome of the byzantine agreement and upon validating the dealer's broadcast of the Star[2].

[2] Validating requires checking certain conditions. We mention the conditions relevant to our discussion when required.

Protocol in the non-optimal threshold setting [3]. As mentioned, the above protocol by Appan et al. [3] crucially relies on the fact that $n > 3t_s + t_a$. Specifically, consider the case of finding an (n, t_s)-Star in the graph G_D with vertex set as W. The output of the Star algorithm is a pair of sets say (C, D) where $C \subseteq D \subseteq W$ such that $|C| \geq n - 2t_s$ and $|D| \geq n - t_s$, and additionally, there exists an edge between each $i \in C$ and every $j \in D$. This implies $|C| \geq t_s + t_a + 1$ and $|D| \geq 2t_s + t_a + 1$. Their protocol guarantees commitment to a polynomial in the synchronous network by ensuring that all the *honest* parties in W are indeed consistent with each other. We say that parties P_i, P_j are consistent if their pairwise consistency check is successful, thus $\mathsf{OK}(i, j)$ and $\mathsf{OK}(j, i)$ are received from their broadcast respectively. Specifically, the protocol is designed with appropriate timeouts which ensures that if any pair of honest parties have a conflict (their pairwise exchanged points do not match) in the synchronous network setting, then this conflict would be conveyed to all the honest parties before they accept the (n, t_s)-Star. Parties accept the dealer's (n, t_s)-Star, that is the sets C, D, if and only if there are no conflicts among the parties in it and $C, D \subseteq W$. This ensures that if a Star is accepted, then all the honest parties included in W (and hence in C, D) are consistent with each other. Thus, there is a unique bivariate polynomial defined by the honest parties.

Now consider the scenario when the network is asynchronous; however, the adversary behaves similarly to the synchronous case till the honest parties accept (n, t_s)-Star. Now, we cannot argue that the honest parties in W are consistent with each other and define a unique bivariate polynomial based on the timeout argument. A pair of honest parties which are in conflict may be included in W solely due to the delay of their NOK messages, which can never occur in the synchronous network. Instead of the timeout guarantees, the argument for the asynchronous case relies the threshold of $n > 3t_s + t_a$. Observe that $|C| \geq n - 2t_s$, and we also have that the adversary can corrupt at most t_a parties in the asynchronous network. This ensures that there are at least $|C| - t_a > t_s$ honest parties in the set C, which are consistent with each other (by the properties of the Star algorithm). These parties thus define a unique bivariate polynomial of degree t_s in each variable. Further, this guarantees that all the parties in D also have their shares on this unique polynomial. This follows from the fact that by the properties of Star algorithm, parties in D are bound to be consistent with all the parties in C, which in turn includes at least $t_s + 1$ honest parties defining the polynomial.

Challenges with optimal-threshold. Translating the above protocol to optimal resilience has immediate problems. In particular, consider the latter case described above, where the network is asynchronous and the parties have accepted an (n, t_s)-Star. The condition $|C| - t_a > t_s$ no longer holds. This in turn implies that there is no unique bivariate polynomial defined by the shares of honest parties in C, and consequently parties in D. Thus, we do not get any guarantees from the synchronous protocol when run in the asynchronous network, which are typically required to ensure correctness. This is one of the primary hurdles in constructing our protocol and requires us to introduce new techniques.

Extending to the optimal resilience. Our first crucial observation is that the issue of ensuring the dealer's commitment can be mitigated if we consider an (n, t_a)-Star regardless of the network type. This is because, in this case, the sets C, D are such that $|C| \geq n - 2t_a$ and $|D| \geq n - t_a$. This guarantees us that $|C| - t_a = n - 3t_a > t_s$, and thus the honest parties in C indeed define a unique bivariate polynomial with their shares. However, we cannot expect an (n, t_a)-Star to be found in the synchronous network even when the dealer is honest. Given that $t_s > t_a$, even for an honest dealer, the biggest clique that the consistency graph may have is of size $n - t_s$. Whereas the Star algorithm guarantees an output of (n, t_a)-Star only when the graph contains a bigger clique of size $n - t_a$. Therefore, we start with a clique of size $n - t_s$ and find a way to expand it to a clique of size $n - t_a$ so that we end with (n, t_a)-Star regardless of the network type.

At a high level, our protocol has the following structure. It follows similar to [3] till the broadcasting of the result of consistency check among parties. Following this, the dealer identifies and broadcasts an $n - t_s$ size clique. If the dealer successfully broadcasts this within a designated time, then parties proceed to the clique extension phase. Otherwise, it is guaranteed that the dealer is either corrupt in a synchronous network, or the network is asynchronous. To handle this, parties run an agreement and immediately decide to switch modes and expect the dealer to broadcast a clique of size $n - t_a$. We now discuss the clique extension phase.

The extension combines the following observations to satisfy our requirements while maintaining privacy in each network condition. First, we observe that in the synchronous network, a pair of honest parties will broadcast the outcome of their pairwise consistency checks within a designated time. Thus, when the dealer is honest, if any pair of parties does not have an edge between them by this time, then it is assured that at least one of these parties is corrupt. Hence, we can publicly reveal the common point these parties hold without breaching privacy. On the other hand, if the network is asynchronous, this claim does not hold true. That is, a pair of parties without an edge may indeed be slow honest parties whose broadcast is delayed. Hence, such a revelation of points leads to the adversary learning more points on the polynomial. However, we observe that the protocol operating with degree (t_s, t_s) bivariate polynomial in the asynchronous network has an additional degree of freedom of $t_s - t_a$. We leverage this freedom to ensure privacy in the asynchronous network. Precisely, the dealer first identifies a clique of the maximum possible size in the consistency graph. We are done if the clique is already of size $n - t_a$. Otherwise, we expect a clique of size at least $n - t_s$. An honest dealer in synchronous network will surely find such a clique consisting of all the honest parties. To extend the clique, the dealer identifies at most $t_s - t_a$ additional parties it wishes to include in the clique as follows.

First, the dealer identifies if any party broadcast an incorrect value during pairwise consistency check or was silent in the consistency check of more than t_s parties. If it finds such parties, it includes them in a set U. Following this, it instructs all the parties to restart the protocol with the polynomials of parties in U now being public. Note that each party added to U by an honest dealer in the synchronous network is guaranteed to be corrupt and will now be forced to behave honestly, thus increasing the clique size by $|U|$. Hence, we have that once U is of size $t_s - t_a$, the dealer will succeed in identifying an $n - t_a$ sized clique consisting of all the honest parties along with the corrupt parties in U. If the dealer finds no such party that can be added to U, then it approaches the clique extension via an alternative technique. Here, the dealer arbitrarily identifies a set of $t_s - t_a - |U|$ parties, say V, outside the clique. Thereafter, it instructs all parties not yet marked consistent with V to broadcast their pairwise points, and similarly, parties in V broadcast their corresponding points. Observe that if all the parties indeed broadcast their correct points within the designated time of the synchronous network, then we have that the clique expands to size $n - t_a$. If not, then the dealer once again is able to identify the parties that are silent or broadcast an incorrect value and add them to U. Following this, similar to the earlier discussion, it instructs parties to restart the protocol. We stress that we limit the number of parties added to U to $t_s - t_a$. While in the synchronous network, the designated time steps ensure privacy for an honest dealer by only adding corrupt parties to U, privacy is maintained even in the asynchronous

Number of points received	Correct	Detect	Outcome	
			Sync	Async
$t_s + t_a + 1$	0	t_a	Success	Wait
$t_s + t_a + 2$	1	$t_a - 1$	Success	Wait
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_s + 2t_a$	$t_a - 1$	1	Success	Wait
$t_s + 2t_a + 1$	t_a	0	Success	Success
$t_s + 2t_a + 2$	t_a	1	Detect	-
$t_s + 2t_a + 3$	t_a	2	Detect	-
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2t_s + t_a + 1$	t_a	$t_s - t_a$	Detect	-

Table 1: Simultaneous error correction and detection

network due to the public revelation of at most $t_s - t_a$ polynomials of honest parties. Together with the t_a polynomials of the corrupt parties, the adversary may learn at most t_s univariate polynomial shares on the dealer's (t_s, t_s)-degree bivariate polynomial which still ensures privacy. It is worth noting that the number of reruns may go up to $t_s - t_a$.

We will briefly discuss how each party computes its share in the weak secret sharing protocol, after accepting a *fully-consistent* clique of size $n - t_a$, where all the parties in the clique are pair-wise consistent. Since the clique has at least $n - t_a - t_s > t_s + \max(t_a, t_s - t_a)$ honest parties, their shares define a unique bivariate polynomial of degree t_s in both variables. Hence, a party inside the clique can output the univariate polynomial it received from the dealer and used during pairwise consistency check. On the other hand, a party lying outside the clique is required to obtain its polynomial share which is consistent with the honest parties in the clique. For this, the parties in the clique send their pairwise common points to a party outside the clique. Again, we use some crucial observations, as below, to ensure that an honest party outside the clique indeed reconstructs a correct polynomial in all cases except when the network is synchronous and the dealer is corrupt. It is because of this exception our protocol does not qualify to be a verifiable secret sharing.

First, in an asynchronous network, online error correction and the fact that the clique is of size $n - t_a > 2t_s + \max(t_a, t_s - t_a) > 3t_a$ allows a party to reconstruct its correct polynomial by correcting at most t_a errors. On the other hand, we observe that if the network is synchronous, then all the honest parties' pairwise points get delivered to a party outside within a designated time which is known beforehand; however, we cannot ensure the correction of t_s errors. Here, we use the properties of the Reed-Solomon decoding algorithm, which allows a party to detect and correct errors simultaneously. A clever application of this technique, as discussed in the next paragraph, allows a party outside the clique to identify if the set of points it has received has more than t_a errors. This in turn allows the party to conclude if the network is synchronous leveraging the fact that $t_a < t_s$ holds. The knowledge that the network is synchronous allows a party outside the clique to conclude if the dealer behaves honestly or not, based on which it can either output the received univariate polynomial or $\perp$. We ensure that for a misbehaved corrupt dealer, it always outputs $\perp$.

We now conclude with the description of how the simultaneous error correction and detection is leveraged in our protocol. As mentioned, in a synchronous network it is guaranteed that a party receives at least $n - t_a - t_s \geq t_s + t_a + 1$ points from the honest parties in the clique. Moreover, these will be received within a designated time which is known beforehand for a synchronous network. Hence, upon receiving $t_s + t_a + 1$ points, a party starts the decoding procedure. It then decides on the number of errors to be detected and corrected as per Table 1 and decides on whether to accept the reconstructed polynomial as indicated. Suppose a party outside the clique receives $m = t_s + t_a + 1 + x$ points from the parties in the clique. Let us analyze the scenario of a synchronous network. If $x \leq t_a$, then the decoding procedure is guaranteed to succeed due to the following: (i) at most x of the total m points are erroneous, and (ii) the number of errors that can be corrected equals $\frac{m-(t_s+1)}{2} \geq x$. Hence, if the reconstruction succeeds, the party can output the reconstructed polynomial. On the other hand, if $x > t_a$, then by properties of the decoding algorithm, it can detect the presence of more than t_a errors and conclude that the network is asynchronous. Now consider the case of an asynchronous network when the party outside receives the same number of points $m = t_s + t_a + 1 + x$. Unlike the case of a synchronous network, we do not have the guarantee that at most x points are erroneous. Since the network is asynchronous and the messages are received in arbitrary or even adversarially controlled order, it is possible that there are up to t_a erroneous points. Hence, we need the mechanism of allowing for correction of up to x and additionally detection of up to $x - t_a$ errors simultaneously. In this case, if there indeed are more than x errors, then the reconstruction fails and the party can afford to receive more correct points from the slow honest parties. Note that in the worst case, when $x = t_a$, the reconstruction will definitely succeed. In our protocol, we use these observations to allow a party outside the clique to recover its polynomial. We refer the readers to Section 3 for more details about simultaneous error correction and detection and the exact bounds.

From weak secret sharing to verifiable secret sharing. We use the standard approach taken in the prior works [29, 28, 25, 3] to extend the weak secret sharing scheme to the stronger primitive of verifiable secret sharing. For this, we rely on a "two-layer" approach, wherein the first layer is similar to the weak secret sharing, whereas the second layer enables parties outside the clique to recover their polynomial even when the dealer is corrupt in the synchronous network. More specifically, in the verifiable secret sharing protocol, parties proceed very similarly to weak secret sharing, however, the pairwise consistency checks are now performed differently. Instead of directly exchanging their pairwise points, each party now instantiates an instance of weak secret sharing to share its univariate polynomial received from the dealer. A party broadcasts $\mathsf{OK}(j)$ for a party P_j in the verifiable secret sharing if and only if it computes the pairwise point as output in P_j's instance of weak secret sharing. Doing so allows a party outside the clique to reconstruct its correct polynomial based on the points from parties in whose weak secret sharing instances it computes an output. This is a standard technique to extend weak secret sharing to verifiable secret sharing, and we refer the readers to [3] and the proof of our protocol for more details of its correctness.

Challenges in achieving polynomial time protocol. We now briefly discuss the challenges we encountered while trying to achieve a polynomial time algorithm for weak secret sharing. Note that the only exponential time component in our protocol is that of clique finding of size $n - t_s$. Specifically, we allow the dealer to run in exponential time and identify a clique of size $n - t_s$. We stress that identifying such a clique is crucial to allow for its extension to size $n - t_a$. Recall that during the clique extension phase if the dealer cannot restart the protocol by adding a new party to U, then it approaches the clique extension via an alternative technique. Here, the dealer identifies a set V of (at most) $t_s - t_a$ parties outside the clique to instruct parties to resolve all their inconsistencies. Suppose for the purpose of this discussion that the dealer has identified a clique of size $n - t_s$ and the set V is of size $t_s - t_a$. If all the parties successfully broadcast consistency with parties in V and vice versa, then we are guaranteed that the parties in V are now consistent with all the parties in the clique. Hence, we now have that guarantee that the consistency graph indeed has a clique of size $(n - t_s) + (t_s - t_a) = n - t_a$. However, the same does not hold true if the parties in V are not carefully chosen from those outside of an $n - t_s$ sized clique. If the set V is chosen to be an arbitrary set of parties, then ensuring that they are consistent with all the parties does not suffice to guarantee a bigger sized clique in the consistency graph. This is because the $t_s - t_a$ parties from V may already be included in the clique. Although they are now consistent with all parties outside the clique, the other parties from the clique may still have inconsistencies with those outside, preventing clique expansion. This is the precise reason why we required identifying the exact clique in our protocol. We leave it as an interesting direction to identify if clique expansion can occur without requiring clique finding, for instance by using techniques such as Star algorithm [18].

2.2 Verifiable Triple Sharing

In a verifiable triple sharing (VTS) protocol, the dealer is required to share a multiplication triple *verifiably* while ensuring privacy of the triple. Our starting point is the verifiable triple sharing schemes of [20] which are designed independently for both the synchronous and the asynchronous networks. We outline their synchronous protocol with $t_s < n/3$ assuming a synchronous verifiable secret sharing scheme, which outputs t_s-sharing of the input secret. This is followed by the slight changes needed for their asynchronous verifiable triple sharing.

To share a multiplication triple, the dealer first chooses $2t_s + 1$ random multiplication triples (a_i, b_i, c_i) for $i \in \{1, \ldots, 2t_s + 1\}$ and shares them via degree-t_s polynomials using the verifiable secret sharing protocol. To verify the multiplicative relation, parties first transform these random triples into correlated triples (x_i, y_i, z_i) such that they lie on polynomials X, Y, Z of degree $t_s, t_s, 2t_s$ respectively such that $XY = Z$ if and only if all the input triples (a_i, b_i, c_i) for $i \in \{1, \ldots, 2t_s + 1\}$ are multiplication triples. Therefore the task of verifying the input triples reduces to the task of verifying $XY = Z$. Towards the latter, the sharings of $X(i), Y(i), Z(i)$, ith point on each of these polynomials is reconstructed to *only* P_i, who locally verifies that $X(i) \cdot Y(i) = Z(i)$ holds and broadcasts the result of its verification. If the verification fails for some party P_i, then parties publicly reconstruct $X(i), Y(i), Z(i)$ and verify the relation. If it fails, then the dealer is discarded. Otherwise, the protocol completes successfully if the (local or public) verification holds for at least $3t_s + 1$ parties, which in turn includes at least $2t_s + 1$ honest parties. The latter confirms that $XY = Z$, since the polynomials are of degree at most $2t_s$. The output of parties is the sharing of $X(\beta), Y(\beta), Z(\beta)$ for some public value $\beta \notin \{1, \ldots, n\}$. In the above protocol, the degree of the polynomials X and Y is crucially defined to t_s to ensure privacy and correctness of triple verification. Observe that the verification process reveals one point on these polynomials to every party, allowing the adversary to learn (at most) t_s points. Setting a smaller degree would allow an adversary to obtain the complete polynomials X, Y, violating the privacy of the output triple $X(\beta), Y(\beta), Z(\beta)$ for an honest dealer. On the other hand, having a higher degree would not ensure verification of a corrupt dealer's triples. Consider the scenario when $n = 3t_s + 1$. If the polynomials X, Y are of degree d such that $d > t_s$, then consequently, Z will be of degree more than $2d > 2t_s + 1$. This requires at least $2d + 1 > 2t_s + 2$ points on the polynomial to be verified by the *honest* parties, which is not possible since there may be only $2t_s + 1$ honest parties in the network in the worst case. We summarize the synchronous verifiable triple sharing scheme of [20] below.

1. The dealer shares $2t_s + 1$ multiplication triples, say (a_i, b_i, c_i) for $i \in \{1, \ldots, 2t_s + 1\}$ though a verifiable secret sharing protocol.
2. Parties transform these triples into correlated triples such that they lie on polynomials $X(\cdot), Y(\cdot), Z(\cdot)$ where $X(\cdot) \cdot Y(\cdot) = Z(\cdot)$ holds. Specifically, parties define the polynomials $X(\cdot), Y(\cdot), Z(\cdot)$ of degree $t_s, t_s, 2t_s$ respectively such that $X(i) = a_i, Y(i) = b_i$ and $Z(i) = c_i$ for each $i \in \{1, \ldots, t_s + 1\}$. Note that the degree t_s polynomials $X(\cdot)$ and $Y(\cdot)$ are completely defined by these points. Parties hold shares of each of these $t_s + 1$ points on the three polynomials.
3. Using linearity of t_s-sharing, parties hold sharing of the extrapolated points $X(i), Y(i)$ for every $i \in \{t_s + 2, \ldots, 2t_s + 1\}$.
4. Given these, they now require to compute $Z(i)$ for every $i \in \{t_s + 2, \ldots, 2t_s + 1\}$ while maintaining the multiplicative relation. For this, parties consume one multiplication triple (a_i, b_i, c_i) shared by the dealer and use Beaver's multiplication protocol to obtain the sharing of $Z(i) = X(i) \cdot Y(i)$ from the sharings of $X(i), Y(i)$ for each $i \in \{t_s + 2, \ldots, 2t_s + 1\}$. Note that the polynomial $Z(\cdot)$ of degree $2t_s$ is now defined completely.
5. Using linearity on the sharings of $\{X(i), Y(i), Z(i)\}$ for $i \in \{1, \ldots, 2t_s + 1$, parties obtain sharings of $X(i), Y(i), Z(i)$ for each $i \in \{2t_s + 2, \ldots, n\}$ though local computation. Thus parties now have sharings of each $X(i), Y(i), Z(i)$ for $i \in \{1, \ldots, n\}$.
6. To verify the multiplicative relation of the shared triples, parties have to ensure that $X(\cdot) \cdot Y(\cdot) = Z(\cdot)$ holds. Towards this, $X(i), Y(i), Z(i)$ are reconstructed to P_i, who verifies that $X(i) \cdot Y(i) = Z(i)$ holds and broadcasts the result of the verification, either OK or NOK, to all the parties. Note that this step leaks t_s points on polynomials X, Y, Z to the adversary when the dealer is honest.
7. For each party P_i whose verification fails, the check is performed publicly by reconstructing the points $X(i), Y(i), Z(i)$ to all.
8. Since the polynomials are of degree $t_s, t_s, 2t_s$ respectively, the triples are successfully verified if $2t_s + 1$ honest parties (and hence $3t_s + 1$ parties in total) confirm the relation. Otherwise, the dealer is discarded.

In the asynchronous setting with $t_a < n/4$, the protocol operates with the appropriate threshold t_a both for sharing as well as the degree of X, Y; the rest of the steps follow closely to the synchronous case with a few caveats. For instance, to avoid an endless wait in the asynchronous setting, parties can afford to wait for the OK or NOK broadcast of at most $n - t_a$ parties. However, given that $n - t_a \geq 3t_a + 1$ and the polynomials X, Y are now of degree t_a, correctness is ensured when the multiplicative relation is verified for $n - t_a$ parties. For an honest dealer, all the $n - t_a$ honest parties will eventually broadcast OK, ensuring that the triple sharing is successful. On the other hand, verifying $n - t_a$ points on the polynomial ensures correctness even for a corrupt dealer.

Network-agnostic protocol in the non-optimal threshold setting [3]. Recall that they use $n > 3t_s + t_a$. Being agnostic of the network style and the threshold, [3] follows the above protocol idea while keeping the degree of the sharings and X, Y as t_s (recall that $t_s > t_a$) and makes sure that $X(i) \cdot Y(i) = Z(i)$ holds for at least $2t_s + 1$ honest parties as follows. They define a set W of parties with $|W| \geq n - t_s$ and make sure that every party in W verifies $X(i) \cdot Y(i) = Z(i)$ either privately or publicly. The set W is constructed such that it contains all the $n - t_s$ honest parties when the network is synchronous and it contains at least $2t_s + 1$ honest parties when the network is asynchronous. For this, they wait till a designated time and add to W the first (at least) $n - t_s$ parties which respond to the verification of dealer's triples. The designated time is such that in the synchronous network, all the honest parties respond exactly at this time and hence get included in W. On the other hand, in the asynchronous network W may consist of arbitrary $n - t_s$ parties depending on the scheduling of messages of these parties. Hence, W may contain t_a corrupt parties, resulting in the presence of at least $|W| - t_a$ honest parties inside W.

As mentioned, they ensure that every party in W verifies $X(i) \cdot Y(i) = Z(i)$ either privately or publicly. This works when the network is synchronous, since every honest party is in W and there are at least $2t_s + 1$ honest parties. A corrupt dealer will get caught if it shares triples that are not multiplication triples. In contrast, when the network is asynchronous, they have at least $|W| - t_a \geq 2t_s + 1$ honest parties in W, which again ensures that either the triples are correct or the dealer is discarded.

Challenges with optimal-threshold. We observe that the protocol of [3] crucially relies on the resilience of $n > 3t_s + t_a$ to ensure correctness of triples. Specifically, reducing the threshold to optimal has an immediate problem in ensuring that the triples shared indeed satisfy the multiplicative relation. When $n > 2t_s + \max(2t_a, t_s)$, we have that $|W| = n - t_s \geq t_s + \max(2t_a, t_s) + 1$. Assume that the network is asynchronous. It no longer holds that $|W| - t_a \geq 2t_s + 1$. Hence, the correctness of the triples cannot be established. Further, expecting a bigger W, say of size $n - t_a$ to ensure the correctness may result in an indefinite wait even for an honest dealer. This is because, an adversary corrupting up to t_s parties may remain silent, preventing the protocol from proceeding.

Extending to the network-agnostic setting with optimal resilience. We now discuss our techniques, which extend the ideas of the above approach to the network-agnostic setting. To account for the worst-case corruption threshold, our protocol too operates with t_s-sharing and polynomials X, Y of degree t_s. Observe that, following a similar template as above, to ensure the correctness of the multiplicative relation, $2t_s + 1$ *honest* parties are required to confirm their local verification, had the network been synchronous. On the other hand, in the asynchronous setting, it suffices for *any* $t_a + (2t_s + 1)$ parties to confirm.

We ensure these two conditions hold in our network agnostic protocol as follows. First we enforce that parties resolve the NOK received from any party within a pre-specified time before computing their output in the protocol. Second, we demand that the total number of distinct points i for which $X(i) \cdot Y(i) = Z(i)$ is verified, either privately or publicly, be at least $n - t_a$. Contrast this with the $n - t_s$ number of points required to be verified in [3]. The first requirement ensures correctness in the synchronous network, whereas the second condition guarantees it in the asynchronous network. Specifically, in a synchronous network, the properties offered by the network-agnostic broadcast protocol make sure that all the honest parties receive the OK or NOK messages from other honest parties within a designated time. Hence, they compute their output only upon verifying each NOK message received. This ensures that if the dealer is not discarded, then $X(i) \cdot Y(i) = Z(i)$ has been verified for all the honest parties. Since there are at least $2t_s + 1$ honest parties in the synchronous case, we are guaranteed correctness of the triples. On the other hand, if the network is asynchronous, the second condition of verifying a total of $n - t_a$ points comes into effect to ensure correctness. Since parties verify the multiplicative relation for $n - t_a$ points and the adversary can corrupt at most t_a parties, we have that the relation holds for at least $n - 2t_a \geq 2t_s + 1$ honest parties. Again, we are guaranteed correctness of the multiplication triples. However, making sure these two conditions hold requires additional techniques.

Observe that in a synchronous network, we can expect at most $n - t_s$ parties to broadcast the result of their local verification within the designated time. There may be t_s corrupt parties which remain silent, that is, these parties neither broadcast OK nor NOK. In such a case, enforcing a support of $n - t_a$ would result in stalling the protocol even for an honest dealer. To remedy this, we perform a *dealer-guided* public reconstruction of points $X(i), Y(i), Z(i)$ of a subset of parties who either broadcast NOK later than the designated time or are silent. We enforce that the total number of points verified, which includes the publicly verified values and those from the OK messages of parties is at least $n - t_a$. Here, the term *dealer-guided* refers to the criteria that the dealer chooses the parties whose points have to be reconstructed publicly. An important aspect to note here is that with t_s-degree polynomials we do not have any additional degree of freedom in the synchronous setting. Moreover, we have only $t_s - t_a$ degree of freedom in the asynchronous setting. Thus, revealing the points has to be performed carefully by the dealer. To maintain privacy here, we ensure that the dealer only begins the guided reconstruction upon waiting for a designated time and additionally receiving *at least* $n - t_s$ OKs, and performs the public reconstruction for *at most* $t_s - t_a$ parties. The intuition behind privacy in the synchronous setting is that honest parties always broadcast their OK messages which are received by all within in the designated time. Hence, their points are never reconstructed publicly. On the other hand, in the asynchronous setting, some of the honest parties may be slow. However, an honest dealer reveals points for at most $t_s - t_a$ honest parties, still ensuring the degree of freedom of 1 and hence maintaining privacy.

In conclusion, our protocol ensures that parties verify all the NOK received within a designated time and that the verification succeeds for at least $n - t_a \geq t_a + 2t_s + 1$ parties in total. This ensures that if the dealer is not discarded, the triples generated are correct regardless of the underlying network type.

2.3 Putting it all together: The MPC Protocol

We now describe the construction of our MPC protocol using the above primitives. At a very high level, the MPC protocol uses Beaver's circuit randomization trick [7] and adopts a two-phase structure. The first phase corresponds to *Beaver triple generation*, followed by the second phase of *circuit evaluation*. The verifiable secret sharing and verifiable triple sharing primitives are utilized in the former phase, whereas existing primitives suffice for the latter.

Beaver triple generation. In more detail, the Beaver triple generation phase ensures that verified random multiplication triples are shared among parties as follows. Each party enacts the role of the dealer and shares random multiplication triples using verifiable secret sharing. Upon completing the sharing, parties must verify the correctness of these triples, which is precisely where they rely on the verifiable triple sharing. If the triples shared by a dealer are verified to be correct, then they are accepted in the further computation. Otherwise, parties discard the dealer and assume a default sharing on behalf of the dealer. Note that the corrupt parties may never initiate an instance of triple sharing. Moreover, given that the network may be asynchronous, waiting for all n dealers' instances may result in an endless wait. To prevent this, we use an instance of a primitive called asynchronous common set (ACS), which allows parties to agree on a common set of at least $n - t_s$ dealers whose shared triples will be used in circuit evaluation. Agreement on this set amongst parties is also crucial since it is likely that different parties compute their output in the triple sharing instances of dealers in a different order due to asynchrony. At this stage, we have that parties have agreed on the set of triples shared by at least $n - t_s$ dealers. To ensure secure evaluation of the circuit, we however require random multiplication triples that are unknown to any party. For this, we use a 'triple extraction' protocol from the literature [20, 3] which consumes one triple shared by each dealer and extracts a random triple unknown to any party. This completes the Beaver triple generation phase, with parties holding shares corresponding to random multiplication triples.

Circuit evaluation. Parties use the triples from the prior phase to perform a shared evaluation of the circuit. Parties begin by sharing their inputs to the circuit. Similar to the case of triple sharing, waiting for the input of all the parties may result in an endless wait. Hence, parties run an instance of ACS to agree on a set of at least $n - t_s$ parties whose input will be considered for evaluation. A default value is assumed as the input of the remaining parties. We remark that in our protocol, parties can simultaneously perform triple sharing and input sharing, followed by a single instance of the ACS protocol. Subsequently, the evaluation of the circuit proceeds as follows. The evaluation of linear gates (addition and multiplication by a constant) is done locally. Parties use one multiplication triple from the first phase and rely on Beaver's multiplication [7] protocol to evaluate a multiplication gate. Following this, parties reconstruct the protocol output. Finally, they terminate upon ensuring that a sufficient number of parties have computed the output so as to ensure that all parties obtain their output. This concludes our MPC protocol.

3 Preliminaries

3.1 Network Model and Definitions

We consider a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ connected via pairwise private and authenticated channels. The distrust among the parties is modeled as a centralized, *computationally unbounded* adversary. We consider a static adversary that decides the set of corrupt parties at the beginning of the protocol execution. The underlying network conditions can be synchronous or asynchronous, and the parties are unaware of the exact network type during the protocol execution. In a synchronous network, every message sent is delivered within a fixed, known time bound Δ. Moreover, the messages are delivered in the same order they are sent in. In contrast, in the asynchronous network, the messages are delivered with an arbitrary but finite delay with the only guarantee that the messages are eventually delivered. Moreover, the messages may be delivered in an arbitrary order. This is modeled by a scheduler which decides on the sequence of message deliveries, where the scheduler is assumed to be controlled by the adversary. The adversary can corrupt up to t_s out of the n parties maliciously when the network is synchronous, whereas it can corrupt up to t_a parties under asynchronous network conditions and make them behave arbitrarily.

Our protocols are defined over a field $\mathbb{F}$, such that $|\mathbb{F}| > n$. We denote the elements of the field by $\{0, 1, \ldots, n\}$. Further, we use $[v]$ to denote the degree-t_s Shamir-sharing of a value v among parties in $\mathcal{P}$.

Additionally, in constructing our protocols, we use several well-known primitives from the literature. We elaborate on these in Section 4 and refer the readers to the same for further details.

3.2 Symmetric Bivariate Polynomials

A degree (l, l) symmetric bivariate polynomial over $\mathbb{F}$ is of the form $F(x, y) = \sum_{i,j=0}^{i=l,j=l} b_{ij} x^i y^j$ where $b_{ij} \in \mathbb{F}$ and $b_{ij} = b_{ji}$ holds for all $i, j \in \{0, \ldots, l\}$. This implies that $F(i, j) = F(j, i)$ holds for every i, j. Moreover, $F(x, i) = F(i, y)$ is also true for each $i \in \{1, \ldots, n\}$.

Our protocol uses (t_s, t_s) symmetric bivariate polynomials. Further, $f_i(x) = F(x, i) = F(i, y)$ is called the ith univariate polynomial of $F(x, y)$ and is associated with party P_i in the protocol.

3.3 Finding a (n, t)-Star

Definition 3.1. *Let G be a graph over the nodes $\{1, \ldots, n\}$. We say that a pair (C, D) of sets such that $C \subseteq D \subseteq \{1, \ldots, n\}$ is an (n, t)-star in G if the following hold: (a) $|C| \geq n - 2t$, (b) $|D| \geq n - t$, (c) For every $j \in C$ and every $k \in D$, the edge (j, k) exists in G.*

3.4 Almost-surely Terminating

Following the approach of Appan et al. [3], we use randomized asynchronous byzantine agreement protocols designed for threshold $t_a < t_s < n/3$ (note that our resiliency matches with this requirement) in our work, which guarantee that *almost-surely* all the honest parties eventually receive their output. This implies that the probability that an honest party receives its output after participating in an infinite number of rounds of a protocol approaches 1 asymptotically [1, 24, 6]. Specifically,

$$\lim_{T \to \infty} \Pr \left[\text{An honest party } P_i \text{ receives its output by local time } T \right] = 1$$

where the probability is over the randomness of the honest parties and the adversary in the protocol. Also, the property of almost-surely receiving the output carries forward to all the protocols that use asynchronous byzantine agreement as a primitive. Similar to [3], for simplicity, we do not specify the terminating condition for each sub-protocol. Rather, when a party terminates the MPC protocol, it also terminates in all the sub-protocol instances.

3.5 Simultaneous Error Correction and Detection of Reed-Solomon Codes

We require the following coding-theory related results. Let C be a Reed-Solomon (RS) code word of length N, corresponding to a k-degree polynomial (containing $k+1$ coefficients). Assume that at most t errors can occur in C. Let $\bar{C}$ be the word after introducing error in C in at most t positions. Let the distance between C and $\bar{C}$ be s where $s \leq t$. Then there exists an *efficient* decoding algorithm that takes $\bar{C}$ and a pair of parameters (e, e') as input, such that $e + e' \leq t$ and $N - k - 1 \geq 2e + e'$ hold and gives one of the following as output:

1. Correction: output C if $s \leq e$, i.e. the distance between C and $\bar{C}$ is at most e;
2. Detection: output "more than e errors" otherwise.

Note that detection does not return the error indices; rather, it simply indicates error correction fails due to the presence of more than correctable (i.e., e) errors. The above property of RS codes is traditionally referred to as *simultaneous error correction and detection.* In fact, the bounds, $e + e' \leq t$ and $N - k - 1 \geq 2e + e'$, are known to be necessary. We cite:

Theorem 3.2 ([19, 23])**.** *Let C be a Reed-Solomon (RS) code word of length N, corresponding to a k-degree polynomial (containing $k+1$ coefficients). Let $\bar{C}$ be a word of length N such that the distance between C and $\bar{C}$ is at most t. Then RS decoding can correct up to e errors in $\bar{C}$ to reconstruct C and detect the presence of up to $e + e'$ errors in $\bar{C}$ if and only if $N - k - 1 \geq 2e + e'$ and $e + e' \leq t$.*

Corollary 3.3. *Let C and $\bar{C}$ be as in Theorem 3.2 with $N = t_s + t_a + 1 + x$, $k = t = t_s$ and $x \leq t_a$. Then RS decoding can correct up to x errors and detect the presence of up to $t_a - x$ errors in $\bar{C}$.*

Proof. This follows since $N - k - 1 = t_a + x$, $2e + e' = 2x + (t_a - x) = t_a + x$ and $e + e' = t_a < t_s$ hold. □

Corollary 3.4. *Let C and $\bar{C}$ be as in Theorem 3.2 with $N = t_s + t_a + 1 + x$, $k = t = t_s$ and $t_a < x \leq t_s$. Then RS decoding can correct up to t_a errors and detect the presence of up to $x - t_a$ errors in $\bar{C}$.*

Proof. This follows since $N - k - 1 = t_a + x$, $2e + e' = 2t_a + (x - t_a) = t_a + x$ and $e + e' = x \leq t_s$ hold. □

4 Existing Primitives

In our work, we use network-agnostic protocols from [3] for several primitives, such as broadcast and byzantine agreement, to name a few. Although designed for the non-optimal threshold, these naturally follow to the optimal threshold scenario. Below, we give a description of each of them along with the (n, t)-Star algorithm from [18] for completeness.

4.1 Finding a (n, t_a)-Star

Definition 4.1. *Let G be a graph over the nodes $\{1, \ldots, n\}$. We say that a pair (C, D) of sets such that $C \subseteq D \subseteq \{1, \ldots, n\}$ is an (n, t)-star in G if the following hold:*

- *$|C| \geq n - 2t$,*
- *$|D| \geq n - t$,*
- *For every $j \in C$ and every $k \in D$, the edge (j, k) exists in G.*

Canetti [17] showed that if a graph has a clique of size $n - t$, then there exists an efficient algorithm which always finds an (n, t)-Star. In our protocol, we use the parameter $t = t_a$. For completeness, we describe the algorithm for finding an (n, t_a)-Star in Algorithm 4.2, which is taken from [8, 18] and modified to suit our parameter. Moreover, we modify the algorithm from [18] to output the extended Star using the techniques of [26].

Protocol 4.2: (n, t_a)-Star

Input: An undirected graph G (over the nodes $\{1, \ldots, n\}$) and a parameter t_a.

1. Find a maximum matching M in $\overline{G}$. Let N be the set of matched nodes (namely, the endpoints of the edges in M) and let $\overline{N} := \{1, \ldots, n\} \setminus N$.

2. Let T be the set of triangle-heads, i.e., all vertices that are not endpoints of the matching but they have two neighbors in the matching.

$$T \quad := \quad \left\{ i \in \overline{N} \mid \exists j,k \text{ s.t. } (j,k) \in M \text{ and } (i,j),(i,k) \in \overline{G} \right\} \ .$$

   Let $C := \overline{N} \setminus T$.
3. Let B the set of matched nodes that have neighbors in C. That is, set:

$$B \quad := \quad \left\{ j \in N \mid \exists i \in C \text{ s.t. } (i,j) \in \overline{G} \right\} \ .$$

   Let $D := \{1,\ldots,n\} \setminus B$.
4. If $|C| \geq n - 2t_a$ (i.e. $|C| \geq 2t_s + 1$) and $D \geq n - t_a$ (i.e. $|D| \geq 2t_s + t_a + 1$) then compute E as the set of all the parties which do not have edges with at least $2t_s + 1$ parties in C. Finally, construct a set F as the set of all the parties that do not have edges with at least $2t_s + 1$ parties in E.
5. **Output:** If $|E| \geq n - t_a$ and $|F| \geq n - t_a$ then output (C, D, E, F). Otherwise, output $\perp$.

4.2 Asynchronous Reliable Broadcast (Acast)

As in [3], we use Bracha's asynchronous reliable broadcast protocol [16] (also referred to as Acast) where there is a designated sender who holds a message $m \in \{0,1\}^\ell$ to be communicated to all the parties. Appan et al. [3] demonstrated that the Acast protocol, although designed for the asynchronous network, also provides certain guarantees in the synchronous network. We recall the protocol and its properties below.

Protocol 4.3: Π_{Acast}

Input: The sender holds a message $m \in \{0,1\}^\ell$.

1. The sender on holding an input m, sends $(\texttt{init}, m)$ to all the parties.
2. Upon receiving $(\texttt{init}, m)$ from the sender, send $(\texttt{echo}, m)$ to all the parties. Do not execute this step more than once.
3. Upon receiving $(\texttt{echo}, m')$ from $n - t$ parties, send $(\texttt{ready}, m')$ to all the parties.
4. Upon receiving $(\texttt{ready}, m')$ from $t + 1$ parties, send $(\texttt{ready}, m')$ to all the parties.
5. Upon receiving $(\texttt{ready}, m')$ from $n - t$ parties, output m'.

Lemma 4.4. *Bracha's Acast protocol Π_{Acast} is secure against an adversary corrupting up to $t < n/3$ parties and achieves the following properties.*

1. *Synchronous Network:*
   (a) *Liveness: If the sender is honest, then all the honest parties obtain an output within time 3Δ.*
   (b) *Validity: If the sender is honest, then every honest party with an output, has the sender's message m as the output.*

   (c) *Consistency: If the sender is corrupt and some honest party outputs m' at time T, then every honest party outputs m' within time $T + 2\Delta$.*

2. *Asynchronous Network:*
   (a) *Liveness: If the sender is honest, then all honest parties eventually obtain an output.*
   (b) *Validity: If the sender is honest, then every honest party with an output, has the sender's message m as the output.*
   (c) *Consistency: If the sender is corrupt and some honest party outputs m', then every honest party eventually outputs m'.*

4.3 Byzantine Broadcast (BC)

Appan et al. [3] construct a broadcast protocol which relies on Bracha's asynchronous reliable broadcast [16] and an existing synchronous byzantine agreement protocol which is denoted by Π_{SBA}. We give the protocol Π_{BC} for broadcast below, assuming the existence of Π_{Acast} and Π_{SBA}. We avoid repetition and refer the readers to [3] for further details on the exact instantiation of these protocols since we make a black-box use of these primitives.

Protocol 4.5: Π_{BC}

Input: The sender holds a message $m \in \{0,1\}^{\ell}$.
(Regular Mode):

1. The sender Acasts the message m using Π_{Acast}.
2. At time 3Δ, each party P_i participates in an instance of synchronous broadcast protocol Π_{SBA} with its input set as follows:
   - If $m' \in \{0,1\}^{\ell}$ is received from the Acast of the sender, then P_i sets m' as the input.
   - Otherwise, P_i sets its input as $\perp$.
3. At time $3\Delta + T_{\mathsf{SBA}}$, each P_i computes its output as follows:
   - If $m' \in \{0,1\}^{\ell}$ is received from the Acast of the sender and m' is computed as the output of Π_{SBA}, then P_i sets m' as the output.
   - Otherwise, P_i sets its output as $\perp$.

(Fallback Mode):

1. Each P_i which has computed its output as $\perp$ at time $3\Delta + T_{\mathsf{SBA}}$, updates it to m' if m' is received from the Acast of the sender.

Lemma 4.6. *Protocol Π_{BC} is secure against an adversary corrupting up to $t < n/3$ parties and has the following properties, where $T_{\mathsf{BC}} = 3\Delta + T_{\mathsf{SBA}} = (12n-3)\Delta$ when Π_{SBA} is instantiated using [12].*

1. *Synchronous network:*
   (a) *(Regular Mode)*
      i. *Liveness: At time T_{BC}, every honest party has an output (through regular-mode).*
      ii. *Validity: If the sender is honest, then every honest party outputs m (through regular-mode).*

iii. *Consistency: If the sender is corrupt, then every honest party has the same output (m' or $\perp$) at the end of T_{BC} (through regular-mode).*

(b) *(Fallback Mode)*

i. *Fallback Consistency: If the sender is corrupt and some honest party outputs m' at time $T > T_{\mathsf{BC}}$ (through fallback-mode), then every honest party outputs m' by time $T + 2\Delta$ (through fallback-mode).*

2. *Asynchronous network:*

(a) *(Regular Mode)*

i. *Liveness: At time T_{BC}, every honest party has an output (through regular-mode).*

ii. *Weak Validity: If the sender is honest, then every honest party outputs m or $\perp$ (through regular-mode).*

iii. *Weak Consistency: If the sender is corrupt, then every honest party has either a common m' or $\perp$ as the output at the end of T_{BC} (through regular-mode).*

(b) *(Fallback Mode)*

i. *Fallback Validity: If the sender is honest, then each honest party that outputs $\perp$ at T_{BC} (through regular-mode) outputs m (through fallback-mode).*

ii. *Fallback Consistency: If the sender is corrupt and some honest party outputs m' at time T (either through regular or fallback-mode), then every honest party eventually outputs m' (either through regular or fallback-mode).*

4.4 Byzantine Agreement (BA)

Appan et al. [3] provide a network-agnostic byzantine agreement protocol by following the approach of [13]. Here, each party first broadcasts its input via an instance of Π_{BC} followed by running an instance of some asynchronous byzantine agreement protocol Π_{ABA}. Each party decides its input to Π_{ABA} based on the number of parties for which it received an output in their respective instance of the broadcast protocol and the plurality of the received values. We provide the protocol from [3] below for completeness, where Π_{ABA} can be instantiated with any existing protocol such as [1, 6].

Protocol 4.7: Π_{BA}

Input: Each P_i holds a bit $b_i \in \{0, 1\}$. Each P_i also initialises a set $R_i \leftarrow \phi$.

1. Each P_i on holding an input b_i, broadcasts b_i using Π_{BC}.
2. For $j \in \{1, \ldots n\}$, let $b_i^{(j)} \in \{0, 1, \perp\}$ be received from the broadcast of P_j via regular mode. Update $R_i = R_i \cup \{j\}$ if $b_i^{(j)} \neq \perp$. Compute the input v_i for an instance of Π_{ABA} as follows:
   - If $|R_i| \geq n - t$ then set v_i to be the majority bit among the $b_i^{(j)}$ values of parties in R_i. If there is no majority, then set $v_i = 1$.
   - Otherwise, set $v_i = b_i$.
3. At time T_{BC}, participate in an instance of Π_{ABA} with input v_i. Set the output as the output computed from Π_{ABA}.

Lemma 4.8. *Protocol Π_{BA} achieves the following properties in the presence of an adversary which corrupts up to $t < n/3$ parties:*

1. *Synchronous network: The protocol is a perfectly-secure SBA protocol, where all the honest parties receive their output within time $T_{\mathsf{BA}} = T_{\mathsf{BC}} + T_{\mathsf{ABA}}$.*
   (a) *Guaranteed liveness: All the honest parties obtain an output by time T_{BA}.*
   (b) *Validity: If all the honest parties have the same input v, then all the honest parties with an output, outputs v.*
   (c) *Consistency: All the honest parties with an output, output the same value v.*
2. *Asynchronous network: The protocol is a perfectly-secure ABA protocol.*
   (a) *Almost-surely liveness: Almost-surely, all the honest parties obtain an output eventually.*
   (b) *Validity: If all the honest parties have the same input v, then all the honest parties with an output, outputs v.*
   (c) *Consistency: All the honest parties with an output, output the same value v.*

4.5 Agreement on a Common Set (ACS)

The ACS primitive [18] allows parties to agree on a common set of at least $n - t$ parties $\mathsf{Com} \subset \mathcal{P}$, such that each party in Com satisfies some predefined property prop which has the following features in the asynchronous network:

1. Every honest party eventually satisfies prop.
2. If some honest P_i sees that a party P_j satisfies prop, then eventually all the honest parties see that P_j satisfies prop.

Although the above protocol was primarily designed for the asynchronous network, it was shown in [3] that the protocol satisfies certain properties in the synchronous network where each party in Com satisfies some predefined property prop which has the following features:

1. Every honest party satisfies prop at the onset of the protocol.
2. If some honest P_i sees that a party P_j satisfies prop, then within a fixed time, all the honest parties see that P_j satisfies prop.

In our protocols, we use the parameter $t = t_s$. We describe the variant of the protocol from [18], which was used in [3] for completeness.

Protocol 4.9: Π_{ACS}

Input: Each party P_i holds a dynamically growing set S_i.
Input Guarantees:

- If the network is synchronous, then for an honest P_i, at the onset $j \in S_i$ for each honest P_j. Moreover, if a corrupt $k \in S_i$ for some honest P_i, then within a fixed time, $k \in S_j$ for all honest parties P_j.
- If the network is asynchronous, then for an honest P_i, eventually $j \in S_i$ for each honest P_j. Moreover, if $k \in S_i$ for some honest P_i, then eventually $k \in S_j$ for all honest parties P_j.

1. Each P_i participates in an instance of byzantine agreement protocol Π^j_{BA} where $j \in \{1, \ldots, n\}$ with input 1 if $j \in S_i$.
2. Once (at least) $n - t_s$ instances of Π_{BA} terminate with output 1, P_i participates with input 0 in the byzantine agreement instances Π^j_{BA} such that $j \notin S_i$.

3. Upon termination of all the n instances of byzantine agreement, P_i outputs Com as the set of parties P_j such that Π_{BA}^{j} terminated with the output 1.

Theorem 4.10. *Protocol Π_{ACS} is secure against an adversary corrupting up to t_s parties in the synchronous network and t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *Liveness: At time $T_{\mathsf{ACS}} = 2T_{\mathsf{BA}}$, every honest party has an output.*
   (b) *t_s correctness: At time T_{ACS}, every honest party outputs Com of size at least $n - t_s$ such that the following holds:*
      - *All the honest parties belong to Com.*
      - *For each $j \in \mathsf{Com}$, it is guaranteed that $j \in S_i$ for each honest party P_i.*
2. *Asynchronous network:*
   (a) *Liveness: Almost-surely, every honest party eventually has an output.*
   (b) *t_a correctness: Almost-surely, every honest party eventually outputs Com of size at least $n - t_s$ such that the following holds:*
      - *For each $j \in \mathsf{Com}$, it is guaranteed that eventually $j \in S_i$ for each honest party P_i.*

5 Lower Bound

Theorem 5.1. *For any n, if $2 \cdot \max(t_s, t_a) + \max(2t_a, t_s) \geq n$, then there is no n-party MPC protocol that is perfectly-secure against an adversary corrupting t_s parties in the synchronous network and t_a parties in the asynchronous network.*

Proof. We first consider two cases, when $t_s \leq t_a$ and otherwise. For the former case, we have that $4t_a \geq n$, and the known impossibility result of [8] follows immediately. For the latter scenario, when $t_s > t_a$, we further analyze it considering two cases. First, when $2t_a < t_s$, we have that $3t_s \geq n$. We now see that the impossibility of a network-agnostic protocol for this setting follows directly from the impossibility of synchronous protocols with this threshold [10]. Thus, what remains to be shown is the case of $2t_s + 2t_a \geq n$ when $t_s > t_a$ and $2t_a \geq t_s$. We prove this by contradiction as follows.

Assume $2t_s + 2t_a = n$, and there exists a generic MPC protocol π which is t_s-secure in the synchronous network and t_a-secure in the asynchronous network. Partition the n parties into four sets S_1, S_2, S_3, S_4 such that $|S_1| = |S_2| = t_s$ and $|S_3| = |S_4| = t_a$ and consider the following scenarios.

Case I: Synchronous network, Parties in S_1 (S_2) are corrupted. The adversary blocks all communication from parties in S_1 (S_2) towards parties in S_2 (S_1). Further, it ignores the messages received from the parties in S_2 (S_1) during its local computation. It performs the rest of the computation and communication as per the protocol specification.

Case II: Asynchronous network, Parties in S_4 are corrupted. In this case, the adversary indefinitely delays all the communication between the (honest) parties in S_1 and S_2. The adversary performs the computation and communication with the parties as per the protocol specification.

Observe that the corruption scenarios described above are valid in the synchronous and asynchronous networks, respectively. Moreover, each party's view is identical in both scenarios, thus guaranteeing that the parties remain unaware of the network type when either of the aforementioned corruption occurs during the protocol. The security guarantees of the protocol ensure that, in either case, parties receive the output of the protocol. We leverage these observations to arrive at a contradiction.

Specifically, we show that given such an n-party generic MPC protocol, we can construct an MPC protocol for 4 parties, say $P_1, \dots, P_4$ where P_i emulates the parties in S_i. This new protocol is secure with respect to an adversary that either corrupts one of P_1, P_2 when the network is synchronous or corrupts one among P_3, P_4 when the network is asynchronous. Now consider an instance of the protocol amongst the four parties to compute the following functionality:

$$f(x_1, x_2, \perp, \perp) \rightarrow (x_1 \wedge x_2, x_1 \wedge x_2, \perp, \perp)$$

We show that it is impossible for the output receiving parties, P_1 and P_2 to have a unanimous output, thus showing the impossibility of the underlying n party network-agnostic protocol. Consider the scenario when the network is asynchronous, and the adversary corrupts the party P_4. Further, the adversary follows the same (valid) strategy of blocking communication between parties as described in **Case II**, which implies blocking communication between P_1 and P_2 in the 4-party protocol.

Let $\pi(x_1, x_2)$ be an instance of the protocol with inputs x_1, x_2 and $r_i^{\pi(x_1,x_2)}$ for each $i \in [4]$ denote the randomness of each P_i in the instance $\pi(x_1, x_2)$. Let T_{ij} $(1 \leq i < j \leq 4)$ denote the transcript of the channels between P_i and P_j. Note that $T_{12} = \phi$. Moreover, due to perfect security, T_{13} and T_{14} individually are independent of P_1's input x_1. Otherwise, a corrupt P_3 or P_4 will be able to learn P_1's input. For the same reason, T_{23} and T_{24} individually are independent of P_2's input x_2. Hence, we can conclude that P_1's output is determined by its internal state and the joint distribution $\{T_{13}, T_{14}\}$. Similarly, P_2's output is determined by its internal state and the joint distribution $\{T_{23}, T_{24}\}$. Suppose these are the transcripts of the protocol instance $\pi(0, 1)$. Since T_{23} and T_{24} are individually independent of x_1, there exists some T'_{24} such that $\{T_{23}, T'_{24}\}$ results in an output 1 for P_2. If not, then it implies that irrespective of T_{24}, the output of P_2 is always 0. This further implies that the output of P_2 is completely decided by its internal state and T_{23}. However, T_{23} itself is independent of x_1. This is because, the view of P_3 in the protocol must be independent of x_1, due to perfect security and T_{23} is contained in the view of P_3. Now note that P_1 does not communicate with P_2 at all. This means P_1's input is ignored in the output computation of P_2, leading to breach of correctness. Therefore, we can conclude that in an instance $\pi(0, 1)$, there exists some T'_{24} such that $\{T_{23}, T'_{24}\}$ results in an output 1 for P_2.

Relying on the above fact, we can now conclude that an adversary corrupting P_4 in an instance of $\pi(0, 1)$ can behave according to T_{14}, T_{34} with P_1, P_3 respectively, and according to T'_{24} with P_2. This results in P_1 having the output 0, while P_2 outputs 1.

□

6 Weak Secret Sharing

Some part of the protocol proceeds in a sequence of time steps. Whereas some parts are action-based, parties execute these steps as and when they receive the messages required to perform these steps. Throughout the descriptions of protocols, we denote the wait period of time steps with red font, whereas the action-based steps of our protocols are denoted using blue font. We use also use the existing primitives such as broadcast and agreement, which are emulated using Protocol 4.5 and

Protocol 4.7 of [3] (recalled in Appendix 4.3 and 4.4 respectively). In the subsequent description, Δ denotes the round delay associated with a synchronous network. We also use the notations T_{BC} and T_{BA} to denote the time required by the broadcast and agreement protocols of [3] in the synchronous network. The exact values for these are inherited from their work and detailed in Section 4.

Protocol 6.1: Π_{WSS}

Input: The dealer holds a secret $s \in \mathbb{F}$.
Initialisation: The dealer initialises two sets W, U to ϕ. Only W is reset in every run to $\emptyset$.

1. (Polynomial Share Distribution) The dealer chooses a symmetric bivariate polynomial $F(x, y)$ of degree t_s in both x, y and delivers $f_i(x) = F(x, i)$ to P_i. If $|U| > t_s - t_a$, then assign U to be the set of first $t_s - t_a$ parties lexicographically. The dealer broadcasts $(U, \{f_i(x)\}_{i \in U})$.
2. (Pair-wise exchange) At time Δ, if $f_i(x)$ is received then every P_i sends $f_{ij} = f_i(j)$ to every P_j.
3. (Pair-wise Consistency Check) At time T_{BC}[3] P_i prepares a vector R_i of length n as follows and broadcasts it. It sets $R_i[j] = \mathsf{NR}$ for all j if any of the following happens:
   (a) it receives no $f_i(x)$
   (b) the dealer's broadcast results in $\perp$
   (c) some $f_j(x)$ in the broadcast $(U, \{f_i(x)\}_{i \in U})$ is of degree more than t_s
   (d) there are indices j, k such that $f_j(k) \neq f_k(j)$ in the broadcast $(U, \{f_i(x)\}_{i \in U})$

   Otherwise, it sets R_i as follows. (1) if $P_j \in U$, then $R_i[j] = f_i(j)$ (2) if $P_j \notin U$, then set (a) $R_i[j] = \mathsf{NR}$ if no f_{ji} is received from P_j, (b) $R_i[j] = f_i(j)$ if f_{ji} is received from P_j and $f_i(j) \neq f_{ji}$, (c) $R_i[j] = \mathsf{OK}$ otherwise.
4. (Asynchronous Pair-wise Consistency Checking) The parties execute the following steps as and when they receive the required values. On receiving the broadcast $(U, \{f_i(x)\}_{i \in U})$ and polynomial $f_i(x)$ from the dealer, every $P_i \notin U$ broadcasts AOK_j if (a) f_{ji} from $P_j \notin U$ is received and $f_i(j) = f_{ji}$ (b) $f_j(i)$ for $P_j \in U$ satisfies $f_i(j) = f_j(i)$.
5. (Restart or Clique Finding) At time $2T_{\mathsf{BC}}$, the dealer puts $P_i \notin U$ in W if either happens (a) P_i's broadcast of R_i resulted in $\perp$ or (b) P_i's broadcasted R_i has more than t_s NRs or (c) $R_i[j] \neq F(i, j)$ when $R_i[j] \neq \mathsf{OK}$ and $R_i[j] \neq \mathsf{NR}$.
   The dealer makes a graph G with n vertices corresponding to n parties. There is an edge when $R_i[j] = R_j[i] = \mathsf{OK}$. There is no edge if $R_i[j] = \mathsf{NR}$ or $R_j[i] = \mathsf{NR}$. The dealer finds a clique Q of size $n - t_s + |U|$ in the graph including U. If $|Q| \geq n - t_a$, then the dealer sets $Q_a = Q$ and broadcasts (sync, G, Q_a). Otherwise, if $|W| > 0$, then the dealer sets $U = U \cup W$ and broadcasts $(\mathsf{restart}, U)$. Otherwise, it broadcasts $(\mathsf{continue}, Q, G, V)$, where V is a set of $(t_s - t_a) - |U|$ parties (vertices) outside $Q \cup U$.
6. (Asynchronous Clique Finding) The dealer executes the following steps as and when it receives the required messages. First, the dealer initiates a graph A with parties as vertices with edges between a pair of parties in U. On receiving broadcasts AOK_{ij} and AOK_{ji} from $P_i, P_j \notin U$, it adds an edge between P_i, P_j. On receiving broadcast AOK_{ij} from $P_i \notin U, P_j \in U$, it adds an edge between P_i, P_j. Each time there is an update in A, it invokes $(C, D, E, F) \leftarrow \mathsf{Star}(A)$ (Protocol 4.2) If $|F| > n - t_a$, it sets $Q_a = F$ and broadcasts (async, A, Q_a).

---

[3] Had the network been synchronous, then we know that $T_{\mathsf{BC}} > \Delta$. Hence, $f_i(x)$ and the dealer's broadcast, both initiated simultaneously, will be received by P_i by T_{BC}.

7. (Conflict Resolution for Clique Expansion or Restart) At time $3T_{\mathsf{BC}}$, the parties do the following:
   (a) If (sync, G, Q_a) is received, then P_i verifies G, Q_a as follows. It checks the validity of G_i in the same way as in Step 7c. It checks if Q_a is a $(n-t_a)$-size clique in G_i including parties in U. If the verification passes, then set $b_i = 1$ and $b_i = 0$ otherwise and participate in an instance of Π_{BA}. If the protocol output is 1 then go to Protocol 6.2. Otherwise, wait for (async, A, Q_a) from the dealer.
   (b) If $(\mathsf{restart}, U)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 6.2. Otherwise, restart the protocol from Step 1.
   (c) If $(\mathsf{continue}, Q, G, V)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 6.2. Otherwise, when the output is 0, verify Q, G, V. For this, construct G_i exactly as the dealer did based on the broadcasts available at time $2T_{\mathsf{BC}}$ at Step 5. G is marked as invalid if
      i. it is different from G_i AND
      ii. there is a pair $P_j, P_k \notin U$ such that $R_j[k] \neq R_k[j]$ or there is a pair $P_j \notin U, P_k \in U$ such that $R_j[k] \neq f_k(j)$.

      Q is invalid if it is not a clique in a valid G of size at least $n - t_a$ and does not include parties in U. V is invalid if it is not a set of $(t_s - t_a) - |U|$ parties (vertices) outside $Q \cup U$ in a valid G.

      If Q, G, V are valid, then for each (P_j, P_k) who do not have an edge and $P_j \in V$, P_j broadcasts $f_j(k)$ and P_k broadcasts $f_k(j)$ if $f_j(x)$ and $f_k(x)$ if received from the dealer at time Δ. Otherwise, they broadcast $\perp$. Let V' be the set of parties in V and the parties they do not have an edge to.

      If G or Q or V from broadcast $(\mathsf{continue}, Q, G, V)$ is invalid, then wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 6.2 on receiving (async, A, Q_a).
   (d) If $\perp$ is received then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 6.2. Otherwise, wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 6.2 on receiving (async, A, Q_a).
8. (Clique Expansion or Restart (for the dealer)) At time $4T_{\mathsf{BC}} + T_{\mathsf{BA}}$, the dealer adds P_i in W if the broadcast of $P_i \in V'$ in the previous step is $\perp$ or if the broadcast is not $F(i, j)$. If $|W| > 0$, then the dealer sets $U = U \cup W$ and broadcasts $(\mathsf{restart}, U)$. Otherwise, the dealer sets clique $Q_a = Q \cup V$ and broadcasts (sync, G, Q_a).
9. (Local Computation: Deciding on exit route or restart (for all)) At time $5T_{\mathsf{BC}} + T_{\mathsf{BA}}$, every P_i does as follows:
   (a) If $(\mathsf{restart}, U)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 6.2. Otherwise, restart the protocol from Step 1 with U and W reset to $\emptyset$.
   (b) If (sync, G, Q_a) is received from the broadcast of the dealer it constructs G_i in the same way as in Step 7c. It then updates G_i based on the broadcasts received at time $4T_{\mathsf{BC}}+T_{\mathsf{BA}}$ and checks its validity as in Step 7c. Next, it checks if Q_a is a $(n-t_a)$-size clique in G_i including parties in U. If the verification passes, then set $b_i = 1$ and $b_i = 0$ otherwise and participate in an instance of Π_{BA}. If the protocol output is 1 then go to Protocol 6.2. Otherwise, wait for (async, A, Q_a) from the dealer.

(c) If $\perp$ is received from the broadcast, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 6.2. Otherwise, wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 6.2 on receiving (async, A, Q_a).

The following steps are executed by a party when it receives an output of 1 from any Π_{BA} instance. Otherwise, parties continue to participate in Π_{WSS} iterations.

Protocol 6.2: $\Pi_{\mathsf{WSS}}^{\mathsf{Output}}$

Condition for Output: Parties output via (async, A, Q_a) only after local time $(t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + 2T_{\mathsf{BA}})$. Parties output via (sync, G, Q_a) only before local time $(t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + 2T_{\mathsf{BA}})$.

Upon receiving (sync, G, Q_a) or (async, A, Q_a) from the dealer, each P_i verifies G, Q_a or A, Q_a as follows: It constructs G_i or A_i exactly the way the dealer does in the respective steps based on the broadcasts available until now. P_i continues to update G_i or A_i based on the broadcasts it receives if the edges in G (respectively A) are not a subset of the edges in G_i (resp. A_i) or Q_a is not a $(n - t_a)$-size clique in G_i (resp. A_i). Otherwise, it does the following:

1. If $P_i \in Q_a \setminus U$, then it sends $f_i(j)$ to every $P_j \notin Q_a \cup U$, waits for time 3Δ and outputs $f_i(x)$.
2. If $P_i \notin Q_a$ waits for 3Δ time[4] and upon receiving $t_s + t_a + 1$ points from parties in Q_a (P_i obtains points of parties in U from the dealer's broadcast) and does the following:
   - Upon receiving $t_s + t_a + 1 + x$ points, if $x \le t_a$ then P_i tries to correct up to x errors and simultaneously detect up to $t_a - x$ errors (Corollary 3.3). If the decoding is successful, then P_i outputs the reconstructed polynomial.
   - Upon receiving $t_s + t_a + 1 + x$ points, if $x > t_a$ then P_i tries to correct up to t_a errors and simultaneously detect up to $x - t_a$ errors (Corollary 3.4). If the decoding is successful, then P_i outputs the reconstructed polynomial. Otherwise, P_i detects that the network is synchronous. It then checks the following: If (sync, G, Q_a) is received at time $3T_{\mathsf{BC}}$, then it checks the validity of G_i in the same way as in Step 7c. It checks if Q_a is a $(n - t_a)$-size clique in G_i or including parties in U. If (sync, G, Q_a) is received at time $5T_{\mathsf{BC}}$, then it first updates G_i based on the broadcasts received at time $4T_{\mathsf{BC}}$ and checks its validity as in Step 7c. Next, it checks if Q_a is a $(n - t_a)$-size clique in G_i including parties in U.

     It outputs $f_i(x)$ if it is received from the dealer within Δ time from the start, Q_a does not include any P_j such that P_j's broadcast at time $2T_{\mathsf{BC}}$ is $f_{ji} \neq f_i(j)$ and the above verification passes. It outputs $\perp$ otherwise.

Theorem 6.3. *Let $T_{\mathsf{WSS}} = (t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + 2T_{\mathsf{BA}}) + 3\Delta$. Protocol Π_{WSS} is perfectly-secure against an adversary corrupting up to t_s parties in the synchronous network and up to t_a parties in the asynchronous network and has the following properties.*

[4] In a synchronous network, if some honest party validates Q_a at time T, other honest parties may receive and validate it by time at most $T + 2\Delta$ when the dealer is corrupt. Hence, their shares may reach parties outside Q_a at time $T + 3\Delta$. Upon receiving Q_a, each party outside it thus waits for 3Δ time to ensure that it receives all the honest parties' shares before starting error correction.

1. *Synchronous network:*
   (a) *t_s correctness: When the dealer is honest, at time T_{WSS}, all the honest parties output $f_i(x) = F(x, i)$ corresponding to $F(x, y)$ held by the dealer.*
   (b) *t_s privacy: The view of the adversary is independent of the honest dealer's secret s.*
   (c) *t_s weak commitment: When the dealer is corrupt, either no honest party computes an output or there exists a set of at least t_s+t_a+1 honest parties P_i such that each P_i outputs $f_i(x)$ where $f_i(x) = F'(x, i)$ for some (t_s, t_s) degree polynomial $F'(x, y)$. Moreover, if some honest party computes its output at $T \leq T_{\mathsf{WSS}}$ then all honest compute their output at the same time. If some honest party computes an output at time $T > T_{\mathsf{WSS}}$ then all the honest parties compute their output within $T + 2\Delta$.*
2. *Asynchronous network:*
   (a) *t_a correctness: When the dealer is honest, almost-surely all the honest parties output $f_i(x) = F(x, i)$ eventually where $F(x, y)$ is held by the dealer.*
   (b) *t_s privacy: The view of the adversary is independent of the honest dealer's secret s.*
   (c) *t_a strong commitment: When the dealer is corrupt, either no honest party computes an output or almost-surely each honest party P_i outputs $f_i(x)$ eventually such that $f_i(x) = F'(x, i)$ for some (t_s, t_s) degree polynomial $F'(x, y)$.*

Proof. We first prove the properties of Π_{WSS} in the synchronous network.

1. Synchronous network:
   (a) t_s correctness: Let the dealer be honest. Since the network is synchronous, we have that the adversary can corrupt up to t_s parties and the network delay is Δ. At the start of the protocol, we also have that W, U are empty. Given this, we have that within Δ time, all the parties will have their univariate polynomial shares. Further, each pair of honest parties P_i, P_j will exchange their common points on the polynomial within time 2Δ. By the liveness and validity property of broadcast in a synchronous network, we have that $(U, \{f_i(x)\}_{i \in U})$ will also be received by all the honest parties by time T_{BC}. Thus, we have that each honest party P_i will set $R_i[j] = \mathsf{OK}$ corresponding to every honest party P_j. Moreover, P_i sets $R_i[j] = f_i(j)$ corresponding to each $P_j \in U$ such that $f_i(j) = F(i, j)$. Thus, each honest P_i has at most t_s NRs corresponding to the corrupt parties. Further, the liveness and validity properties of broadcast ensure that the honest parties' broadcast instances successfully terminate with an output by time $2T_{\mathsf{BC}}$. Moreover, if $R_i[j] = f_{ij}$ is broadcasted by an honest P_i then it is guaranteed that $f_{ij} = F(i, j)$ indeed holds. Given that all the above conditions hold, an honest P_i is never added to W by the dealer. This implies that the dealer is bound to find a clique Q of size at least $n - t_s + |U|$ which contains all the honest parties and the parties in U. We now have the following cases to consider:
      i. **The dealer finds Q such that $|Q| \geq n - t_a$.** This implies that at time $2T_{\mathsf{BC}}$, the dealer receives $R_i[j] = \mathsf{OK}$ and $R_j[i] = \mathsf{OK}$ for each $P_i, P_j \in Q$. Due to the consistency property of broadcast in the synchronous network, we have that all the honest parties will indeed see the same R_i's at time $2T_{\mathsf{BC}}$ as that seen by the dealer. Further, the dealer sets $Q_a = Q$ and broadcasts (sync, G, Q_a) which will be received by all the honest parties by time $3T_{\mathsf{BC}}$. Consequently, each honest party P_i will construct the graph G_i exactly as the dealer, and hence its verification passes. Hence, all the honest parties will participate with input 1 in Π_{BA}, and due to its

liveness and validity, they will receive the output as 1 by time $3T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Further, every honest $P_i \in Q_a$ sends its share $f_i(j)$ to every $P_j \notin Q_a$. It waits for Δ time and outputs $f_i(x)$ at time $3T_{\mathsf{BC}} + \Delta$. Each $P_i \notin Q_a$ receives at least $|Q| - t_s \geq t_s + t_a + 1$ points from the honest parties in Q_a. We then have two cases to consider:

- If P_i receives up to t_a erroneous points from parties in Q_a, then by Corollary 3.3 it will recover the same polynomial after error correction as what the dealer shared and hence output the correct $f_i(x)$ at time $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$.
- If P_i receives more than t_a erroneous points from parties in Q_a, then by Corollary 3.4 we have that P_i will detect this. It in turn learns that the network is indeed synchronous. Moreover, an honest P_i would have received its share $f_i(x)$ from the dealer within time Δ and sent its pairwise points $f_i(j)$ to each P_j. Let the point received by $P_j \in Q_a$ be f_{ij}. If indeed $f_j(i) \neq f_{ij}$ did not hold for P_j, then P_j would have broadcasted $R_j[i] = f_j(i)$ by time $2T_{\mathsf{BC}}$. Given that the dealer is honest, we have that a P_j that broadcasted an incorrect value at $2T_{\mathsf{BC}}$ would be included in W and hence $P_j \notin Q_a$ which is a contradiction. Thus, it must hold that P_j either broadcasted $R_j[i] = \mathsf{NR}$ or $R_j[i] = F(i,j) = f_i(j)$. Thus, P_i can identify a corrupt P_j which sends an erroneous point. In this case, P_i outputs the correct polynomial received from the dealer $f_i(x)$ by time $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$.

ii. **The dealer broadcasts** $(\mathsf{restart}, U)$**.** In this case, we have that the dealer has added at least one party in the set W and hence added at least one new party in the set U. Due to the liveness and validity properties of broadcast in the synchronous network, we have that all the honest parties will receive the dealer's broadcast by time $3T_{\mathsf{BC}}$. Hence, all the honest parties will participate with input 0 in Π_{BA}, and due to its liveness and validity, they will receive the output as 0 by time $3T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Subsequently, they restart the protocol successfully and in synchronization with each other. Moreover, as argued before, it is guaranteed that no honest party gets added to W or U. Thus, after at most $t_s - t_a$ restarts, U will include at least $t_s - t_a$ corrupt parties. The dealer will thus make the polynomials of $t_s - t_a$ corrupt parties public in the subsequent run of the protocol and is guaranteed to find a clique of size $(n - t_a)$ which includes the $(n - t_s)$ honest parties and the $t_s - t_a$ parties from U whose polynomials are public. Hence, in the subsequent run, the prior case is guaranteed to occur and parties will successfully output shares on the dealer's polynomial.

iii. **The dealer broadcasts** $(\mathsf{continue}, Q, G, V)$**.** In this case, it must hold that $|Q| < n - t_a$, $|U| < t_s - t_a$ and $W = \phi$. Again, by the properties of broadcast, we have that the dealer's broadcasted message will be delivered to all the honest parties by time $3T_{\mathsf{BC}}$. Hence, all the honest parties will participate with input 0 in Π_{BA} and due to its liveness and validity, they will receive the output as 0 by time $3T_{\mathsf{BC}} + T_{\mathsf{BA}}$. Thus, they will proceed to check the validity of Q, G, V and identify that it is valid. It is guaranteed that each honest $P_j \in V$, has an edge with every honest P_k and the corresponding OK is received by all the honest parties, including the dealer by time $2T_{\mathsf{BC}}$. Hence, for each (P_j, P_k) pair where $P_j \in V$ and does not have an edge with some P_k, it is guaranteed that at least one of P_j, P_k is corrupt. Further, every honest P_k such that it does not have an edge with $P_j \in V$ broadcasts the correct $f_k(j)$ and is received by all the honest parties and the dealer by time $4T_{\mathsf{BC}}$. Hence, once again, no honest party gets added to W. We now have two cases to consider:

- The dealer broadcasts $(\mathsf{restart}, U)$. This implies that the broadcast of some P_k or P_j such that P_k does not have an edge with $P_j \in V$ results in a $\perp$ or results in value not equal to $F(k, j)$ at time $4T_{\mathsf{BC}} + T_{\mathsf{BA}}$. The dealer adds at least one party to W and hence adds at least one new party to U. By the argument above, we have that P_k or P_j added to W is guaranteed to be corrupt. The dealer then broadcasts $(\mathsf{restart}, U)$ which is received by all the honest parties by time $5T_{\mathsf{BC}}$. All the honest parties will participate with input 0 in Π_{BA} and due to its liveness and validity, they will receive the output as 0 by time $5T_{\mathsf{BC}} + 2T_{\mathsf{BA}}$. Thus, all honest parties restart the protocol in synchronization. By the same argument as earlier, upon at most $t_s - t_a$ restarts, we have that an honest dealer will conclude the protocol by finding a $|Q| \geq n - t_a$ which includes the $(n - t_s)$ honest parties and $t_s - t_a$ parties from U.
- The dealer broadcasts (sync, G, Q_a). In this case, it must hold that for every $P_j \in V$, such that (P_j, P_k) did not have an edge at time $2T_{\mathsf{BC}}$, both parties indeed broadcasted the correct value $F(k, j)$ by time $4T_{\mathsf{BC}}$, thus ensuring that (P_j, P_k) are now consistent. Given that a valid Q is of size $|Q| = n - t_s + |U|$ and the dealer has additionally resolved conflicts with $(t_s - t_a) - |U|$ parties, this implies that $Q_a = Q \cup V$ is indeed of size at least $n - t_a$. Hence, the dealer's broadcast of (sync, G, Q_a) actually contains a clique of the required size and will be received by the parties within time T_{BC}. Consequently, each honest party P_i will construct the graph G_i exactly as the dealer, hence its verification passes. All the honest parties will thus participate with input 1 in Π_{BA} and due to its liveness and validity, they will receive the output as 1 by time $5T_{\mathsf{BC}} + T_{\mathsf{BA}}$. The parties then compute their output similar to the first case (**The dealer finds Q such that** $|Q| \geq n - t_a$) at time $5T_{\mathsf{BC}} + 2T_{\mathsf{BA}} + 3\Delta$.

(b) t_s privacy: Observe that the only step at which the dealer reveals information regarding the secret (excluding the initial step of sharing the polynomial) corresponds to the public broadcast of $f_i(x)$ for parties in U. Note that a party P_i is added to U at time $2T_{\mathsf{BC}}$ if its broadcast corresponding to the pairwise consistency checks results in $\perp$, has more than t_s NRs or has an incorrect $f_i(j)$ value. Neither of these conditions holds true for an honest party; hence, an honest party does not get added to U at this time step. Further, parties also get included to U at time $4T_{\mathsf{BC}}$. Here, a party P_i may get added to U if its broadcast corresponding to a party $P_j \in V$ results in a $\perp$ or has an incorrect value. Given that an honest party receives an honest dealer's broadcast of $(\mathsf{continue}, Q, G, V)$ at time $3T_{\mathsf{BC}}$ and its own polynomial from the dealer in time Δ, it broadcasts the required (correct) values which are received by the dealer at time $4T_{\mathsf{BC}}$. Hence, an honest party does not get added to U. Thus, we have that from the dealer's communication, an adversary can learn at most t_s univariate polynomials corresponding to the corrupt parties, thus ensuring privacy.
Further, we show that the adversary does not learn any additional information from the broadcast of honest parties. During pairwise exchange, it is ensured that the honest parties successfully send common points to each other. Hence, every honest P_i broadcasts $R_i[j] = \mathsf{OK}$ corresponding to every honest P_j and does not reveal any information o an adversary. Further, an honest party only broadcasts points for a party in $P_j \in V$ such

that (P_i, P_j) does not have an edge by time $2T_{\mathsf{BC}}$. Given that this does not hold for any honest P_j as argued earlier, each $f_i(j)$ revealed by an honest party P_i corresponds to a corrupt P_j, thus not revealing any information to the adversary. In conclusion, the adversary cannot learn any information beyond (at most) t_s univariate polynomial shares it can obtain from (at most) t_s corrupt parties, ascertaining t_s privacy.

(c) t_s weak commitment: If no honest party computes an output, then the weak commitment holds trivially. Hence, we consider the case when there exists some honest party P_k which computes the output at time T. We further analyze this in the following cases:

i. **P_k computes the output via obtaining (sync, G, Q_a) in some iteration of the protocol:** In this case, it implies that P_k obtains the output of Π_{BA} as 1 either at time $3T_{\mathsf{BC}} + T_{\mathsf{BA}}$ or $5T_{\mathsf{BC}} + 2T_{\mathsf{BA}}$. This further implies that some honest party P_h participates in Π_{BA} with input 1. If not, then the liveness and validity of Π_{BA} would ensure that parties output 0 and not compute output via (sync, G, Q_a). Consider the case that P_h has $b_h = 1$ in Π_{BA} instance at time $3T_{\mathsf{BC}}$. In this case, note that P_h must have verified that the dealer's graph G is indeed the same as G_h constructed using the broadcast it receives by time $2T_{\mathsf{BC}}$. Moreover, Q_a also satisfies the requirements. By the consistency and liveness properties of broadcast in the synchronous network, we have that the output computed by all the honest parties in the broadcast instance of the dealer is the same at time $3T_{\mathsf{BC}}$. Similarly, by the properties of broadcast, it also holds that the output of broadcast instances computed by all the honest parties at time $2T_{\mathsf{BC}}$ is identical to that computed by P_h. Hence, it must hold that (sync, G, Q_a) is received and verified by all the parties successfully. Hence, all the parties must have set $b_i = 1$ in the instance of Π_{BA} and obtained the output 1 at time $3T_{\mathsf{BC}}+T_{\mathsf{BA}}$. Given that $|Q_a| \geq n - t_a$ and all the parties are consistent with each other, we have that all the honest parties in Q_a output $f_i(x)$ such that $F'(x, i) = f_i(x)$ for some (t_s, t_s)-degree bivariate polynomial F' at time $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$ in that iteration. Now consider an honest party $P_i \notin Q_a$. By time $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$, P_i is guaranteed to receive $f_j(i)$ from each honest $P_j \in Q_a$. If P_i receives at most t_a erroneous points (points not lying on $F'(x, y)$) and additionally it holds $f_i(x) = F'(x, i)$ received from the dealer at time Δ in this iteration, then by Corollary 3.3, P_i must have successfully reconstructed the same $f_i(x)$ from the points of parties in Q_a and set it as its output, thus ensuring the correct output. On the other hand, suppose P_i receives more than t_a erroneous points from the parties in Q_a. In this case, by Corollary 3.4, P_i identifies that the network is synchronous. If P_i has not received $f_i(x)$ from the dealer by time Δ then it outputs $\perp$. Otherwise, P_i had received its polynomial by time Δ and sent $f_i(j)$ to every P_j, an honest party $P_j \in Q_a$ for whom the value did not match would have indeed broadcasted its own value in $R_j[i]$ at time $2T_{\mathsf{BC}}$ or its value $f_j(i)$ would already be public if $P_j \in U$. If indeed $P_j \in Q_a$ has broadcasted $R_j[i]$ at time $2T_{\mathsf{BC}}$ or $P_j \in U$ has $f_j(i)$ which is not equal to $f_i(j)$, then P_i identifies that the dealer is corrupt since it has included $P_j \in Q_a$ while it has sent $f_i(x)$ such that $f_i(j) \neq F'(i, j)$. Hence, P_i outputs $\perp$. On the other hand, if P_j has broadcasted $R_j[i] = f_i(j)$ or $R_j[i] = \mathsf{NR}$ at time $2T_{\mathsf{BC}}$, then P_i identifies that P_j is corrupt and has sent it an incorrect value at time $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$. In this case, P_i ignores f_{ji} sent by P_j. If P_i successfully reconstructs a polynomial after discarding these points which is equal to $f_i(x)$ received from the dealer, then it is guaranteed

that the polynomial is indeed consistent with all the honest parties in Q_a. This is because P_i only discards the points of corrupt parties who behaved inconsistently at times $2T_{\mathsf{BC}}$ and $3T_{\mathsf{BC}} + T_{\mathsf{BA}} + 3\Delta$. Thus, we have that an honest $P_i \notin Q_a$ indeed outputs $f_i(x) = F'(x, i)$, where $F'(x, y)$ is the (t_s, t_s)-degree bivariate polynomial defined by the honest parties in Q_a.
The other case, that P_h has $b_h = 1$ in Π_{BA} instance at time $5T_{\mathsf{BC}} + 2T_{\mathsf{BA}}$ follows similarly. Here, it must also hold that the output of Π_{BA} at time $3T_{\mathsf{BC}} + T_{\mathsf{BA}}$ was 0. Otherwise, P_h and all the honest parties would proceed as in the former case. Thus we have that P_h received (sync, G, Q_a) at time $5T_{\mathsf{BC}} + T_{\mathsf{BA}}$ and accepted it, then it implies that it also received a valid $(\mathsf{continue}, Q, G, V)$ at time $3T_{\mathsf{BC}}$ and broadcasts of parties corresponding to V at time $4T_{\mathsf{BC}} + T_{\mathsf{BA}}$. If not, then P_h would have either received an invalid (sync, G, Q_a) or $(\mathsf{restart}, U)$ or $\perp$ or an invalid G, Q, V in $(\mathsf{continue}, Q, G, V)$. In either of these cases, P_h would not have proceeded to execute steps designated for time beyond $4T_{\mathsf{BC}} + T_{\mathsf{BA}}$ and not participated in Π_{BA} with input 1, which is a contradiction. Since P_h received valid broadcasts from the dealer at time $3T_{\mathsf{BC}}$ and $5T_{\mathsf{BC}} + T_{\mathsf{BA}}$, by the consistency and liveness properties of broadcast in the synchronous network, we have that all the honest parties also received it at the designated time steps. Following this, the argument for t_s weak commitment follows exactly as that for the previous case, where all the honest parties either output shares on the same polynomial or $\perp$ at time $5T_{\mathsf{BC}} + 2T_{\mathsf{BA}} + 3\Delta$.

ii. P_h **computes the output via obtaining** (async, A, Q_a) **at time** T **in some iteration of the protocol:** First note that this implies that none of the $(t_s - t_a)$ iterations terminated via the (sync, G, Q_a) path for P_h. Since the decision of output computation is taken via Π_{BA}, by its liveness and consistency property, it is guaranteed that no honest party computes its output via (sync, G, Q_a). Note that since P_h computes the output at time T, it implies that the dealer's broadcast indeed has a valid clique which was received and verified by P_h by time $T - 3\Delta$. Moreover, by the fallback consistency property of broadcast in a synchronous network, we have that all the honest parties will receive (async, A, Q_a) and the AOK messages to validate its correctness within time $(T - 3\Delta) + 2\Delta = T - \Delta$. By the fact that $|Q_a| \geq n - t_a$ and includes at least $t_s + t_a + 1$ honest parties, we have that $f_i(x)$ held by each $P_i \in Q_a$ is such that $f_i(x) = F'(x, i)$ for some (t_s, t_s)-degree bivariate polynomial $F'(x, y)$. Moreover, each $P_i \in Q_a$ will compute an output by time $(T - \Delta) + 3\Delta = T + 2\Delta$. Consider an honest $P_i \notin Q_a$. As before, since the network is synchronous, it is guaranteed to receive $f_j(i)$ from each honest $P_j \in Q_a$ by time at most T. Since $P_i \notin Q_a$ waits for time at least 3Δ upon accepting (async, A, Q_a), it is guaranteed to receive the points of all the honest parties before proceeding for reconstruction. At this time, if it receives less than t_a erroneous points, then it successfully recovers the correct polynomial $f_i(x) = F'(x, i)$ defined by the honest parties in Q_a and computes an output at time $T + 2\Delta$. Otherwise, P_i identifies that the network is synchronous. In this case, if the dealer was honest then P_i knows that it would have terminated via (sync, G, Q_a). Hence, P_i identifies that the dealer is corrupt and outputs $\perp$. Thus, we have that all the honest parties output $f_i(x)$ such that $f_i(x) = F'(x, i)$ holds for some (t_s, t_s) degree polynomial $F'(x, y)$. Moreover, all honest parties compute their output within a delay of 2Δ from each other.

2. Asynchronous network: We now prove the properties of Π_{WSS} in the asynchronous network.

   (a) t_a correctness: Let the dealer be honest. Since the network is asynchronous, we have that the adversary can corrupt at most t_a parties. Given this and the fact that all the honest parties' messages (including the dealer's) get delivered eventually, we have that the set of all the honest parties eventually constitutes an $(n - t_a)$ sized clique. Thus we have that via the sequence of steps corresponding to an asynchronous network, the dealer will eventually broadcast (async, A, Q_a) which will be validated by all the honest parties. Moreover, each honest $P_i \in Q_a$ will output a correct $f_i(x)$ which it received from an honest dealer. Now consider the case of an honest party P_i outside Q_a. An honest party $P_i \notin Q_a$ will eventually receive $f_j(i)$ from every honest party $P_j \in Q_a$. Since at most t_a parties are corrupt and can send erroneous points to P_i, by Corollary 3.3 we have that P_i will successfully reconstruct and output a correct $f_i(x)$ consistent with the dealer's bivariate polynomial. Moreover, if some honest party actually receives (sync, G, Q_a) and obtains an output, by the consistency of Π_{BA} we have that some honest party participated with input 1 in the agreement protocol. Hence, all the honest parties will eventually receive the output as 1 and compute their output correctly.

   (b) t_s privacy: As in the synchronous case, the only step at which the dealer reveals information regarding its secret beyond the sharing of polynomials is when it broadcasts $f_i(x)$ corresponding to each $P_i \in U$. Note that the dealer adds a party in U if $R_i[j] \neq F(i, j)$ or P_i's broadcast results in NR for more than t_s parties. Given that the network is asynchronous, an honest P_i may thus get added to U. However, it is ensured that the dealer reveals $f_i(x)$ for at most $t_s - t_a$ such parties. Thus, in the worst case, we have that the adversary learns t_s such univariate polynomials $f_i(x)$ corresponding to t_a corrupt parties and additionally $t_s - t_a$ honest parties in U. Hence, the adversary can learn exactly as much information regarding the secret as in the synchronous case, thus ensuring privacy. Further, we show that the adversary does not learn anything beyond t_s univariate polynomials, even from the broadcast of the parties. First, observe that during the pairwise exchange, if an honest party P_i does not receive $f_j(i)$ from an honest P_j then it broadcasts NR at time T_{BC}. When it eventually receives $f_j(i)$ from P_j, it is guaranteed that $f_j(i) = f_i(j)$ and hence P_i broadcasts AOK_j. Hence, the broadcasts corresponding to pairwise checks do not reveal any information regarding the honest parties' secrets. Next, we have that parties may reveal information regarding their polynomial if they receive (sync, Q, G, V) from the dealer. In this case again, it is possible that V contains an honest party P_i for whom all the parties P_j not having an edge with P_i reveal their common point $f_j(i)$. However, note again that $|U \cup V| t_s - t_a$, and hence, at most $t_s - t_a$ honest parties' polynomials may be revealed to the adversary. By the same argument as earlier, we have that the adversary gains no information beyond t_s univariate polynomials on the dealer's polynomial, which is exactly as in the case of the synchronous network. Thus, we have that t_s privacy holds even in the asynchronous network.

   (c) t_s strong commitment: We now show that when the network is asynchronous, irrespective of the adversary's behavior, each honest party P_i will output $f_i(x)$ such that $f_i(x) = F'(x, i)$ for some (t_s, t_s) degree polynomial $F'(x, y)$.
   We will first show that given two cliques, say Q_a and Q'_a, each of size at least $n - t_a$, we have that the shares held by the honest parties in Q_a as well as Q'_a are consistent with the same (t_s, t_s) degree bivariate polynomial. Given that $|Q_a| \geq n - t_a$, $|Q'_a|$ and

we have a total of n parties, it must hold that $|Q_a \cap Q'_a| \geq n - 2t_a$. Moreover, we know that $n - 2t_a \geq 2t_s + 1$. Hence, it holds that $Q_a \cap Q'_a$ contains at least $t_s + 1$ honest parties who hold polynomials that define a unique (t_s, t_s) degree bivariate polynomial, say $F(x, y)$. Let H be the set of (at least) $t_s + 1$ honest parties such that an honest $P_i \in H$ when $P_i \in Q_a \cap Q'_a$. Further, since both Q_a and Q'_a are cliques, it holds that each honest $P_j \in Q_a \cup Q'_a$ is consistent with every $P_i \in H$. Given that the degree t_s polynomial $f_j(x)$ held by P_j is consistent with (at least) $t_s + 1$ points of $F(x, j)$, it must hold that $f_j(x) = F(x, j)$ for each $P_j \in Q_a$ as well as every $P_j \in Q'_a$. This ensures that all the honest parties belonging to different cliques of size at least $n - t_a$ are guaranteed to hold polynomials consistent with a unique (t_s, t_s) degree bivariate polynomial. Given this, we now argue that our protocol ensures t_s strong commitment in an asynchronous network.

If no honest party computes an output when the dealer is corrupt, commitment holds trivially. Thus, we consider the case when some honest party P_h computes an output. Note that this implies that P_h has received either (sync, G, Q_a) and (async, A, Q_a) and verified it to compute an output. In the former case, we have that P_h received 1 as the output of Π_{BA} during some iteration of the protocol. Hence, there exists some honest party that participated in an instance of Π_{BA} with input 1. If not, then by the validity of Π_{BA}, all the honest parties would have output 0. Hence, P_h would not have computed its output via (sync, G, Q_a) which is a contradiction. By the consistency of Π_{BA} it thus holds that all the honest parties will receive 1 as the output of Π_{BA} and eventually compute their output. This is because parties in Q_a will eventually verify the clique and compute their output. For parties outside, Q_a, by Corollary 3.3, we have that they will be able to reconstruct the polynomial that is consistent with the honest parties in Q_a. Similarly, if P_h computes its output via (async, A, Q_a), the same argument holds.

□

7 Verifiable Secret Sharing

The weak secret sharing protocol falls short of providing the properties of verifiable secret sharing. This is because, when the dealer is corrupt and the network is synchronous, it is possible that some honest parties, specifically the parties lying outside the $n - t_a$-Star, may not receive their shares. To fix this, and ensure that all or none of the honest parties receive their shares, we follow the approach of [3]. As described in [3], this results in a verifiable secret sharing protocol Π_{VSS} with two layers. The VSS protocol ensures that all the parties hold a degree-t_s Shamir-sharing of the dealer's input secret.

Here, we give a complete description of our verifiable secret sharing protocol, which allows a dealer to generate a degree-t_s sharing of its input among parties, followed by its proof.

Protocol 7.1: Π_{VSS}

Input: The dealer holds a secret $s \in \mathbb{F}$.

Initialisation: The dealer initialises two sets W, U to ϕ. Only W is reset in every run to $\emptyset$.

1. (Polynomial Share Distribution) The dealer chooses a symmetric bivariate polynomial $F(x, y)$ of degree t_s in both x, y and delivers $f_i(x) = F(x, i)$ to P_i. If $|U| > t_s - t_a$, then assign U to be the set of first $t_s - t_a$ parties lexicographically. The dealer broadcasts $(U, \{f_i(x)\}_{i \in U})$.

2. (Pair-wise exchange) At time T_{BC}, if $f_i(x)$ is received at time Δ then every P_i participates in an instance of Π_{WSS} as the dealer, say $\Pi^{(i)}_{\mathsf{WSS}}$ with input $f_i(x)$. P_i also participates in $\Pi^{(j)}_{\mathsf{WSS}}$ instances for every $j \in \{1, \dots, n\} \setminus U$.
3. (Pair-wise Consistency Check) At time $T_{\mathsf{BC}} + T_{\mathsf{WSS}}$ P_i prepares a vector R_i of length n as follows and broadcasts it. It sets $R_i[j] = \mathsf{NR}$ for all j if either of the following happens:
   (a) it receives no $f_i(x)$
   (b) the dealer's broadcast results in $\perp$
   (c) some $f_j(x)$ in the broadcast $(U, \{f_i(x)\}_{i \in U})$ is of degree more than t_s
   (d) there are indices j, k such that $f_j(k) \neq f_k(j)$ in the broadcast $(U, \{f_i(x)\}_{i \in U})$

   Otherwise, it sets R_i as follows. (1) if $P_j \in U$, then $R_i[j] = f_i(j)$ (2) if $P_j \notin U$, then set (a) $R_i[j] = \mathsf{NR}$ if f_{ji} is not computed as output in P_j's instance $\Pi^{(j)}_{\mathsf{WSS}}$, (b) $R_i[j] = f_i(j)$ if f_{ji} is received as output from $\Pi^{(j)}_{\mathsf{WSS}}$ and $f_i(j) \neq f_{ji}$, (c) $R_i[j] = \mathsf{OK}$ otherwise.
4. (Asynchronous Pair-wise Consistency Checking) The parties execute the following steps as and when they receive the required values. On receiving the broadcast $(U, \{f_i(x)\}_{i \in U})$ and polynomial $f_i(x)$ from the dealer, every $P_i \notin U$ participates in an instance of Π_{WSS} as the dealer, say $\Pi^{(i)}_{\mathsf{WSS}}$ with input $f_i(x)$. P_i also participates in $\Pi^{(j)}_{\mathsf{WSS}}$ instances for every $j \in \{1, \dots, n\}$. P_i broadcasts AOK_j if (a) f_{ji} is computed from $\Pi^{(j)}_{\mathsf{WSS}}$ from $P_j \notin U$ and $f_i(j) = f_{ji}$ (b) $f_j(i)$ for $P_j \in U$ satisfies $f_i(j) = f_j(i)$.
5. (Restart or Clique Finding) At time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, the dealer puts $P_i \notin U$ in W if either happens (a) P_i's broadcast of R_i resulted in $\perp$ or (b) P_i's broadcasted R_i has more than t_s NRs or (c) $R_i[j] \neq F(i,j)$ when $R_i[j] \neq \mathsf{OK}$ and $R_i[j] \neq \mathsf{NR}$.
   The dealer makes a graph G with n vertices corresponding to n parties. There is an edge when $R_i[j] = R_j[i] = \mathsf{OK}$. There is no edge if $R_i[j] = \mathsf{NR}$ or $R_j[i] = \mathsf{NR}$. The dealer finds a clique Q of size $n - t_s + |U|$ in the graph including U. If $|Q| \geq n - t_a$, then the dealer sets $Q_a = Q$ and broadcasts (sync, G, Q_a). Otherwise, if $|W| > 0$, then the dealer sets $U = U \cup W$ and broadcasts $(\mathsf{restart}, U)$. Otherwise, it broadcasts $(\mathsf{continue}, Q, G, V)$, where V is a set of $(t_s - t_a) - |U|$ parties (vertices) outside $Q \cup U$.
6. (Asynchronous Clique Finding) The dealer executes the following steps as and when it receives the required messages. First, the dealer initiates a graph A with parties as vertices with edges between a pair of parties in U. On receiving broadcasts AOK_{ij} and AOK_{ji} from $P_i, P_j \notin U$, it adds an edge between P_i, P_j. On receiving broadcast AOK_{ij} from $P_i \notin U, P_j \in U$, it adds an edge between P_i, P_j. Each time there is an update in A, it invokes $(C, D, E, F) \leftarrow \mathsf{Star}(A)$ (Protocol 4.2) If $|F| > n - t_a$, it sets $Q_a = F$ and broadcasts (async, A, Q_a).
7. (Conflict Resolution for Clique Expansion or Restart) At time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, the parties do the following:
   (a) If (sync, G, Q_a) is received, then P_i verifies G, Q_a as follows. It checks the validity of G_i in the same way as in Step 7c. It checks if Q_a is a $(n - t_a)$-size clique in G_i including parties in U. If the verification passes, then set $b_i = 1$ and $b_i = 0$ otherwise and participate in an instance of Π_{BA}. If the protocol output is 1 then go to Protocol 7.2. Otherwise, wait for (async, A, Q_a) from the dealer.
   (b) If $(\mathsf{restart}, U)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 7.2. Otherwise, restart the protocol from Step 1.

(c) If $(\mathsf{continue}, Q, G, V)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 7.2. Otherwise, when the output is 0, verify Q, G, V. For this, construct G_i exactly as the dealer did based on the broadcasts available at time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$ at Step 5. G is marked as invalid if

i. it is different from G_i AND

ii. there is a pair $P_j, P_k \notin U$ such that $R_j[k] \neq R_k[j]$ or there is a pair $P_j \notin U, P_k \in U$ such that $R_j[k] \neq f_k(j)$.

Q is invalid if it is not a clique in a valid G of size at least $n - t_a$ and does not include parties in U. V is invalid if it is not a set of $(t_s - t_a) - |U|$ parties (vertices) outside $Q \cup U$ in a valid G.

If Q, G, V are valid, then for each (P_j, P_k) who do not have an edge and $P_j \in V$, P_j broadcasts $f_j(k)$ and P_k broadcasts $f_k(j)$ if $f_j(x)$ and $f_k(x)$ if received from the dealer at time Δ. Otherwise, they broadcast $\perp$. Let V' be the set of parties in V and the parties they do not have an edge to.

If G or Q or V from broadcast $(\mathsf{continue}, Q, G, V)$ is invalid, then wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 7.2 on receiving (async, A, Q_a).

(d) If $\perp$ is received then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 7.2. Otherwise, wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 7.2 on receiving (async, A, Q_a).

8. (Clique Expansion or Restart (for the dealer)) At time $4T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$, the dealer adds P_i in W if the broadcast of $P_i \in V'$ in the previous step is $\perp$ or if the broadcast is not $F(i, j)$. If $|W| > 0$, then the dealer sets $U = U \cup W$ and broadcasts $(\mathsf{restart}, U)$. Otherwise, the dealer sets clique $Q_a = Q \cup V$ and broadcasts (sync, G, Q_a).

9. (Local Computation: Deciding on exit route or restart (for all)) At time $5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$, every P_i does as follows:

(a) If $(\mathsf{restart}, U)$ is received, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 7.2. Otherwise, restart the protocol from Step 1 with U and W reset to $\emptyset$.

(b) If (sync, G, Q_a) is received from the broadcast of the dealer it constructs G_i in the same way as in Step 7c. It then updates G_i based on the broadcasts received at time $4T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$ and checks its validity as in Step 7c. Next, it checks if Q_a is a $(n - t_a)$-size clique in G_i including parties in U. If the verification passes, then set $b_i = 1$ and $b_i = 0$ otherwise and participate in an instance of Π_{BA}. If the output of the protocol is 1 then go to Protocol 7.2. Otherwise, wait for (async, A, Q_a) from the dealer.

(c) If $\perp$ is received from the broadcast, then set $b_i = 0$ and participate in an instance of Π_{BA}. If the output is 1, then go to Protocol 7.2. Otherwise, wait until a broadcast (async, A, Q_a) from the dealer is received. Go to Protocol 7.2 on receiving (async, A, Q_a).

Protocol 7.2: $\Pi_{\mathsf{VSS}}^{\mathsf{Output}}$

Condition for Output: Parties output via (async, A, Q_a) only after local time $(t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + 2T_{\mathsf{BA}})$. Parties output via (sync, G, Q_a) only before local time $(t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + 2T_{\mathsf{BA}})$.

Upon receiving (sync, G, Q_a) or (async, A, Q_a) from the dealer, each P_i verifies G, Q_a or A, Q_a as follows: It constructs G_i or A_i exactly the way the dealer does in the respective steps based on the broadcasts available until now. P_i continues to update G_i or A_i based on the broadcasts it receives if the edges in G (respectively A) are not a subset of the edges in G_i (resp. A_i) or Q_a is not a $(n - t_a)$-size clique in G_i (resp. A_i). Otherwise, it does the following:

1. If $P_i \in Q_a$, outputs $f_i(x)$.
2. If $P_i \notin Q_a$ then and upon computing f_{ji} as output from $\Pi^{(j)}_{\mathsf{WSS}}$ corresponding to $t_s + 1$ parties $P_j \in Q_a$, P_i reconstructs its polynomial $f_i(x)$ and outputs it.

Theorem 7.3. *Let $T_{\mathsf{VSS}} = (t_s - t_a + 1) \cdot (5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + 2T_{\mathsf{BA}})$. Protocol Π_{VSS} is perfectly-secure against an adversary corrupting up to t_s parties in the synchronous network and up to t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *t_s correctness: When the dealer is honest, at time T_{VSS}, all the honest parties output $s_i = f_i(0)$.*
   (b) *t_s privacy: The view of the adversary is independent of the honest dealer's secret s.*
   (c) *t_s strong commitment: When the dealer is corrupt, either no honest party computes an output or each honest party P_i such that each P_i outputs s_i. Moreover, it holds that $s_i = f'(i)$ for some degree-t_s polynomial $f'(x)$. Also, if some honest party outputs by time T, then all honest parties have an output by time $T + 2\Delta$.*
2. *Asynchronous network:*
   (a) *t_a correctness: When the dealer is honest, almost-surely all the honest parties output $s_i = f_i(0)$ eventually.*
   (b) *t_s privacy: The view of the adversary is independent of the honest dealer's secret s.*
   (c) *t_a strong commitment: When the dealer is corrupt, either no honest party computes an output or almost-surely each honest party P_i outputs s_i eventually such that $s_i = f'(i)$ for some degree-t_s polynomial $f'(x)$.*

Proof. At a very high level, the proof follows closely to that of Π_{WSS}. We first prove the properties of Π_{VSS} in the synchronous network.

1. Synchronous Network:
   (a) t_s correctness: Consider the dealer to be honest. Given that the network is synchronous, we have that the network has a delay of at most Δ. Thus, each honest party P_i will receive its $f_i(x)$ from the dealer within time Δ. Moreover, the dealer's broadcast of $(U, \{f_i(x)\}_{i \in U})$ will be received within time T_{BC} from the start and initiate the instance of $\Pi^{(i)}_{\mathsf{WSS}}$. Further, every honest party will also participate in the instances initiated by all the other honest parties. We have that by time $T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, all the honest parties will compute output in the Π_{WSS} instance of every other honest party. Thus, we have that by time $T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, P_i has all the required information to compute the vector R_i. Hence, it will compute R_i such that $R_i[j] = \mathsf{OK}$ for every honest P_j and the correct $f_i(j)$

corresponding to every $P_j \in U$ and broadcasts it. By the liveness and validity property of broadcast in the synchronous network, we have that all the honest parties broadcast will be received successfully by time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. Hence, we have that no honest party gets added to W. Similar to Π_{WSS}, we now have the following cases to consider:

i. **The dealer finds a Q such that $|Q| \geq n - t_a$.** This implies that the dealer received the broadcasts of all the parties in Q by time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. By the consistency property of broadcast, we have that all the honest parties would also have received the same. Further, we have that the dealer will broadcast (sync, G, Q_a) which will be received by all the parties at time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. And hence, all the honest parties will participate in Π_{BA} with input 1. By the liveness and validity of Π_{BA} in the synchronous network, all the honest parties will output 1 at time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$ and hence compute the output as follows. Every $P_i \in Q_a$ will output the polynomial $f_i(x)$ it received from the dealer. Now consider an honest party $P_i \notin Q_a$. Since $|Q_a| \geq n - t_a$, we have that at least $n - t_a - t_s \geq t_s + 1$ honest parties. Thus, it holds that P_i must have computed f_{ji} as the output in P_j's instance of $\Pi_{\mathsf{WSS}}^{(j)}$. Hence, P_i has at least $t_s + 1$ points on a t_s degree polynomial. We now consider the case when P_i has computed its output in $\Pi_{\mathsf{WSS}}^{(j)}$ for some corrupt party $P_j \in Q_a$. Since (sync, G, Q_a) is such that the dealer honestly computed Q_a, it must hold that Q_a was indeed a clique at time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. This implies that each honest $P_k \in Q_a$ is broadcasted $R_k[j] = \mathsf{OK}$ corresponding to every corrupt $P_j \in Q_a$. By the t_s weak commitment property of Π_{WSS}, we have that all the honest parties would have indeed computed f_{kj} which lie on a unique t_s degree polynomial. Given that at least $t_s + 1$ honest parties broadcasted OK to such a corrupt party P_j, it must indeed hold that P_j participated with the dealer's correct polynomial $f_j(x)$ in $\Pi_{\mathsf{WSS}}^{(j)}$. Hence, we have that even if $P_i \notin Q_a$ has computed an output f_{ji} in $\Pi_{\mathsf{WSS}}^{(j)}$ corresponding to a corrupt P_j, it must hold that $f_{ji} = f_i(j)$. Hence, the polynomial interpolated by $P_i \notin Q_a$ is indeed $f_i(x) = F(x, i)$ where $F(x, y)$ is the (t_s, t_s) degree bivariate polynomial held by the dealer.

ii. **The dealer broadcasts $(\mathsf{restart}, U)$.** In this case, we have that the dealer added at least one party to W, and hence added at least one new party to U. Further, by the validity of broadcast, all the honest parties will receive $(\mathsf{restart}, U)$ at time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}}$ and set their input to Π_{BA} as 0. By the validity of Π_{BA}, all parties will output 0 at time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$ and consequently, restart the protocol in synchronization. Moreover, note that a party is added to W if and only if it broadcasts an incorrect value or its broadcast results in more than t_s NRs. However, given that each honest P_i computes f_{ji} in $\Pi_{\mathsf{WSS}}^{(j)}$ corresponding to every honest party P_j, and it receives $f_i(x)$ from the dealer within Δ time, we have that $R_i[j] = \mathsf{OK}$ for every honest P_j. Moreover, for every corrupt P_j such that $f_{ji} \neq f_i(j)$, it holds that $R_i[j] = f_i(j)$ broadcasted by P_i is indeed the correct value. Hence, an honest party is never added to W and hence not added to U. Given this observation, we have that upon $t_s - t_a$ restarts of the protocol, an honest dealer would have added $t_s - t_a$ corrupt parties to U, and hence their polynomials would be public. Thus, in the subsequent iteration, the dealer is bound to find a clique of size $n - t_a$ and successfully terminate via the former path of (sync, G, Q_a).

iii. **The dealer broadcasts** $(\mathsf{continue}, Q, G, V)$**.** This implies that $|Q_a| < n - t_a$, $|U| < t_s - t_a$ and $W = \phi$. Since the dealer broadcasts this at time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, we have that all the honest parties receive it by $3T_{\mathsf{BC}} + T_{\mathsf{WSS}}$ and participate with input 0 in Π_{BA}. Parties will thus output 0 at time $3T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$ and proceed to verify the dealer's broadcasted sets. By the validity of broadcast at time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, we have that all the honest parties will identify Q, G, V to be valid. Moreover, we have that an honest $P_i \in V$, would have computed an output in $\Pi_{\mathsf{WSS}}^{(j)}$ every honest P_j and hence neither P_i nor P_j broadcast their value at this stage. Also, each honest P_i broadcasts the correct $f_i(j)$ for every corrupt $P_j \in V$ which is received by all the honest parties including the dealer at time $4T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$. Hence, an honest party does not get added to W. We now have two cases to consider:

- The dealer broadcasts $(\mathsf{restart}, U)$. This implies that the broadcast of some corrupt party P_j either resulted in a $\perp$ or had an incorrect value. In either case, the dealer adds this party to W and thus has identified a new party to be added to U. The dealer's broadcast of $(\mathsf{restart}, U)$ is received by all the honest parties by time $5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$ who participated with input 0 in an instance of Π_{BA}. Thus, by the validity of Π_{BA}, the honest parties will obtain 0 as the output at time $5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + 2T_{\mathsf{BA}}$ and restart the protocol in synchronization.
- The dealer broadcasts (sync, G, Q_a). This implies that for every $P_j \in V$ such that (P_j, P_k) did not have an edge, both P_j and P_k broadcasted the correct $f_j(k)$ by time $4T_{\mathsf{BC}} + T_{\mathsf{WSS}} + T_{\mathsf{BA}}$. By the validity of broadcast, we have that all the honest parties indeed have the same output. Due to this, parties will participate in Π_{BA} with input 1. By the validity of Π_{BA}, we have that all the honest parties will output 1 at time $5T_{\mathsf{BC}} + T_{\mathsf{WSS}} + 2T_{\mathsf{BA}}$ and compute their output. The output computation will be successful due to the same argument as the first case (**The dealer finds a Q such that $|Q| \geq n - t_a$.**) and hence we avoid repetition.

In all the above cases, note that parties compute their output within time T_{VSS}.

(b) t_s privacy: Apart from sending the pairwise shares to each party, the dealer reveals information corresponding to its secret only when it broadcasts $f_i(x)$ corresponding to every $P_i \in U$. Moreover, a party P_i is added to U only if it broadcasts the incorrect value corresponding to $f_i(j)$ or its broadcasts result in a $\perp$ or more than t_s NRs. Given this, we note that no honest party gets added to U. Thus, we have that every party in U is corrupt when the dealer is honest and hence already knows the $f_i(x)$ broadcasted by the dealer. Now consider the values broadcasted by the honest parties. Since every honest P_i computes an output in $\Pi_{\mathsf{WSS}}^{(j)}$ instance of every honest P_j, we have that by time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, P_i broadcasts their $R_i[j] = \mathsf{OK}$. By the validity property of broadcast, this will be received by all the honest parties including the dealer and the consistency graph constructed by all the honest parties contains an edge for every honest (P_i, P_j). The only other time step at which an honest party P_i broadcasts $f_i(j)$ for some party P_j is when the dealer broadcasts $(\mathsf{continue}, Q, G, V)$. Again, at this step, an honest $P_i \in V$ will only broadcast the correct $f_i(j)$ corresponding to every corrupt P_j. And similarly, an honest $P_i \notin V$ will broadcast $f_i(j)$ for every corrupt $P_j \in V$. These are the values that the adversary already knows having obtained the $f_j(x)$ corresponding to every P_j and hence does not learn anything additionally. Finally, we have that for every corrupt P_j, the adversary learns $f_i(j)$ corresponding to an honest P_i in its instance of $\Pi_{\mathsf{WSS}}^{(i)}$.

However, this information is already available to the adversary due to its univariate polynomial share $f_j(x)$. Further, the t_s privacy of Π_{WSS} ensures that the adversary's view remains independent of an honest party's polynomial $f_i(x)$. In conclusion, we have that the adversary can learn at most t_s univariate polynomials $f_j(x)$ corresponding to (at most) t_s corrupt parties, thus ensuring t_s privacy.

(c) t_s strong commitment: Consider the case when the dealer is corrupt. If no honest party computes an output, then strong commitment holds trivially. We thus consider the case when some honest party, say P_k, computes its output. We now have two cases to consider:

i. **P_k computes the output by obtaining (sync, G, Q_a) in some iteration of the protocol.** This implies that P_k received 1 as the output of Π_{BA} in some iteration of the protocol before T_{VSS}. This further implies that there exists some honest party P_h which participated in Π_{BA} with input 1. If not, then all the honest parties would have set their input as 0, and by the validity property of Π_{BA}, all the parties would have received 0. In this case, parties would not have output via (sync, G, Q_a) which is a contradiction. Thus, it must be that some P_h set $b_h = 1$ as its input to Π_{BA}. This also implies that P_h received the dealer's broadcasts as well as the necessary broadcasts from the parties as per the synchronous time steps and verified it. By the liveness and consistency properties of broadcast, we thus have that all the honest parties must have computed the same output in all the broadcast instances and set their input to Π_{BA} as 1. This in turn implies that all the honest parties will compute their output via (sync, G, Q_a). Further, since accepting (sync, G, Q_a) involves verifying the dealer's graph based on the broadcast of parties at time $2T_{\mathsf{BC}} + T_{\mathsf{WSS}}$, it must hold that honest parties indeed broadcasted their R_i vector at time $T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. This also implies that there exist honest parties obtained the output of $\Pi^{(j)}_{\mathsf{WSS}}$ instantiated by some corrupt party $P_j \in Q_a$ within time T_{WSS} of its start. By the t_s weak commitment property of Π_{WSS}, it must thus hold that all the honest parties that compute an output in $\Pi^{(j)}_{\mathsf{WSS}}$ do so within the same time and hence have their output by time $T_{\mathsf{BC}} + T_{\mathsf{WSS}}$. Now consider the honest parties in Q_a. Since parties verify the validity of the clique Q_a, it is ensured that all the honest parties in Q_a are actually consistent with each other. Thus, each honest $P_i \in Q_a$ must hold $f_i(x)$ such that $f_i(x) = F'(x, i)$ for some (t_s, t_s) degree bivariate polynomial $F'(x, y)$. Now consider an honest party $P_i \notin Q_a$. Given that $|Q_a| \geq n - t_a$, we have that there are at least $n - t_a - t_s \geq t_s + 1$ honest parties in Q_a. Hence, it is guaranteed that an honest $P_i \notin Q_a$ will compute an output in Π_{WSS} instances of at least $t_s + 1$ parties from Q_a. Consequently, P_i can reconstruct its $f_i(x)$ consistent with the polynomial $F'(x, y)$ defined by the shares of the honest parties in Q_a. Finally, in case P_i has computed its output in $\Pi^{(j)}_{\mathsf{WSS}}$ for some corrupt party $P_j \in Q_a$, then by the same argument as in the case of t_s correctness, we have that the output f_{ji} computed by P_i is indeed the same as $F'(i, j)$. This holds since the corrupt $P_j \in Q_a$ is consistent with at least $t_s + 1$ honest parties, thus ensuring that $f_j(x)$ shared by P_j in its Π_{WSS} instance is actually $F'(x, j)$. Hence, we have that an honest $P_i \notin Q_a$ successfully reconstructs its $f_i(x) = F'(x, i)$ within time T_{VSS} ensuring t_s strong commitment.

ii. **P_k computes its output via obtaining (async, A, Q_a) at time T in some iter-**

ation of the protocol. We first note that in this case, $T > T_{\mathsf{VSS}}$ since the parties did not output via (sync, G, Q_a) in any of the $(t_s - t_a)$ iterations of the protocol. Since P_k computes its output at time T, it implies that it received (async, A, Q_a) and verified the broadcasts of all the parties in Q_a by time T. Given that the network is synchronous, by the t_s fallback consistency property of broadcast, we have that all the honest parties will receive (async, A, Q_a) as well as the corresponding broadcasts by time at most $T + 2\Delta$. Thus, an honest party $P_i \in Q_a$ will output $f_i(x)$ by time $T+2\Delta$. Since $|Q_a| - t_s \geq t_s + 1$, we have that the univariate polynomial shares of all the honest parties in Q_a indeed define a (t_s, t_s) degree bivariate polynomial $F'(x, y)$ such that $f_i(x) = F'(x, i)$ holds for each $P_i \in Q_a$. Further, since Q_a is verified to be a clique by some honest party at time T, it implies that $\Pi^{(i)}_{\mathsf{WSS}}$ instance of each honest $P_i \in Q_a$ terminated before time T. By the t_s correctness property of Π_{WSS} in the synchronous network, we have that all the honest parties compute their output at the same time and hence would have computed their output in $\Pi^{(i)}_{\mathsf{WSS}}$ before time T. This further implies that every honest $P_j \notin Q_a$ must have computed its output in at least $t_s + 1$ instances of Π_{WSS} corresponding to the honest parties in the clique, and hence will compute its output by time $T + 2\Delta$ in the worst case upon receiving (async, A, Q_a) and validating it. Moreover, the correctness of the polynomial $f_j(x)$ interpolated by an honest $P_j \notin Q_a$ can be established as in the earlier cases. We avoid repeating the argument since it's identical to the prior cases.

2. Asynchronous Network: We now prove the properties of Π_{VSS} in the asynchronous network.
   (a) t_a correctness: Let the dealer be honest. Given that the network is asynchronous, we have that the adversary can corrupt at most t_a of the parties. Given this, we have that eventually, each honest P_i will successfully compute the output in $\Pi^{(j)}_{\mathsf{WSS}}$ corresponding to every honest P_j and broadcast AOK_j. Thus, it is guaranteed that the dealer will eventually identify a clique Q_a of size at least $n - t_a$ consisting of all the honest parties and broadcast it. Thus, if the parties do not compute their output via (sync, G, Q_a), then we have that they will eventually receive (async, A, Q_a) and compute their output. For every honest $P_i \in Q_a$, we have that it will output $f_i(x)$ received from the dealer. On the other hand, an honest $P_i \notin Q_a$ will eventually compute its output in $\Pi^{(j)}_{\mathsf{WSS}}$ corresponding to at least $t_s + 1$ honest parties in Q_a and hence compute its output as in the prior cases. If $P_i \notin Q_a$ computes f_{ji} as output in $\Pi^{(j)}_{\mathsf{WSS}}$ corresponding to some corrupt $P_j \in Q_a$, then by the same argument as the synchronous case, $f_{ji} = f_i(j)$ must hold where $f_i(x) = F(x, i)$ corresponding to the (t_s, t_s) degree bivariate polynomial held by the dealer.
   In the case that some honest party computes its output via (sync, G, Q_a), then it must hold that some honest party participated with input 1 in Π_{BA} instance. Otherwise, all the honest parties would have input 0 and the validity of Π_{BA} would ensure that parties received 0 and do not compute their output via (sync, G, Q_a) which is a contradiction. Thus, we have that some honest party input 1 to Π_{BA}. By the consistency of Π_{BA}, we first have that all the honest parties will output 1 and compute their output via (sync, G, Q_a). Moreover, the party which participated with 1 would have verified the validity of Q_a before accepting it. Thus, all the honest parties will eventually validate Q_a, accept it and compute their output as described in the prior cases.
   (b) t_s privacy: The argument for t_s privacy is exactly as in the case of the synchronous

network, hence we avoid repetition.

(c) t_a strong commitment: Let the dealer be corrupt. Since the network is asynchronous, we have that the dealer can corrupt at most t_a parties. Note that t_s strong commitment was already achieved by the weaker variant of Π_{WSS}. This property follows very closely to Π_{VSS}. Strong commitment holds trivially if no honest party computes an output in a corrupt dealer's instance. Thus, we consider the case when some honest party P_h computes an output. We have two cases here: either P_h computes an output via (sync, G, Q_a) or (async, A, Q_a). In the former case, we have that some honest party participated in an instance of Π_{BA} with input 1. If not then the validity of Π_{BA} would ensure that parties output 0 and do not compute their output via (sync, G, Q_a) which is a contradiction. Thus, we have that there exists some honest party which input 1 to Π_{BA}. This honest party is guaranteed to have checked the validity of Q_a as required in the protocol at designated time steps. Hence, it must hold that Q_a is indeed a clique, which will eventually be verified by all the honest parties to compute the output. Every $P_i \in Q_a$ will thus output $f_i(x)$ such that $f_i(x) = F'(x, i)$ for some (t_s, t_s)-degree bivariate polynomial defined by the honest parties in Q_a. Further, given that $|Q_a| \geq n - t_a$, we have that the number parties in $Q_a \setminus U$ is at least $n - t_a - (t_s - t_a)$, that is $n - t_s$. Of these, we are guaranteed to have at least $n - 2t_s \geq t_s + 1$ honest parties. For every $P_i \notin Q_a$, it is thus ensured that P_i will compute its output f_{ji} in the instance $\Pi_{\mathsf{WSS}}^{(j)}$ of every honest $P_j \in Q_a$, and hence successfully reconstruct $f_i(x) = F'(x, i)$ eventually. In the latter case, it must hold that P_h did not output via (sync, G, Q_a) in any of the $(t_s - t_a)$ iterations of the protocol. Since P_h indeed computes its output upon receiving (async, A, Q_a), it must hold that P_h verified the validity of Q_a. This implies that all the honest parties will eventually receive the same and compute their output. As in Π_{WSS}, we also have that the shares of the honest parties in two different cliques Q_a and Q'_a define the same (t_s, t_s)-degree bivariate polynomial. Hence, irrespective of which $n - t_a$ sized clique an honest party accepts, it is ensured that its output will be consistent with all the honest parties.

□

8 Verifiable Triple Sharing

In this section, we give our triple sharing protocol which was discussed in Section 2.2.

Protocol 8.1: Π_{VTS}

Input: The dealer holds $2t_s + 1$ random multiplication triples denoted by $\{(a_i, b_i, c_i)\}_{i \in \{1, \ldots, 2t_s+1\}}$.
Common Input: $n + 1$ distinct elements from $\mathbb{F}$, $1, \ldots, n$ and β.
Condition: Parties continue to resolve conflicts by publicly reconstructing $X(i), Y(i), Z(i)$ for $\mathsf{NOK}(i)$ received from a party P_i until they discard the dealer or compute an output.

1. The dealer generates the degree-t_s sharings by executing Π_{VSS} to compute $([a_i], [b_i], [c_i])$ for every $i \in \{1, \ldots, 2t_s + 1\}$.
2. Upon computing the output in all the instances of Π_{VSS}, wait for the time to be a multiple of Δ. Then, for each $i \in \{1, \ldots, t_s + 1\}$, parties locally set $[x_i] = [a_i]$, $[y_i] = [b_i]$ and $[z_i] = [c_i]$.

3. Let $X(\cdot)$ and $Y(\cdot)$ be the unique polynomials of degree at most t_s defined by the points $\{(i, x_i)\}_{i \in \{1,\ldots,t_s+1\}}$ and $\{(i, y_i)\}_{i \in \{1,\ldots,t_s+1\}}$ respectively. The parties locally compute $[x_i] = [X(i)]$ and $[y_i] = [Y(i)]$, for each $i \in \{t_s+2, \ldots, 2t_s+1\}$.[5]
4. Parties invoke Π_{Beaver} with $\{[x_i], [y_i], [a_i], [b_i], [c_i]\}_{i \in \{t_s+2,\ldots,2t_s+1\}}$ and wait for time T_{Beaver}. Upon obtaining the output $\{[z_i]\}_{i \in \{t_s+2,\ldots,2t_s+1\}}$ where $z_i = x_i y_i$ for every $i \in \{t_s+2, \ldots, 2t_s+1\}$, wait for the time to be a multiple of Δ and then proceed to the next step.
5. Let $Z(\cdot)$ be the polynomial of degree at most $2t_s$ defined by the points $\{(i, z_i)\}_{i \in \{1,\ldots,2t_s+1\}}$.
6. Parties compute $\{([X(i)], [Y(i)], [Z(i)])\}$ for each $i \in \{2t_s+2, \ldots, n\}$ using $\{([X(i)], [Y(i)], [Z(i)])\}_{i \in \{1,\ldots,2t_s+1\}}$.
7. For each $P_i \in \mathcal{P}$, parties invoke Π_{privRec} 3 times with $[X(i)]$, $[Y(i)]$ and $[Z(i)]$ as input respectively to enable P_i to privately reconstruct $X(i), Y(i)$ and $Z(i)$. Each party waits for time T_{PrivRec}. Upon computing the output, wait for the time to be a multiple of Δ and proceed to the next step.
8. If $X(i) \cdot Y(i) = Z(i)$ holds, P_i broadcasts $\mathsf{OK}(i)$, and broadcasts $\mathsf{NOK}(i)$ otherwise. Parties publicly reconstruct $X(i), Y(i), Z(i)$ for each $\mathsf{NOK}(i)$ by broadcasting their shares. Each party waits for T_{BC} before proceeding to the next step.
9. The dealer constructs a set $\mathsf{OK} = \{i | \mathsf{OK}(i)$ was received from P_i's broadcast$\}$. Once $|\mathsf{OK}| \geq n - t_s$, the dealer constructs a set NOK of size $(n - t_a) - |\mathsf{OK}|$ such that $\mathsf{NOK} \subset \mathcal{P} \setminus \mathsf{OK}$ and broadcasts $(\mathsf{OK}, \mathsf{NOK})$. Parties wait for time T_{BC} before proceeding.
10. Parties publicly reconstruct $X(i), Y(i), Z(i)$ for each $i \in \mathsf{NOK}$,by broadcasting their shares. Wait for time T_{BC}. Upon receiving $X(i), Y(i), Z(i)$, verify that $X(i) \cdot Y(i) = Z(i)$ holds. If not, then discard the dealer.
11. Upon receiving $\mathsf{OK}(i)$ from each $i \in \mathsf{OK}$, completing the prior check for each $i \in \mathsf{NOK}$, and ensuring that $\mathsf{OK} \cup \mathsf{NOK} \geq n - t_a$, each party proceeds to the next step.
12. Discard the dealer if $X(i) \cdot Y(i) = Z(i)$ does not hold for some party which broadcasted $\mathsf{NOK}(i)$. If the dealer is discarded, parties output a default degree-t_s sharing of a publicly known value. Otherwise, parties locally compute and output their shares of $([X(\beta)], [Y(\beta)], [Z(\beta)])$, where $\beta \neq i$ for every $i \in \{1, \ldots, n\}$.

Theorem 8.2. *Protocol Π_{VTS} is perfectly-secure against an adversary corrupting up to t_s parties in the synchronous network and t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *t_s privacy: The view of the adversary is independent of the output triple shared on behalf of an honest dealer.*
   (b) *t_s correctness: Within time $T_{\mathsf{VTS}} = T_{\mathsf{VSS}} + T_{\mathsf{Beaver}} + T_{\mathsf{PrivRec}} + 3T_{\mathsf{BC}} = T_{\mathsf{VSS}} + 3T_{\mathsf{BC}} + 2\Delta$, the honest parties output a degree-t_s Shamir-sharing of a multiplication triple on behalf of an honest dealer.*
   (c) *t_s strong commitment: If the dealer is corrupt, then either no honest party has an output, or all the honest parties output a degree-t_s Shamir-sharing of a multiplication triple on*

[5] Computing a new point on a polynomial of degree t_s is a linear function of $t_s + 1$ given unique points on the same polynomial.

behalf of the dealer. Moreover, if some honest party computes its output at time T, then all the honest parties compute their output by time $T + 2\Delta$.

2. *Asynchronous network:*
   (a) *t_a privacy: The view of the adversary is independent of the output triple shared on behalf of an honest dealer.*
   (b) *t_a correctness: Almost-surely, the honest parties eventually output a degree-t_s Shamir-sharing of a multiplication triple on behalf of an honest dealer.*
   (c) *t_a strong commitment: If the dealer is corrupt, then either no honest party has an output, or all the honest parties eventually output a degree-t_s Shamir-sharing of a multiplication triple on behalf of the dealer.*

Proof. We first prove the properties of Π_{VTS} in the synchronous network, followed by the proof for the asynchronous network.

1. Synchronous network:
   (a) t_s privacy: In a synchronous network, for an honest dealer, each honest party computes its shares in Π_{VSS} by time T_{VSS}. All the parties thus begin the execution of Π_{Beaver} simultaneously, and by the guarantees of Π_{Beaver}, they receive the output within time Δ, that is each honest party computes its output of Π_{Beaver} by time $T_{\mathsf{VSS}} + \Delta$. Further, by the guarantees of Π_{privRec}, we have that each honest party P_i receives its points $X(i), Y(i), Z(i)$ within time $T_{\mathsf{VSS}} + 2\Delta$ and broadcasts $\mathsf{OK}(i)$. Since the honest parties start their broadcast simultaneously, all honest parties (including the dealer) receive the $\mathsf{OK}(i)$ messages by time $T_{\mathsf{VSS}} + 2\Delta + T_{\mathsf{BC}}$. Moreover, no honest party broadcasts $\mathsf{NOK}(i)$ when the dealer is honest. Thus, the dealer constructs its set OK which includes all the honest parties, which also ensures that $|\mathsf{OK}| \geq n - t_s$. This guarantees that the set $\mathsf{NOK} \subset \mathcal{P} \setminus \mathsf{OK}$ does not include any honest party. Hence, the publicly reconstructed points $X(i), Y(i), Z(i)$ for each $i \in \mathsf{NOK}$ correspond to points held by the corrupt parties. This implies that an adversary knows t_s points on each polynomial $X(\cdot), Y(\cdot), Z(\cdot)$ which are of degree $t_s, t_s, 2t_s$ respectively, thus ensuring one degree of freedom. Hence, we have that for every candidate output triple $(X(\beta), Y(\beta), Z(\beta))$, we have a corresponding input triple (a_k, b_k, c_k) for some $k \in \{1, \ldots, m\}$ unknown to the adversary that is consistent with the adversary's view.
   (b) t_s correctness: Let the dealer be honest. Note that all the honest parties obtain the output of Π_{VSS} instantiated by an honest dealer within time T_{VSS}. This further implies that Π_{Beaver} and Π_{privRec} succeed for all the honest parties by time $T_{\mathsf{VSS}} + T_{\mathsf{Beaver}} + T_{\mathsf{PrivRec}} = T_{\mathsf{VSS}} + 2\Delta$. Hence, each honest party P_i broadcasts $\mathsf{OK}(i)$, which, by the validity of broadcast in the synchronous network, is received by all the honest parties, including the dealer by time $T_{\mathsf{VSS}} + T_{\mathsf{BC}} + 2\Delta$. Hence, all the honest parties simultaneously proceed to the next step at time $T_{\mathsf{VSS}} + T_{\mathsf{BC}} + 2\Delta$. Parties additionally keep broadcasting their shares corresponding to every $\mathsf{NOK}(j)$ which is received. By the validity property of broadcast in the synchronous network, we also have that the dealer's broadcast of $(\mathsf{OK}, \mathsf{NOK})$ sets will be received by all the parties by time $T_{\mathsf{VSS}} + 2T_{\mathsf{BC}} + 2\Delta$. Finally, parties broadcast their shares corresponding to every $j \in \mathsf{NOK}$. Again, by the validity and liveness of broadcast in the synchronous network, we have that every honest party's shares will be received by all the honest parties by time $T_{\mathsf{VSS}} + 3T_{\mathsf{BC}} + 2\Delta$. Further, we have that if $\mathsf{NOK}(j)$ was broadcasted by some corrupt P_j and $X(j), Y(j), Z(j)$ is

reconstructed by this time, then it would hold that $X(j) \cdot Y(j) = Z(j)$ and hence the dealer is not discarded. Thus, we have that all the honest parties output their shares by time $T_{\mathsf{VSS}} + 3T_{\mathsf{BC}} + 2\Delta$.

(c) t_s strong commitment: If no honest party computes an output in the protocol then strong commitment holds trivially. Hence, we consider the case when there exists some honest party which computes an output. Note first that to ensure the correctness of the output, that is, to ensure that the honest parties output shares of a multiplication triple, it is required to verify that $X(\cdot) \cdot Y(\cdot) = Z(\cdot)$ holds for at least $2t_s + 1$ distinct points on these polynomials. In the protocol, this translates to ensuring that the relation holds for (at least) $2t_s + 1$ honest parties. Suppose there exists some honest P_h party that successfully outputs its shares in the protocol without discarding the dealer. For contradiction, suppose that the relation does not hold for some honest party P_i. First, observe that since P_h outputs its shares, it implies that P_h computes the outputs of all the Π_{VSS} instances initiated by the dealer. Suppose the time at which P_h computed this is T. Note that every honest party would thus have computed its output by time $T + 2\Delta$ in the worst case. Suppose the worst case, that is P_i computed its output in Π_{VSS} at time $T + 2\Delta$. It is thus possible that P_i received its $X(i), Y(i), Z(i)$ at time $T + 2\Delta + T_{\mathsf{Beaver}} + T_{\mathsf{PrivRec}} = T + 4\Delta$, whereas P_h obtained its $X(h), Y(h), Z(h)$ at time $T + T_{\mathsf{Beaver}} + T_{\mathsf{PrivRec}} = T + 2\Delta$. That is, it is possible that some honest parties broadcast their $\mathsf{OK}(i)$ or $\mathsf{NOK}(i)$ after a 2Δ delay compared to other honest parties. Specifically, P_h may have broadcast $\mathsf{OK}(h)$ and proceed to the next step by time $T + T_{\mathsf{BC}} + 2\Delta$. Moreover, P_h's broadcast would have been received by all within this time. In contrast, P_i's broadcast may be delivered to parties (including P_h) by time $T + T_{\mathsf{BC}} + 4\Delta$. This implies that $X(i), Y(i), Z(i)$ would have been reconstructed by time $T + 2T_{\mathsf{BC}} + 4\Delta$. However, the earliest P_h can compute its output is at time $T + 3T_{\mathsf{BC}} + 2\Delta$. This is because P_h waits for T_{BC} time for the dealer's broadcast of $(\mathsf{OK}, \mathsf{NOK})$. It waits for another T_{BC} time for ensuring reconstruction of values corresponding to parties in NOK. Since we have that $T + 3T_{\mathsf{BC}} + 2\Delta > T + 2T_{\mathsf{BC}} + 4\Delta$, P_h must have received $X(i), Y(i), Z(i)$ before it proceeded to compute its output. If indeed $X(i) \cdot Y(i) = Z(i)$ did not hold, then P_h would have discarded the dealer, which is a contradiction. Thus, it must be that $X(i) \cdot Y(i) = Z(i)$. This also implies that every honest party P_i's $\mathsf{NOK}(i)$ would have been received by time at most $T + 2T_{\mathsf{BC}} + 4\Delta$ and verified by P_h. Since P_h did not discard the dealer, $X(i) \cdot Y(i) = Z(i)$ must hold for every honest P_i. Given that the number of honest parties is at least $2t_s + 1$, $X(\cdot) \cdot Y(\cdot) = Z(\cdot)$ must hold. Consequently, no honest party will discard the dealer and hence all output their shares on the dealer's polynomials. Moreover, if some honest party computes its output by time T', we have that it received all the corresponding broadcasts by time T'. By the fallback validity of broadcast, all the honest parties receive the necessary broadcasts by time $T' + 2\Delta$ and subsequently compute the output.

2. Asynchronous network:

   (a) t_a privacy: In the asynchronous network, for an honest dealer, each honest party computes its shares in Π_{VSS} eventually. This ensures that all the parties eventually begin the execution of Π_{Beaver} and receive their output. Further, this also guarantees that parties invoke Π_{privRec}, and each honest party P_i receives its points $X(i), Y(i), Z(i)$ eventually and broadcasts $\mathsf{OK}(i)$. Thus we have that even if the corrupt parties are silent, an honest dealer can compute the set OK of size (at least) $n - t_s$ eventually. In the worst case, the set NOK broadcasted by the dealer may be of size (at most) $(t_s - t_a)$ and moreover may comprise completely of honest parties. Note that the points $X(i), Y(i), Z(i)$ of each $i \in \mathsf{NOK}$ are revealed publicly. This causes the adversary to learn $t_s - t_a$ points on each of $X(\cdot), Y(\cdot), Z(\cdot)$ corresponding to the $t_s - t_a$ honest parties included in NOK, in addition to the t_a points of the corrupt parties. The adversary thus learns t_s points on each of $X(\cdot), Y(\cdot), Z(\cdot)$, which is exactly the information available to the adversary in the synchronous setting. By the same argument as privacy in the synchronous setting, we have that the adversary's view is independent of the output multiplication triple.

(b) t_a correctness: Let the dealer be honest. By the t_a correctness of Π_{VSS} in the asynchronous network, we have that the honest parties will eventually compute their output. Similarly, by the t_a correctness of Π_{Beaver} and Π_{privRec}, we also have that each honest P_i eventually receives its $X(i), Y(i), Z(i)$ and consequently broadcasts $\mathsf{OK}(i)$. Since we have $n - t_a$ honest parties, it must hold that the dealer will indeed be able to construct the set OK consisting of at least $n - t_s$ parties. Again, due to the validity property of broadcast in the asynchronous network, it is ensured that the parties will receive the dealer's broadcast of $(\mathsf{OK}, \mathsf{NOK})$, and consequently reconstruct the values $X(j), Y(j), Z(j)$. These reconstructed values are guaranteed to be correct. Moreover, every $\mathsf{NOK}(j)$ broadcasted by a corrupt P_j will eventually be received by all the honest parties due to the consistency property of broadcast. Hence, we have that parties will eventually reconstruct $X(j), Y(j), Z(j)$ for each P_j and verify its correctness. Thus, all the honest parties will eventually output shares on the polynomials shared by the dealer as the shares of its multiplication triple.

(c) t_a strong commitment: This follows similarly to the synchronous case due to the t_a strong commitment property of Π_{VSS} and consistency of Π_{BC} in the asynchronous network. Specifically, if no honest party computes its output, then commitment holds trivially. On the other hand, if any honest party computes its output in Π_{VSS}, then by the t_s strong commitment property, we have that all the honest parties compute their output. Similarly, by the t_a correctness property of Π_{privRec} and Π_{Beaver}, it is ensured that parties eventually compute their point on $X(\cdot), Y(\cdot), Z(\cdot)$. Now observe that every honest party computes its output only upon verifying that the multiplicative relation holds true for at least $n - t_a$ parties. Since all the communication occurs via broadcast for the verification, the consistency property of broadcast ensures that if some honest party computes its output then eventually all the honest parties compute their output. Further, given that the adversary can corrupt at most t_a parties in the asynchronous network, this ensures that the multiplicative relation of $X(\cdot), Y(\cdot), Z(\cdot)$ is verified for at least $n - 2t_a \geq 2t_s + 1$ honest parties. As mentioned earlier, since $X(\cdot), Y(\cdot), Z(\cdot)$ are degree $t_s, t_s, 2t_s$ polynomials respectively, this verification ensures the correctness of the multiplication triples shared by a corrupt dealer.

$\square$

9 Preprocessing Phase

9.1 Private Reconstruction Protocol

We now describe the reconstruction of a degree-t_s shared value $[v]$ to a particular party P^*. For this, all the parties reveal their shares of $[v]$ to P^*, who tries to recover the secret as follows. P^* waits for Δ time to receive the shares from other parties. P^* waits for $2t_s + 1$ shares, all of which lie on a degree-t_s polynomial. If such a polynomial is reconstructed, it is guaranteed to be correct since it agrees with the shares of at least $t_s + 1$ honest parties. Recovering such a polynomial requires P^* to apply error correction repeatedly in an "online" manner to recover the secret in the case of an asynchronous network. Whereas, in the synchronous network case, it is guaranteed that all the honest parties will send their shares lying on the same polynomial within Δ time, and hence, the reconstruction will succeed. Reconstruction towards all can be performed similarly with n instances of the protocol, one towards each party. Alternatively, parties can also broadcast their respective shares to reconstruct a value publicly.

Protocol 9.1: Π_{privRec}

Input: Parties hold the degree-t_s Shamir-sharing of a value $[v]$.
Common Input: Description of a field $\mathbb{F}$, n non-zero distinct field elements $1, \ldots, n$ and the identity of a party P^*.

1. Each P_i sends its share $[v]_i$ to P^*.
2. P^* waits for Δ time and then applies online error correction on the received shares as follows. For each $r = 0, \ldots, t_s$:
   (a) Upon receiving $n - t_s \geq 2t_s + 1$ values, P^* looks for a codeword of a polynomial of degree-t_s with a distance of at most r from the values it received. If there is no such codeword, then P^* proceeds to the next iteration. Otherwise, P^* sets $p_r(x)$ as the unique Reed-Solomon reconstruction.
   (b) If $p_r(j) = [v]_j$ holds for at least $2t_s + 1$ parties, P^* computes $v = p_r(0)$. Otherwise, it proceeds to the next iteration.

Theorem 9.2. *Protocol Π_{privRec} is secure against an adversary corrupting up to t_s parties in the synchronous network and t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *t_s correctness: Within time Δ, each honest party outputs v.*
2. *Asynchronous network:*
   (a) *t_a correctness: Each honest party eventually outputs v.*

Proof. We prove the properties of Π_{privRec} in the synchronous network and subsequently in the asynchronous network.

1. Synchronous network:
   (a) t_s correctness: In a synchronous network, each honest party receives shares on the degree-t_s polynomial from every other honest party within time Δ. Thus, an honest party receives at least $n - t_s \geq 2t_s + 1$ correct points on the polynomial. Moreover, if

an honest party receives r incorrect shares by time Δ, then by the guarantees of Reed-Solomon codes and given that $r \leq t_s$, an honest party having (at least) $2t_s+1+r$ points can correct r points and recover the correct polynomial.

2. Asynchronous network:
   (a) t_a correctness: In the asynchronous network, note that each honest party will eventually receive $n-t_a \geq 2t_s+t_a+1$ correct points from the honest parties. By reasoning similar to that in the synchronous setting, the honest parties will eventually compute the correct polynomial defined by the honest parties' shares.

□

9.2 Beaver's Multiplication Protocol

This protocol uses the well-known Beaver's circuit randomization [7] technique to perform the multiplication of two shared values. Specifically, given a pre-shared random and private multiplication triple $([a],[b],[c])$, this technique reduces the computation of $[z]=[xy]$ from $[x]$ and $[y]$ to two public reconstructions. Towards this, parties first locally compute $[d]=[x]-[a]$ and $[e]=[y]-[b]$, followed by public reconstruction of d and e. Now, parties can compute $[z]$ locally using these values together with the shared multiplication triple. More precisely, since $z=xy=((x-a)+a)((y-b)+b)=(d+a)(e+b)=de+db+ea+ab=de+db+ea+c$ parties can compute $[z]=[xy]=de+d[b]+e[a]+[c]$. The formal description of the protocol appears below.

Protocol 9.3: Π_{Beaver}

Input: Parties hold the degree-t_s Shamir-sharing of a triple $([a],[b],[c])$ and the inputs $[x]$ and $[y]$.

1. Parties locally compute $[d]=[x]-[a]$ and $[e]=[y]-[b]$.
2. Parties execute $2n$ instances of Π_{privRec}, two towards every party for reconstructing d and e respectively. Wait for time Δ.
3. Parties locally compute $[z]=de+d[b]+e[a]+[c]$.

Theorem 9.4. *Protocol Π_{Beaver} is secure against an adversary corrupting up to t_s parties in the synchronous network and t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *Liveness: At time Δ, every honest party has an output.*
   (b) *t_s privacy: If (a,b,c) is a random multiplication triple from the adversary's view, then the view of the adversary is independent of x and y (and thus z).*
   (c) *t_s correctness: Within time Δ, the honest parties output a degree-t_s Shamir-sharing of z such that $z=xy$ if and only if (a,b,c) is a correct multiplication triple, i.e. $c=ab$ holds.*
2. *Asynchronous network:*
   (a) *Liveness: Every honest party eventually has an output.*
   (b) *t_s privacy: If (a,b,c) is a random multiplication triple from the adversary's view, then the view of the adversary is independent of x and y (and thus z).*

(c) t_a *correctness: The honest parties eventually output a degree-t_s Shamir-sharing of z such that $z = xy$ if and only if (a, b, c) is a correct multiplication triple, i.e. $c = ab$ holds.*

Proof. We prove the properties of Π_{Beaver} in both networks simultaneously.

1. Liveness: By the linearity property of Shamir-sharing, parties can locally compute the degree-t_s Shamir-sharing of d and e. Further, due to t_s (resp. t_a) correctness of Π_{privRec} in the synchronous (resp. asynchronous) network, we have that parties will receive the reconstructed values d and e within time Δ (resp. eventually). The computation of $[z]$ is local thereafter; hence, we have the required liveness guarantees.
2. t_s (resp. t_a) privacy: If (a, b, c) is a random multiplication triple from the adversary's view, then for every possible x and y values, there exist a and b such that they are consistent with the adversary's view and the publicly reconstructed values of d and e. Thus, the adversary's view is independent of x and y (hence z).
3. t_s (resp. t_a) correctness: Note that $z = de+db+ea+c = (x-a)(y-b)+(x-a)b+(y-b)a+c = xy + c - ab$. Hence, by inspection, it is clear that $z = xy$ if and only if $c - ab = 0$, that is, $c = ab$.

□

9.3 Triple Extraction Protocol

The last component of the Beaver triple generation phase of our protocol is a triple extraction protocol that consumes one (verified) multiplication triple, say $([a_i], [b_i], [c_i])$, shared by each party $P_i \in \mathsf{Com}$ in the prior stage and extracts $h + 1 - t$ random triples not known to any party, where $h = \lfloor \frac{|\mathsf{Com}|-1}{2} \rfloor$. For simplicity, let $m = |\mathsf{Com}|$ and without loss of generality, we assume $\mathsf{Com} = \{P_1, \ldots, P_m\}$. At a high level, the protocol proceeds as follows. First, the parties "transform" the m random shared triples $([a_i], [b_i], [c_i])$ for each $i \in \{1, \ldots, m\}$ into m correlated triples $([x_i], [y_i], [z_i])$ for every $i \in \{1, \ldots, m\}$ such that the values $\{x_i, y_i, z_i\}_{i \in \{1,\ldots,m\}}$ lie on the polynomials $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ of degree h, h and $2h$ respectively where $X(\cdot) \cdot Y(\cdot) = Z(\cdot)$. Specifically, for each $i \in \{1, \ldots, m\}$, it holds that $X(i) = x_i, Y(i) = y_i$ and $Z(i) = z_i$ where $1, \ldots, m$ are publicly known distinct elements from $\mathbb{F}$. Furthermore, the transformation ensures that the adversary knows $\{x_i, y_i, z_i\}$ only if P_i is corrupt. This implies that the adversary may know (at most) t points on each of the polynomials $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ of degree h, h and $2h$ respectively, thus guaranteeing a degree of freedom of $h + 1 - t$ in $X(\cdot), Y(\cdot)$ (and thus $Z(\cdot)$). Parties thus output the shared evaluation of these polynomials at $h + 1 - t$ publicly known points $\beta_1, \ldots, \beta_{h+1-t}$ as the extracted shared multiplication triples.

The transformation itself works as follows. The parties simply set $x_i = a_i, y_i = b_i, z_i = c_i$ for $i \in \{1, \ldots, h+1\}$. Next, $[x_i]$ and $[y_i]$ for every $i \in \{h+2, \ldots, m\}$ can be computed *non-interactively* by taking linear combination of $\{x_i, y_i\}_{i \in \{1,\ldots,h+1\}}$. Following this, $[z_i]$ for every $i \in \{h+2, \ldots, m\}$ is computed using Beaver's trick where the inputs are $[x_i]$ and $[y_i]$ and the random multiplication triple consumed is $([a_i], [b_i], [c_i])$. Clearly, if P_i is corrupt, then x_i, y_i, z_i is known to the adversary as claimed. Finally, we note that triple extraction reduces to running a batch of $\mathcal{O}(h)$ Beaver multiplications. The formal description appears in Protocol 9.5.

Protocol 9.5: Triple Extraction $- \Pi_{\mathsf{tripleExt}}$

Common input: The description of a field $\mathbb{F}$, a set $\mathsf{Com} \subseteq \mathcal{P}$ such that $m = |\mathsf{Com}| \geq n - t_s$, $m =$

$2h+1$ non-zero distinct elements $1, \ldots, m$ and $h+1-t_s$ non-zero distinct elements $\beta_1, \ldots, \beta_{h+1-t_s}$. Without loss of generality, assume $\mathsf{Com} = \{P_1, \ldots, P_m\}$.

Input: Parties hold the degree-t_s shared triples $([a_i], [b_i], [c_i])$ for every $i \in \{1, \ldots, m\}$ such that (a_i, b_i, c_i) is known to party P_i.

1. For each $i \in \{1, \ldots, h+1\}$, parties locally set $[x_i] = [a_i]$, $[y_i] = [b_i]$ and $[z_i] = [c_i]$.
2. Let $X(\cdot)$ and $Y(\cdot)$ be the degree-h polynomials defined by the points $\{x_i\}_{i \in \{1,\ldots,h+1\}}$ and $\{y_i\}_{i \in \{1,\ldots,h+1\}}$ respectively such that $X(i) = x_i$ and $Y(i) = y_i$ for all $i \in \{1, \ldots, h+1\}$.
3. For each $i \in \{h+2, \ldots, m\}$, parties locally compute $[x_i] = [X(i)]$ and $[y_i] = [Y(i)]$.
4. Parties invoke Π_{Beaver} with $\{[x_i], [y_i], [a_i], [b_i], [c_i]\}_{i \in \{h+2,\ldots,m\}}$ and obtain $\{[z_i]\}_{i \in \{h+2,\ldots,m\}}$ where $z_i = x_i y_i$ for every $i \in \{h+2, \ldots, m\}$. Wait for time Δ.
5. Let $Z(\cdot)$ be the degree-$2h$ polynomial defined by the points $\{z_i\}_{i \in \{1,\ldots,m\}}$ such that $Z(i) = z_i$ for all $i \in \{1, \ldots, m\}$.
6. Parties locally compute $[\mathbf{a}_i] = [X(\beta_i)]$, $[\mathbf{b}_i] = [Y(\beta_i)]$ and $[\mathbf{c}_i] = [Z(\beta_i)]$ for every $i \in \{1, \ldots, h+1-t_s\}$.

Theorem 9.6. *Protocol $\Pi_{\mathsf{tripleExt}}$ is secure against an adversary corrupting up to t_s parties in the synchronous network and up to t_a parties in the asynchronous network and has the following properties.*

1. *Synchronous network:*
   (a) *t_s privacy: The triples $\{(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)\}_{i \in \{1,\ldots,h+1-t_s\}}$ are random from the adversary's view.*
   (b) *t_s correctness: Within time $T_{\mathsf{tripleExt}} = \Delta$, the honest parties output a degree-t_s Shamir-sharing of of each triple $\{(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)\}_{i \in \{1,\ldots,h+1-t_s\}}$.*
2. *Asynchronous network:*
   (a) *t_s privacy: The triples $\{(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)\}_{i \in \{1,\ldots,h+1-t_s\}}$ are random from the adversary's view.*
   (b) *t_a correctness: The honest parties eventually output a degree-t_s Shamir-sharing of each triple $\{(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)\}_{i \in \{1,\ldots,h+1-t_s\}}$.*

Proof. We prove the properties of $\Pi_{\mathsf{tripleExt}}$ in both networks simultaneously.

1. t_s privacy: Note that (since $t_a < t_s$) in the worst case, there will be at most t_s corrupt parties in the set Com. This implies that at most t_s points are known to the adversary on each of the polynomials $X(\cdot)$ and $Y(\cdot)$. This ensures a degree of freedom of $h+1-t_s$ on each of these polynomials (and hence on $Z(\cdot)$). Hence, we have that for every candidate set of triples $\{(\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i)\}_{i \in \{1,\ldots,h+1-t_s\}}$, there exists a set of $h+1-t_s$ corresponding candidate input triples $([a_j], [b_j], [c_j])$ unknown to the adversary that is consistent with the adversary's view.
2. t_s (resp. t_a) correctness: By the properties of Π_{Beaver} in the synchronous (resp. asynchronous) network, we have that parties will receive obtain $\{[z_i]\}_{i \in \{h+2,\ldots,m\}}$ within time Δ (resp. eventually). Since all the input triples are guaranteed to be valid multiplication triples, by construction of the protocol, it holds that $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$ such that $X(\cdot), Y(\cdot)$ are degree-h polynomials and $Z(\cdot)$ is a degree-$2h$ polynomial. It thus follows that all the honest parties output $([\mathbf{a}_i], [\mathbf{b}_i], [\mathbf{c}_i]) = ([X(\beta_i)], [Y(\beta_i)], [Z(\beta_i)])$ for every $i \in \{1, \ldots, h+1-t_s\}$ within time Δ (resp. eventually). Moreover, the relation $\mathbf{c}_i = \mathbf{a}_i \cdot \mathbf{b}_i$ holds for every $i \in \{1, \ldots, h+1-t_s\}$ since $Z(\cdot) = X(\cdot) \cdot Y(\cdot)$ holds.

□

10 The Complete MPC Protocol

This section describes our complete MPC protocol as a composition of the primitives described so far and the existing primitives detailed in Section 4. It has the following well-known two-phase structure: a preprocessing phase wherein parties generate random Beaver triples and an online phase wherein parties consume these triples to evaluate the circuit. We elaborate on these two phases below.

Beaver Triple Generation. In this phase, the goal is to generate degree-t_s shares of random multiplication triples of the form (a, b, c) where $c = a \cdot b$. We require C random triples to be shared to evaluate a circuit with C multiplication gates. This phase can be further viewed as consisting of three stages:

1. Triples with a dealer: In this stage, each party P_i acts as a dealer and shares triples of the form (a_i, b_i, c_i) such that $c_i = a_i \cdot b_i$ must hold. The dealer is required to provide a perfect zero-knowledge proof to establish the correctness of its triples. Our main contribution lies in this stage, where the sharing of triples is performed using the verifiable triple sharing protocol (Π_{VSS}) discussed in Section 7. Further, we also give a protocol for verifiable triple sharing (Π_{VTS}), which allows the dealer to prove that the triples it shared are indeed correct. If a dealer's sharing fails, then its triples are ignored by all the parties. This protocol appears in Section 8.
2. Agreement on a Common Set (ACS): Irrespective of the network type, we have that the triple sharing instances of the honest dealers will eventually terminate. However, the instances corresponding to t_s corrupt dealers in the synchronous network, and analogously t_a corrupt dealers in the asynchronous network may never terminate. Further, parties are unaware of the underlying network condition, and in the worst case, t_s corrupt parties may not even initiate their triple sharing. To prevent endless waiting, parties proceed upon successful completion of (at least) $n - t_s$ instances of triple sharing. Since parties may receive messages in different order, we need to ensure that all the parties agree on the set of parties for whom triple sharing is successfully completed. This task is handled by the ACS protocol, Π_{ACS}, described in Section 4.
3. Triples without a dealer: Once a common set of parties Com whose triple sharing has terminated successfully been determined, the goal is to then extract random triples unknown to any party. For this, we use the existing triple sharing protocol, $\Pi_{\mathsf{tripleExt}}$, which consumes the triples shared by each party in Com and extracts random triples.

Circuit Evaluation. This is the second phase of our MPC protocol, which at a high level, consists of four stages. At the input sharing stage, parties share their inputs to the circuit. Following this, parties run Π_{ACS} to agree on a common set of at least $n - t_s$ parties that have provided input to the MPC protocol. The second stage comprises of the shared evaluation of the circuit. Since our sharing is linear, addition and multiplication by a constant operations can be performed locally. For multiplication, we rely on the well-known technique of Beaver's circuit randomisation [7]. Here, parties use the triples generated in the prior phase to evaluate multiplication gates in the circuit using Beaver multiplication. In this protocol, by using a pre-shared triple $([a], [b], [c])$, the task of computing a degree-t_s sharing $[xy]$ from $[x]$ and $[y]$ reduces to two public reconstructions. The protocol description for Beaver's multiplication protocol, Π_{Beaver}, appears in Section 4. The third stage corresponds to the reconstructing the output of the circuit to the parties. Finally, the last stage ensures that sufficiently many parties have obtained the same output. If this holds, then parties safely terminate with the output, in the MPC protocol as well as all the underlying protocols. This concludes our MPC protocol, which appears below. In the protocol description, we perform input sharing along with triple sharing in the first phase and invoke Π_{ACS} to decide on a common set of parties that successfully share both. Hence, we avoid an additional invocation of Π_{ACS} during the circuit evaluation phase.

Protocol 10.1: Network-Agnostic MPC – $\Pi^{\mathsf{na}}_{\mathsf{MPC}}$

Common input: The description of a circuit, the field $\mathbb{F}$, n non-zero distinct elements $1, \ldots, n$ and a parameter h where $n - t_s = 2h + 1$. Let $m = \lceil \frac{C}{h+1-t_s} \rceil$.
Input: Parties hold their inputs (belonging to $\mathbb{F} \cup \{\perp\}$) to the circuit.
(Beaver triple generation and Input sharing:)

1. **(Beaver Triple generation with a dealer)** Each P_i chooses m random multiplication triples and executes m instances of Π_{VTS} (Protocol 8.1, Section 8) simultaneously.
2. **(Input sharing)** Each party P_i holding k_i inputs to the circuit executes k_i instances of Π_{VSS} simultaneously (Protocol 7.1, Section 7). Parties wait for time T_{VTS}.
3. **(Input to ACS)** Each P_i initialises a set $S_i \leftarrow \phi$. It includes j in S_i if it receives an output in all the Π_{VSS} and Π_{VTS} instances of P_j.
4. **(ACS Execution)** Parties invoke Π_{ACS} (Protocol 4.9, Section 4) to agree on a set Com of at least $n - t_s$ parties whose instances of triple sharing and input sharing will terminate eventually for all the honest parties. Let $(\left[a_i^j\right], \left[b_i^j\right], \left[c_i^j\right])$ for $j \in [m]$ denote the triples shared by $P_i \in \mathsf{Com}$. The input sharing for the parties outside Com is taken as default sharing of 0. Parties wait for time T_{ACS}.
5. **(Beaver Triple Extraction)** Upon receiving output from Π_{ACS}, parties execute m instances of $\Pi_{\mathsf{tripleExt}}$ (Protocol 9.5, Section 4) with Com as the common input and additionally $(\left[a_i^j\right], \left[b_i^j\right], \left[c_i^j\right])$ for every $P_i \in \mathsf{Com}$ as the input for the j^{th} instance. Let $([\mathbf{a}_i], [\mathbf{b}_i], [\mathbf{c}_i])$ for $i \in [C]$ denote the random multiplication triples generated. Wait for time Δ.

(Circuit evaluation:)

1. **(Linear Gates)** Parties locally apply the linear operation on their respective shares of the inputs.
2. **(Multiplication Gates)** Let $([\mathbf{a}_i], [\mathbf{b}_i], [\mathbf{c}_i])$ be the multiplication triple associated with the i^{th} multiplication gate with shared inputs $([x_i], [y_i])$. Parties invoke Π_{Beaver} (Protocol 9.3, Section 4) with $\{[x_i], [y_i], [\mathbf{a}_i], [\mathbf{b}_i], [\mathbf{c}_i]\}$ for all gates i at the same layer of the circuit and obtain the corresponding $[z_i]$ as the output sharing for every gate i. Wait for time Δ.
3. **(Output)** For an output gate y with the associated sharing $[y]$, upon computing the share of y, parties execute Π_{privRec} (Protocol 9.1, Section 4) towards every party P_i. Wait for time Δ.
4. **(Termination:)** Each party P_i does the following:
   - If y has been computed during the output step, then send (ready, y) to all the parties.

- If (ready, y) has been received from at least $t_s + 1$ distinct parties, then send (ready, y) to all the parties, if not sent before.
- If (ready, y) has been received from at least $2t_s + 1$ distinct parties, then output y and terminate the protocol.

Theorem 10.2. *Let n, t_s, t_a be such that $t_a < t_s$ and $n > 2t_s + \max(2t_a, t_s)$. Protocol 10.1, Π_{MPC}, is a network-agnostic MPC protocol that is perfectly-secure against an adversary corrupting up to t_s parties in a synchronous network and up to t_a parties in the asynchronous network. It has the following properties:*

- *t_s correctness: In a synchronous network, all the honest parties compute $y = f(x_1, \ldots, x_n)$, where $x_i = 0$ if $i \notin \mathsf{Com}$ such that $|\mathsf{Com}| \geq n - t_s$ and every honest party belongs to Com within time $T_{\mathsf{MPC}} = T_{\mathsf{VTS}} + T_{\mathsf{ACS}} + T_{\mathsf{tripleExt}} + D.T_{\mathsf{Beaver}} + T_{\mathsf{PrivRec}}$.*
- *t_a correctness: When the network is asynchronous, all the honest parties eventually compute $y = f(x_1, \ldots, x_n)$, where $x_i = 0$ if $i \notin \mathsf{Com}$ such that $|\mathsf{Com}| \geq n - t_s$.*
- *t_s privacy: Irrespective of the network type, the adversary's view is independent of the inputs of the honest parties in Com.*

Proof. We first consider a synchronous network with up to t_s corruptions. By the t_s correctness property of the triple sharing and verifiable secret sharing protocols, we have that the triple sharing and input sharing instances of all the honest parties will terminate within time T_{VTS}. Further, this implies that the input requirements of the protocol Π_{ACS} for synchronous network will hold true. Hence, by the t_s correctness property of Π_{ACS}, within time T_{ACS} parties will output a set Com such that $|\mathsf{Com}| \geq n - t_s$ and it includes all the honest parties. Moreover, if there is some corrupt $P_i \in \mathsf{Com}$, it implies that some honest party P_h computed the output of verifiable secret sharing and triple sharing in P_i's instances. If not, then it would mean that no honest party includes P_i in its set S_i, and hence, all the parties would input 0 for the instance Π^i_{BA}. By the validity property of Π_{BA} in the synchronous network, we have that parties output 0 in the instance Π^i_{BA}. Thus, P_i is excluded from Com, which is a contradiction. Therefore, given that some honest party computes the output in P_i's instances of verifiable secret sharing and triple sharing, by the t_s strong commitment property of both these protocols, we have that all the honest parties compute an output. Consequently, we have that parties hold shares corresponding to m multiplication triples shared by each party in Com. Subsequently, by the t_s correctness property of $\Pi_{\mathsf{tripleExt}}$, within time Δ parties will compute the shares of random triples for $h + 1 - t_s$ for each instance of $\Pi_{\mathsf{tripleExt}}$. Given that we have $m = \lceil \frac{C}{h+1-t_s} \rceil$, parties obtain the random shares for C multiplication triples. In the circuit evaluation phase, the linear gates are computed locally. Whereas for the multiplication gates, the t_s correctness property of Π_{Beaver} ensures that all the honest parties obtain the correct sharing of the output of the gates within time Δ. Finally, the t_s correctness of the reconstruction protocol Π_{privRec} in the synchronous network ensures that parties receive their output within time Δ. Thus, we have that in a synchronous network, all the honest parties will send (ready, y) messages. Since there are at least $2t_s + 1$ honest parties, termination is guaranteed. The proof for the t_s correctness in the asynchronous network follows similarly, with the modification that it now relies on the t_s correctness of all the subprotocols in the asynchronous network. For termination, note that at least $2t_s + 1$ honest parties will eventually send (ready, y) message which all the honest parties will receive. Moreover, if some honest party terminates with an output y, then it implies that it received (ready, y) from at least $t_s + 1$ honest parties. All honest parties will eventually receive these messages and send (ready, y) to all. Since there are at least $2t_s + 1$ honest parties, termination is ensured.

The t_s privacy of the MPC protocol in either of the network conditions follows from the t_s privacy of the subprotocols. Specifically, from the t_s privacy of Π_{VSS}, we have that the inputs of honest parties are random from the adversary's view. Further, from the t_s privacy of Π_{VTS}, it follows that the multiplication triples shared by each honest P_i for $i \in \mathsf{Com}$ are random from the adversary's view. Given this, the t_s privacy of $\Pi_{\mathsf{tripleExt}}$ ensures that the multiplication triples extracted from the triples of parties in Com are indeed random from the view of the adversary. Finally, the t_s privacy of Π_{Beaver} guarantees that the adversary does not learn any additional information during the evaluation of a multiplication gate. Moreover, the rest of the gates are computed non-interactively, thus ensuring t_s privacy of the MPC protocol. □

References

[1] Abraham, I., Dolev, D., Halpern, J.Y.: An almost-surely terminating polynomial protocol for asynchronous byzantine agreement with optimal resilience. In: Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing. pp. 405–414 (2008)

[2] Abraham, I., Dolev, D., Stern, G.: Revisiting asynchronous fault tolerant computation with optimal resilience. In: Proceedings of the 39th Symposium on Principles of Distributed Computing. pp. 139–148 (2020)

[3] Appan, A., Chandramouli, A., Choudhury, A.: Perfectly-secure synchronous mpc with asynchronous fallback guarantees. In: Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing. pp. 92–102 (2022)

[4] Appan, A., Chandramouli, A., Choudhury, A.: Network agnostic perfectly secure mpc against general adversaries. In: 37th International Symposium on Distributed Computing (DISC 2023). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2023)

[5] Appan, A., Choudhury, A.: Network agnostic mpc with statistical security. arXiv preprint arXiv:2306.01401 (2023)

[6] Bangalore, L., Choudhury, A., Patra, A.: The power of shunning: efficient asynchronous byzantine agreement revisited. Journal of the ACM (JACM) **67**(3), 1–59 (2020)

[7] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference (1991)

[8] Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: Proceedings of ACM symposium on Theory of computing (1993)

[9] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC (1988)

[10] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of Annual ACM Symposium on Theory of Computing (1988)

[11] Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing. pp. 183–192 (1994)

[12] Berman, P., Garay, J.A., Perry, K.J.: Bit optimal distributed consensus. In: Computer science (1992)

[13] Blum, E., Katz, J., Loss, J.: Synchronous consensus with optimal asynchronous fallback guarantees. In: Theory of Cryptography Conference. pp. 131–150. Springer (2019)

[14] Blum, E., Katz, J., Loss, J.: Network-agnostic state machine replication. arXiv preprint arXiv:2002.03437 (2020)

[15] Blum, E., Liu-Zhang, C.D., Loss, J.: Always have a backup plan: fully secure synchronous mpc with asynchronous fallback. In: Annual International Cryptology Conference. pp. 707–731. Springer (2020)

[16] Bracha, G.: An asynchronous [(n-1)/3]-resilient consensus protocol. In: Proceedings of the third annual ACM symposium on Principles of distributed computing. pp. 154–162 (1984)

[17] Canetti, R.: Asynchronous secure computation. Technion - Computer Science Department - Technical Report **CS0755** (1993)

[18] Canetti, R.: Studies in secure multiparty computation and applications. Ph.D. thesis, Citeseer (1996)

[19] Choudhury, A.: Protocols for Reliable and Secure Message Transmission. Ph.D. thesis, Citeseer (2010)

[20] Choudhury, A., Patra, A.: An efficient framework for unconditionally secure multiparty computation. IEEE Transactions on Information Theory (2016)

[21] Deligios, G., Hirt, M., Liu-Zhang, C.D.: Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In: Theory of Cryptography Conference. pp. 623–653. Springer (2021)

[22] Deligios, G., Liu-Zhang, C.D.: Synchronous perfectly secure message transmission with optimal asynchronous fallback guarantees. In: International Conference on Financial Cryptography and Data Security. pp. 77–93. Springer (2023)

[23] MacWilliams, F.J., Sloane, N.J.A.: The theory of error correcting codes, vol. 16. Elsevier (1977)

[24] Mostéfaoui, A., Moumen, H., Raynal, M.: Signature-free asynchronous binary byzantine consensus with t¡ n/3, o (n2) messages, and o (1) expected time. Journal of the ACM (JACM) **62**(4), 1–21 (2015)

[25] Patra, A., Choudhary, A., Rabin, T., Rangan, C.P.: The round complexity of verifiable secret sharing revisited. In: Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 487–504. Springer (2009)

[26] Patra, A., Choudhury, A., Rangan, C.P.: Communication efficient perfectly secure vss and mpc in asynchronous networks with optimal resilience. In: International Conference on Cryptology in Africa. pp. 184–202. Springer (2010)

[27] Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM (JACM) (1980)

[28] Rabin, T.: Robust sharing of secrets when the dealer is honest or cheating. Journal of the ACM (JACM) **41**(6), 1089–1109 (1994)

[29] Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Proceedings of ACM Symposium on Theory of Computing (1989)