# Perfect Asynchronous MPC with Linear Communication Overhead

Ittai Abraham* Gilad Asharov† Shravani Patil‡ Arpita Patra§

March 13, 2024

**Abstract**

We study secure multiparty computation in the asynchronous setting with perfect security and optimal resilience (less than one-fourth of the participants are malicious). It has been shown that every function can be computed in this model [Ben-OR, Canetti, and Goldreich, STOC'1993]. Despite 30 years of research, all protocols in the asynchronous setting require $\Omega(n^2C)$ communication complexity for computing a circuit with $C$ multiplication gates. In contrast, for nearly 15 years, in the synchronous setting, it has been known how to achieve $\mathcal{O}(nC)$ communication complexity (Beerliova and Hirt; TCC 2008). The techniques for achieving this result in the synchronous setting are not known to be sufficient for obtaining an analogous result in the asynchronous setting.

We close this gap between synchronous and asynchronous secure computation and show the first asynchronous protocol with $\mathcal{O}(nC)$ communication complexity for a circuit with $C$ multiplication gates. Linear overhead forms a natural barrier for general secret-sharing-based MPC protocols. Our main technical contribution is an asynchronous weak binding secret sharing that achieves rate-1 communication (i.e., $\mathcal{O}(1)$-overhead per secret). To achieve this goal, we develop new techniques for the asynchronous setting, including the use of *trivariate polynomials* (as opposed to bivariate polynomials).

*Intel Labs. ittai.abraham@intel.com

†Department of Computer Science, Bar-Ilan University, Israel. Gilad.Asharov@biu.ac.il. Sponsored by the Israel Science Foundation (grant No. 2439/20), by JPM Faculty Research Award, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

‡Indian Institute of Science, Bangalore, India. shravanip@iisc.ac.in. Supported by C3iHub IIT Kanpur 2020-2025.

§Indian Institute of Science, Bangalore, India. arpita@iisc.ac.in. Supported by C3iHub IIT Kanpur 2020-2025, Google India Faculty Award, and JPM Faculty Research Award.

# Contents

# 1 Introduction

Secure multiparty computation (MPC) protocols can be divided into two broad categories: *synchronous* and *asynchronous*, depending on their resilience to network conditions. In the *synchronous* model of MPC, the assumption is that all messages sent between honest parties arrive after some known bounded delay. This delay bound needs to be fixed in advance and must hold for the lifetime of the system. Fixing a large delay bound may cause the protocol to be inefficient and slow. More worrisome, using a delay that is smaller than the actual delay the adversary can impose may lead to non-termination. In many real world settings it is very hard to guess in advance a bound on the maximum delay the adversary can impose.

The second category of protocols is the *asynchronous* model, where each message sent between honest parties arrives after some finite delay. This model allows protocols to dynamically adjust to any adversarial network conditions, and obtain termination (with probability 1) even under very powerful adversaries that can adaptively manipulate network delays.

In this paper we consider the most demanding setting: *perfect security with optimal resilience in the asynchronous model*. From the lower bound of [11, 12, 4], perfect security implies that the number of corruptions in this setting is at most $t < n/4$, so optimal resilience is when $n = 4t + 1$ (this is in contrast to $n = 3t + 1$ in the synchronous setting). The seminal work of [11, 16] obtains perfect security with optimal resilience in the asynchronous model.

### Communication Complexity of Asynchronous MPC

The communication efficiency of MPC protocols is measured by the (amortized) cost of their communication complexity per multiplication gate. In the perfectly secure, optimally-resilient synchronous model, $\mathcal{O}(n \log n)$ communication complexity per multiplication gate was obtained nearly 15 years ago by the work of [8] (recently [2] improves the round complexity from $\mathcal{O}(D + n)$ to expected $\mathcal{O}(D)$ for circuits of depth $D$). Linear communication complexity per multiplication gate seems to be a natural barrier. While there is no lower bound, getting $o(n)$ per multiplication gate seems to require fundamentally different techniques and comes at the cost of trading off optimal threshold (e.g., see [23]).

Progress in the (perfectly secure, optimally resilient) asynchronous model over the last 30 years has been slower. The work of [11] obtained $\tilde{\mathcal{O}}(n^6)$ per gate. [30, 29] improve to $\tilde{\mathcal{O}}(n^5)$ per gate. [7] improves to $\tilde{\mathcal{O}}(n^3)$ per gate. The best current bounds are by [27, 28] that obtained $\tilde{\mathcal{O}}(n^2)$ communication complexity per multiplication gate. A natural question remained open for 30 years:

*Is there a perfectly secure, optimally-resilient asynchronous MPC with $\tilde{\mathcal{O}}(n)$ communication complexity per multiplication gate?*
*Or is there an inherent lower bound due to asynchrony?*

### Our Main Result

Our main result is a perfectly secure, optimally-resilient asynchronous MPC protocol that achieves $\boldsymbol{\tilde{\mathcal{O}}(n)}$ communication per multiplication gate.

**Theorem 1.1** (Main Result)**.** *For a circuit with $C$ multiplication gates and depth $D$ there exists a perfectly-secure, optimally-resilient asynchronous MPC protocol with $\mathcal{O}((Cn + Dn^2 + n^5) \log n)$ communication complexity and $\mathcal{O}(D)$ expected run-time.*

Previously, the best-known result required $\mathcal{O}((Cn^2+Dn^2+n^5)\log n)$ communication complexity [27].

### Main Technical Result

Our main technical result is a new Asynchronous Weak-Binding Secret Sharing that costs just $\mathcal{O}(\log n)$ bits of communication complexity per secret of size $\log n$ bits. It is perfectly secure, resilient to $t < n/4$, and has constant round complexity and polynomial computation complexity. For our MPC purposes, we do not need the reconstruction of those shares; we just need the dealer to commit to a well-defined polynomial. We call this property as "weak-binding".

**Theorem 1.2** (Asynchronous Weak-binding Secret Sharing (informal))**.** *There exists a perfectly secure, optimally resilient protocol for asynchronous weak-binding secret sharing that can share* $\mathcal{O}(n^4)$ *secrets in constant time, with communication complexity of* $\mathcal{O}(\log n)$ *bits per secret of size* $\log n$ *bits.*

Many forms of verifiable secret sharing, both in the synchronous and asynchronous settings, rely on bivariate polynomial sharing. Our protocol is based on *trivariate polynomial* sharing. Our use of trivariate polynomial sharing approach follows the recent work of Appelbaum and Kachlon [5]. Nevertheless, we show how to reconstruct the trivariate polynomial for future reference and for independent interest. This variant is "weak" in the sense that reconstruction might fail (or not terminate). In that case, however, the honest dealer can shun a set of $t/2+1$ corrupted parties. Nonetheless, we remark again that we do not use reconstruction, and in particular, our final MPC construction does not shun parties and does not use player elimination.

The first asynchronous verifiable secret sharing protocol [10] achieves $\mathcal{O}(n^4)$ (amortized) overhead per secret. In comparison, our trivariate-based asynchronous weak-binding secret sharing achieves $\mathcal{O}(1)$ (amortized) overhead per secret. Nevertheless, we remark again that our primitive is weaker as it does not guarantee reconstruction. Despite this fact, we show that this weak primitive suffices for the crux of our MPC, which is a distributed ZK proof of multiplication triplets. The aim of the ZK proof is to prove that some (secret) polynomial $p(x)$ possesses a certain degree. To accomplish this, the prover incorporates the coefficients of $p(x)$ as secrets in the trivariate polynomial and distributes shares on this trivariate polynomial. Since the trivariate polynomial can contain a predetermined number of secrets, the mere success of the sharing process and the existence of a well-defined trivariate polynomial are sufficient evidence to confirm that $p(x)$ indeed has the desired degree. Here, there is no need for reconstruction. Reconstruction would also reveal the coefficients of the polynomial $p(x)$, which have to remain secret.

## 1.1 Related work

In the setting of perfect security with a synchronous network, the work of [9, 26] achieved $\tilde{\mathcal{O}}(n)$ communication per multiplication gate. A lower bound of $\tilde{\mathcal{O}}(n)$ was later established in [25] for a resilience of $t < n/3$ which is known to be necessary in this setting. The recent work of [2] improves the round complexity of [9, 26] from $\mathcal{O}(D+n)$ to $\mathcal{O}(D)$ in expectation while maintaining linear communication complexity in the number of parties.

The results in the perfect asynchronous setting have been mentioned earlier and we avoid repetition here. We simply summarize that there is no linear overhead protocol in this setting thus far. Nonetheless, linear-overhead protocols have been achieved earlier in two weaker setting– (a)

statistical security with non-optimal resilience of $t < n/4$ over asynchronous networks[1] [22] (b) perfect security with $t < n/4$ over hybrid network where the network permits a single synchronous round before turning to fully asynchronous mode [21, 22].

As mentioned, our trivariate secret sharing protocol is inspired by the work of Applebaum and Kachlon [5]. This work uses trivariate polynomial for constructing error-correcting code with quasipolynomial-time conflict-decoder.

# 2 Technical Overview

In this section, we provide a technical overview of our work. We give some background on basic asynchronous verifiable secret sharing (basically covering previous work) in Section 2.1, and proceed to our asynchronous weak-binding secret sharing in Section 2.2. We overview our MPC protocol in Section 2.3. Lastly, we conclude our triple secret sharing protocol in Section 2.4 which acts as the building block for MPC and builds upon our asynchronous weak-binding secret sharing.

**The model.** Before we proceed, let us first introduce the model. We assume asynchronous communication, which means the adversary can arbitrarily delay messages sent between honest parties. However, such messages are eventually received. It is important to note that the adversary does not see the content of the messages (as we assume ideal channels between the honest parties), but it can see the type of messages that are being sent (e.g., identifying whether it's the first, second, or third message of the protocol). Since the adversary controls the corrupted parties, messages that are supposed to be sent by the corrupted parties to the honest parties might never be sent. Honest parties cannot distinguish whether a message is merely delayed or has not been sent altogether. Consequently, honest parties must continue waiting, with the potential consequence of certain foundational processes never reaching completion. However, it's important to highlight that the complete MPC protocol guarantees termination. This means that it possesses mechanisms to recognize non-terminating sub-protocols and take appropriate measures to bring them to a halt.

Besides the point-to-point channels, we assume for now the existence of a broadcast channel with the guarantee that (1) If the sender is honest and broadcasts $M$ then eventually all honest parties will receive $M$; (2) If some honest party received a message $M$ (in an instance of a corrupted sender), then eventually all honest parties will receive $M$. This can be implemented by asynchronous broadcast or A-cast primitive [13]. The cost is $\mathcal{O}(n^2|M|)$ for broadcasting the message $M$.

## 2.1 Basic Asynchronous Verifiable Secret Sharing

Our starting point is a variant of the verifiable secret sharing due to Ben-Or, Canetti, and Goldreich [11]. In asynchronous verifiable secret sharing (AVSS), the dealer holds some secret $s$, and its goal is to distribute the shares to the parties. The parties then verify that the shares define a unique secret. At a later point, the parties might reconstruct the secret $s$. The properties that the AVSS offers are:

- **Validity:** If the dealer is honest, then the protocol must terminate. At the end of the reconstruction phase, all honest parties output $s$, the input of the dealer in the sharing phase;
- **Secrecy:** The view of the adversary in the sharing phase in the case of an honest dealer is independent of $s$;

[1] The optimal resilience for statistical asynchronous MPC is $t < n/3$ [12].

- **Binding:** The view of the honest parties at the end of the sharing phase (if terminated) uniquely defines some secret $s'$.

For simplicity, we assume for now that the dealer can efficiently solve the problem of finding the maximal clique in a graph. At a high level, the secret-sharing protocol proceeds as follows:

1. The dealer: Choose a random bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ of degree-$t$ in both variables such that $S(0,0) = s$. Send over the private channels to each party $P_i$ its shares $(S(\mathbf{x}, i), S(i, \mathbf{y}))$.
2. Each party $P_i$: Upon receiving the shares $(f_i(\mathbf{x}), g_i(\mathbf{y}))$ from the dealer, send to $P_j$ the sub-shares $(f_i(j), g_i(j))$.
3. Each party $P_i$: Upon receiving $(u_{j,i}, v_{j,i})$ from party $P_j$, verify that $u_{j,i} = g_i(j)$ $(= f_j(i))$ and $v_{j,i} = f_i(j)$ $(= g_j(i))$. If so, then $P_i$ broadcasts $\mathsf{Good}(i, j)$.
4. The dealer: Initialize an undirected graph $G$ over the vertices $V = [n]$. Upon seeing $\mathsf{Good}(i, j)$ broadcasted by $P_i$ and $\mathsf{Good}(j, i)$ broadcasted by $P_j$, add the edge $(i, j)$ to the graph. If a clique $K \subseteq [n]$ of cardinality at least $3t + 1$ is found in $G$, then broadcast $(\mathsf{Clique}, K)$.
5. Each party $P_i$: Initialize a similar graph as the dealer in the previous step. Upon seeing a broadcasted message $(\mathsf{Clique}, K)$ from the dealer, verify that $K$ is a $3t + 1$ clique in the graph. If not, continue to listen to $\mathsf{Good}$ messages broadcast and update the graph. Once $K$ is verified:
   (a) If $i \in K$, then halt and output $(f_i(x), g_i(y))$.
   (b) If $i \notin K$, then wait to receive all sub-shares from parties in $K$ (received from Step 3), and reconstruct the polynomials $f_i(x)$, $g_i(y)$ using Reed-Solomon decoding.

We do not specify the reconstruction phase, as it is immediate and less relevant to our discussion. Moreover, note that the protocol might never terminate in a case of a corrupted dealer. E.g., the parties might wait forever for the dealer to broadcast the message $(\mathsf{Clique}, K)$. A party cannot decide whether to abort or whether this message will eventually arrive.

We first claim that if one honest party terminates, all honest parties eventually terminate. An honest party terminates only after the dealer has broadcasted a clique $K$, and it validated that the clique exists in its graph. Since those are all broadcasted messages if one honest party saw this, all honest parties would eventually see the same property.

If a clique of $3t + 1$ parties is found, the clique must contain at least $2t + 1$ honest parties. All the messages of those honest parties are broadcasted; therefore, we know that their shares agree. Their shares define a unique bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ of degree-$t$ in both variables. It is also guaranteed that all honest parties eventually output shares on the same bivariate polynomial. Specifically, when running the Reed-Solomon decoding on messages received from parties in $K$, there are $2t + 1$ shares on the polynomial $S(\mathbf{x}, \mathbf{y})$ and at most $t$ errors. Reed Solomon decoding results in shares on the bivariate polynomial. Finally, validity holds from the fact that when the dealer is honest, all honest parties agree with each other, and therefore a $3t+1$-clique must appear in the graph.

**Making it polynomial time – the STAR algorithm.** The problem with the above protocol is that the dealer has to solve clique, which is an $\mathsf{NP}$-hard problem. A beautiful idea in the work of Ben-Or, Canetti, and Goldreich [11] (credit within is given to Canetti's thesis [17]) shows that an *approximation* of clique suffices to bind a unique bivariate polynomial. Specifically, the dealer searches for a $(\mathsf{C}, \mathsf{D})$-star, which is defined as follows:

**$(\mathsf{C}, \mathsf{D})$-Star:** sets $\mathsf{C}, \mathsf{D} \subseteq [n]$ are Star in $G$ if (1): $\mathsf{C} \subseteq \mathsf{D}$; (2) $|C| \geq 2t+1$ and $|\mathsf{D}| \geq n-t$; (3) For every $c \in \mathsf{C}$ and $d \in \mathsf{D}$ it holds that $(c, d) \in G$.

Note that $\mathsf{C}$ is a clique, whereas nodes in $\mathsf{D}$ agree with all nodes in $\mathsf{C}$, but not necessarily with each other. The main idea is that if there exists a clique $K$ of size $n - t \geq 3t + 1$ in $G$, it might be hard to find it, but it is easy to find smaller cliques of size $n - 2t \geq 2t + 1$. For example, to find such a clique, look at the complement graph $\overline{G}$. The clique $K$ is now an independent set; Find a maximal matching in the graph $\overline{G}$; let $M$ be that maximal matching. The set $[n] \setminus M$ is an independent set in $\overline{G}$ and, therefore, a clique in $G$. Moreover, when the dealer is honest, since all the edges are between honest parties and corrupted parties, or between corrupted parties, then $M$ is of size at most $2t$. Therefore, $[n] \setminus M$ is of size $n-2t$ and is a clique in the graph $G$. Canetti [17] shows a procedure that, if a $3t+1$ clique exists, then it efficiently finds a $(\mathsf{C}, \mathsf{D})$-STAR in a graph – i.e., a smaller clique ($\mathsf{C}$) of size $2t + 1$, together with a larger set $\mathsf{D}$ where each $d \in \mathsf{D}$ is connected to all of $\mathsf{C}$.

The verifiable secret sharing is slightly more involved when the dealer finds a $(\mathsf{C}, \mathsf{D})$-star and not a $3t + 1$-clique. We do not get into the exact details. Yet, the main ideas why the STAR structure suffices are as follows:

- Validity: When the dealer is honest, then the dealer must eventually find a STAR. That is, when the dealer is honest, then eventually, we will have a clique $K$ of size $3t+1$ in the graph. In that case, the STAR algorithm always finds a $(\mathsf{C}, \mathsf{D})$-star.
- Binding: If a $(\mathsf{C}, \mathsf{D})$-star was found (either when the dealer is honest or corrupted), then a unique bivariate polynomial is defined from the shares of the honest parties in $\mathsf{C}$. Since the size of $\mathsf{C}$ is at least $2t+1$, it contains at least $t+1$ honest parties. The shares of those honest parties uniquely define a bivariate polynomial. Moreover, the honest parties in $\mathsf{D}$ agree with all the honest parties in $\mathsf{C}$; therefore, their shares lie on the same bivariate polynomial. At this point, we have at least $2t + 1$ honest parties that hold correct shares, and therefore all honest parties can eventually reconstruct correct shares.

**Communication complexity.** Before we proceed, let us first elaborate on the communication complexity of the protocol above. It is easy to see that the parties exchange a total of $\mathcal{O}(n^2 \log n)$ bits over the point-to-point channels and additional $\mathcal{O}(n^2 \log n)$ bits over the broadcast channel. This is translated to $\mathcal{O}(n^4 \log n)$ total communication complexity over point-to-point channels and no broadcast (using the broadcast protocol of [13]). That is, we have an overhead of $\mathcal{O}(n^4 \log n)$ for sharing just a single value!

### 2.2 Our Asynchronous Weak-Binding Secret Sharing

**Our goal: $\mathcal{O}(1)$ overhead per secret.** To achieve secure computation with linear communication complexity, our ultimate goal is to reach $\mathcal{O}(1)$ overhead per secret. This necessitates a substantial enhancement of the basic scheme by a factor of $\mathcal{O}(n^4)$. Borrowing ideas from the synchronous MPC [2], this goal is achieved via two routes: (1) batching; and (2) packing. However, packing in the asynchronous case gets an intriguing turn and requires borrowing new ideas.

**Reducing the communication complexity by batching.** It is immediate to reduce the communication complexity using a batching technique: The dealer will invoke $m$ instances of AVSS in parallel. At the same time, the broadcasted information will be shared for all $m$ instances. That is,

a party $P_i$ will broadcast $\mathsf{Good}(i,j)$ only after it receives from $P_j$ the sub-shares in all $m$ instances and verifies that they agree with the share it received from the dealer in each one of the $m$ instances. This reduces the total cost to $\mathcal{O}(m \cdot n^2 \log n)$ over point-to-point plus $\mathcal{O}(n^2 \log n)$ broadcast (notice that the broadcast cost is independent of $m$). Setting $m = n^2$, we obtain a protocol that runs in total communication complexity of $\mathcal{O}(n^4 \log n)$ bits for sharing $n^2$ secrets (each is of size $\log n$ bits). At this point, we have an overhead of $\mathcal{O}(n^2)$ per secret.

**Reducing the communication complexity by packing.** Packing in the asynchronous case turns out to be radically different than in the synchronous case. The main reason for the $\mathcal{O}(n^2)$ overhead in the secret sharing is because a single secret is hidden in a structure of size $\mathcal{O}(n^2)$, i.e., a bivariate polynomial. To achieve $\mathcal{O}(1)$ overhead, we have to pack $\mathcal{O}(n^2)$ secrets in a single bivariate polynomial or use a different structure.

In the basic AVSS protocol, a single secret is shared using a $(t,t)$-bivariate polynomial.[2] The adversary, which controls at most $t$ parties, receives $f_i(\mathbf{x}) = S(\mathbf{x}, i)$ and $g_i(\mathbf{y}) = S(i, \mathbf{y})$ for every $i \in I$, when $I$ is the set of corrupted parties, and assume for simplicity that $|I| = t$ (and not smaller). The $f$-shares give the adversary a total of $t(t+1)$ points on the polynomial. Since $f_i(k) = g_k(i)$ for every $i, k \in I$, the $g$-shares gives the adversary just additional $t(t+1) - t^2 = t$ "new" points on the polynomial (those are $g_i(0)$ for $i \in I$), and thus the adversary gets a total of $t^2 + 2t$. The polynomial $S(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{t} \sum_{j=0}^{t} a_{i,j} x^i y^j$ contains a total of $(t+1)^2$ points, and thus we have just a single degree of freedom, i.e., we can hide just a single secret.

The main idea of packing is to use a polynomial of a higher degree while maintaining the same cost for the sharing and verification:

**Using $(t + t/2) \times t$–bivariate polynomial.**[3] If we use a bivariate polynomial of degree, e.g., $(t + t/2, t)$, then there are in total $(t + t/2 + 1)(t + 1)$ values on the polynomial. The important part of this particular choice of parameters is that the degree of the $\mathbf{y}$ is $t$, which still allows using Reed-Solomon decoding as part of the protocol. Similar calculation as in the $(t,t)$ case leads to the adversary's shares revealing a total of $t(t + t/2 + 1) + t$ values. The bivariate polynomial, therefore, contains $t/2 + 1$ degrees of freedom, i.e., we can hide $\mathcal{O}(n)$ values in a single bivariate polynomial.

Packing $\mathcal{O}(n)$ values instead of $\mathcal{O}(1)$ reduced the overhead per secret from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. The important message here is that the STAR algorithm still suffices. Specifically, a $(\mathsf{C}, \mathsf{D})$-star defines a unique bivariate polynomial: Recall that $\mathsf{C}$ is a clique of $2t + 1$ parties, and therefore it must contain at least $t + 1$ honest parties with their shares agreeing with each other. Moreover, all the honest parties in $\mathsf{C}$ are consistent with all the parties in $\mathsf{D}$ where $\mathsf{D}$ contains at least $2t + 1$ honest parties. The shares $f$ shares of honest parties in $\mathsf{C}$ together with the $g$-shares of honest parties in $\mathsf{D}$ define a unique bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ that satisfies $S(\mathbf{x}, c) = f_c(\mathbf{x})$ for every $c \in \mathsf{C}$ (recall that each $f_c$ has degree $t + t/2$). Moreover $S(d, \mathbf{y}) = g_d(\mathbf{y})$ holds for every honest party $d \in \mathsf{D}$. Guaranteeing these parties will also hold correct $f$-shares requires some additional work, which was already shown in previous works (see [22]).

This protocol plays a pivotal role in our final construction, but for our ultimate goal, we still need to go one step further and push for $\mathcal{O}(1)$-word overhead per secret. Note, however, that this one step forward will not give us verifiable secret sharing, but only some weaker form of sharing.

[2] We use $(q, \ell)$-bivariate polynomial to denote a bivariate polynomial which is of degree at most $q$ in $\mathbf{x}$ and at most $\ell$ in $\mathbf{y}$.

[3] We use $t + t/2$ just as an example. The above works for any $t \times (t + d)$-bivariate polynomial for $d \leq t$ and $d \in O(n)$.

**Using $(t + t/2) \times (t + t/2)$–bivariate polynomial.** If we use a bivariate polynomial of degree $(t+t/2, t+t/2)$, there are $(t+t/2+1)^2$ total values on the polynomial. Similar calculations as above lead to $(t/2+1)^2$ values that we can pack in the polynomial, i.e., $\mathcal{O}(n^2)$ secrets.

However, here the STAR technique does not give us a binding guarantee. A $(\mathsf{C}, \mathsf{D})$-star provides a clique $\mathsf{C}$ of size $2t+1$, which contains, in the worst case, $t+1$ honest parties. Their $f$-shares, their $g$-shares, separately or combined, do not uniquely define a $(t+t/2)\times(t+t/2)$-bivariate polynomial. Since the parties in $\mathsf{D}$ do not necessarily agree with each other, we cannot use their shares to define the bivariate polynomial before we have a unique one that is defined by the parties in $\mathsf{C}$.

It is easy to see that an alternative, stronger guarantee, would suffice for binding:

> **$(\mathsf{C}, \mathsf{D})$-BigStar:** sets $\mathsf{C}, \mathsf{D} \subseteq [n]$ are BigStar in $G$ if: (1) $\mathsf{C} \subseteq \mathsf{D}$; (2) $|C| \geq 2t + t/2 + 1$ and $|\mathsf{D}| \geq n - t$; (3) For every $c \in \mathsf{C}$ and $d \in \mathsf{D}$ it holds that $(c, d) \in G$.

Note that the only difference between BigStar and Star is that $\mathsf{C}$ is of size $2t + t/2 + 1$ instead of $2t + 1$ as in Star. Such a larger set $\mathsf{C}$ ensures $t + t/2 + 1$ honest parties that agree with each other, hence defining a unique bivariate polynomial. However, we are unaware of any polynomial-time algorithm that finds BigStar in a graph. At this point, we can potentially reach the $\mathcal{O}(1)$ overhead in communication complexity, but at the expense of having the dealer find a large clique in exponential time. This is clearly not ideal.

**Exponential-time improvement using trivariate polynomials.** To solve the above problem, we add one more dimension, which will allow us, in particular, to pack $\mathcal{O}(n^3)$ secrets and find a BigStar efficiently, if it exists. Instead of using a bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ of degree $t + t/2$ in both variables $\mathbf{x}, \mathbf{y}$, the dealer will use a *trivariate* polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ of degree $t + t/2$ in all three variables $\mathbf{x}, \mathbf{y}, \mathbf{z}$. We then naturally extend the protocol to trivariate sharing. E.g., the share of each party $P_i$ is now *three* bivariate polynomials:

$$\mathbf{S}(\mathbf{x}, \mathbf{y}, i), \quad \mathbf{S}(\mathbf{x}, i, \mathbf{z}), \quad \mathbf{S}(i, \mathbf{y}, \mathbf{z}) \ .$$

Two parties, $P_i$ and $P_j$ then exchange *six* univariate polynomials:

$$\begin{aligned} &\mathbf{S}(\mathbf{x}, j, i), \quad &&\mathbf{S}(\mathbf{x}, i, j), \quad &&\mathbf{S}(i, \mathbf{y}, j) \ , \\ &\mathbf{S}(j, \mathbf{y}, i), \quad &&\mathbf{S}(j, i, \mathbf{z}), \quad &&\mathbf{S}(i, j, \mathbf{z}) \ . \end{aligned}$$

and a party $P_i$ broadcasts $\mathsf{Good}(i, j)$ if the shares it receives from $P_j$ (i.e., the six univariate polynomials) agree with those it received from the dealer.

The dealer then constructs a graph $G$ as in the basic AVSS scheme. However, as mentioned, we are now looking for a more robust condition on the graph since our polynomials are of degree $t + t/2$. Thus, we need a clique of size $2t + t/2 + 1$, which will imply having at least $t + t/2 + 1$ honest parties. BigStar now also suffices for binding, but it is unclear how to find it just as in the bivariate sharing case. However, at this point, some seemingly weaker property on the graph also suffices to achieve binding, which was insufficient in bivariate sharing. The property is:

> Dense*: A set of vertices $L \subseteq [n]$ is called* Dense *if it is of cardinality at least $3t + 1$,*[4] *and each node in $L$ has at least degree $3t + t/2 + 1$.*

[4] We note that an $L$ of size $2t+t/2+1$ would, in fact, suffice for defining a unique trivariate polynomial. However, we are able to find a larger set of size $3t + 1$ in polynomial time.

This is clearly a property that is easy to find in a graph $G$. The intuition is that the set $L$ contains at least $2t+1$ honest parties. Moreover, two honest parties $P_k, P_\ell$ that have degree $3t+t/2+1$ *must agree with each other, even though they did not necessarily hear the shares of one another* due to communication delays (we will see why this holds soon). The honest parties in Dense therefore define a clique of at least $2t+1$ honest parties, and their shares define a unique trivariate polynomial.

**Binding:** For binding, it suffices to have one of the two following properties:

1. Dense: which essentially defines a clique of $2t+1$ honest parties;
2. BigStar: where $|\mathsf{C}| \geq 2t+t/2+1$, and $|\mathsf{D}| \geq n-t$. The set C defines a clique of $t+t/2+1$ honest parties that agree with each other, which defines a unique trivariate polynomial. Moreover, the set D defines overall $2t+1$ honest parties that have correct shares.

Therefore, whenever one of those properties occurs in the graph, we are satisfied and can terminate the protocol. We show a poly-time algorithm that finds those properties.

**Validity.** For validity, we have to guarantee that when the dealer is honest, it must find one of these two properties (Dense or BigStar) in the graph. In the honest dealer case, the graph will eventually contain a clique of size $3t+1$. At this point, the following algorithm must find either Dense or BigStar:

1. The dealer looks for a Dense set $L$ in the graph. If found, output $(\mathsf{Dense}, L)$.
2. If the graph does not contain a dense set $L$, then it implies that there is at least one honest party, say $P_j$, whose degree is less than $3t+t/2+1$. Moreover, all missing edges in the graph are between honest parties and corrupted parties, or between corrupted parties. We then consider the graph $G$ while considering only the neighbors of $P_j$ (including $P_j$). We will obtain a graph of size $n' = 3t+t/2+1$ vertices, where all removed vertices correspond to corrupted parties, and so we have $t' = t/2$ corrupted parties remaining. Moreover, this graph contains a clique of size $3t+1$. The standard $(\mathsf{C}, \mathsf{D})$-star returns a clique $|\mathsf{C}| \geq n' - 2t' \geq 2t + t/2 + 1$, and a set $|\mathsf{D}| \geq n' - t' \geq 3t+1$, which is essentially a BigStar in $G$. The dealer does the above for every low-degree party $P_j$ until it hits on a BigStar, and this procedure is efficient.

Therefore, once we have a clique of $3t+1$ honest parties, the dealer is guaranteed to find either a Dense or a BigStar in polynomial time.

**Why do bivariate polynomials not suffice?** A natural question is why we could not find the same property on the graph with bivariate polynomials, and we have to work with trivariates. This is an idea we borrow from [5], and is the crux of this part of the paper. The property we need here is *transitivity*. Suppose $P_k$ and $P_j$ are both honest, and they agree with a common set $E$ of at least $2t+1$ honest parties, but they did not hear the sub-shares from each other since the adversary delays their communication. Do their shares agree?

Note that in the setting of the Dense graph, $P_k$ and $P_j$ are two honest parties that have a high degree of $3t+t/2+1$. This means they have at least $3t+1$ parties in their intersection, which implies that they agree with some common set $E$ of at least $2t+1$ honest parties. To see whether or not their shares agree, let's consider bivariate sharing versus trivariate sharing:

1. **Bivariate:** In bivariate sharing, the parties hold univariate polynomials as shares, and they exchange points. It is easy to see that the shares of $P_k$ and $P_j$ do not necessarily agree with each other, even if they agree with a common set $E$ of cardinality $2t+1$. We can set $P_k$

to have $f_k(\mathbf{x}), g_k(\mathbf{y})$ and $P_j$ to have $f_j(\mathbf{x}), g_j(\mathbf{y})$ such that $f_k(j) \neq g_j(k)$ and $f_j(k) \neq g_k(j)$. Yet, we can give a set $E$ of cardinality $2t+1$ arbitrary shares such that $f_j(e) = g_e(j)$ and $g_j(e) = f_e(j)$ for every $e \in E$, and likewise, $f_k(e) = g_e(k)$ and $g_k(e) = f_e(k)$ for every $e \in E$. This does not impose many constraints on the polynomials $f_e(\mathbf{x}), g_e(\mathbf{y})$ for every $e \in E$. Therefore, the existence of the property Dense does not necessarily imply a clique of honest parties of large cardinality that agree with each other.

2. **Trivariate:** In trivariate sharing, the parties hold bivariate polynomials as shares, and they exchange univariate polynomials. If $P_j$ and $P_k$ have a common set of neighbors $E$ of cardinality $2t+1$, then *they necessarily hold shares that agree with each other* [5]. The key idea is that $P_k$ exchanges with each party $P_e$ for $e \in E$ univariate polynomials. Since the univariate polynomials agree, they also agree on the index $j$. The $2t+1$ points of parties in $E$ on the index $j$ uniquely define the univariate polynomials that $P_k$ expects to receive from $P_j$. Thus, even though $P_k$ did not hear yet the message from $P_j$, and $P_j$ is honest, it knows that the shares would agree, even if the dealer is corrupted. A similar argument holds for $P_j$. The formal argument appears in Section 3.

To conclude, to share the trivariate polynomial, we have a total of $\mathcal{O}(n^3 \log n)$ bits over point-to-point channel together with $\mathcal{O}(n^2 \log n)$ bits over the broadcast channel. If we batch $n$ instances together, we get a total of $\mathcal{O}(n^4 \log n)$ bits over point-to-point channels and no broadcast for sharing $\mathcal{O}(n^4)$ secrets, each of $\log n$ bits. I.e., we obtain an overhead of $\mathcal{O}(1)$.

Note, however, that the sharing is weak. Namely, we know that there is a well-defined trivariate polynomial, but we cannot necessarily reconstruct it robustly. In particular, even in the case of an honest dealer, reconstruction might fail. In that case, however, the dealer can shun at least $t/2+1$ corrupted parties. We show the reconstruction in Section 6.

However, as we will see, the reconstruction is not required for our MPC, and the sharing itself suffices. We gave it for completeness and as it might be useful as an independent primitive.

### 2.3 Our MPC Protocol

Our MPC protocol follows the following structure: an offline phase in which the parties generate Beaver triplets [6], and an online phase in which the parties compute the circuit while consuming those triples.

**Beaver triplets generation.** Our goal is to distribute (Shamir, univariate degree-$t$) shares of random secret values $a, b$, and $c$, such that $c = ab$. If the circuit contains $C$ multiplication gates, we need $C$ such triplets. Towards that end, we follow the same steps as in [22], and generate such triplets in three stages:

1. **Triplets with a dealer:** Each party generates shares of $a_i, b_i, c_i$ such that $c_i = a_i \cdot b_i$. We generate all the triplets in parallel using expected $\mathcal{O}(1)$ rounds. We overview this step below in Section 2.4. Our main contribution is in improving this step using the asynchronous weak-binding secret sharing with $\mathcal{O}(1)$-overhead. Despite the fact that this secret sharing with $\mathcal{O}(1)$-overhead is not robust, it suffices since it is used as part of a perfect zero-knowledge proof where the dealer proves that the shares it generated indeed correspond to a product relation. If the sharing fails, then we can simply ignore the contribution of that dealer.

[5] In fact, as shown in [5], having a common set of honest neighbors of size $t + t/2 + 1$ suffices for $P_j$ and $P_k$ to hold shares that are consistent with the same trivariate polynomial of degree $t + t/2$ in each variable.

In our protocol, each party acts as a dealer to generate $C/n$ triplets. This step requires a cost of $\mathcal{O}(n^4 \log n + C \log n)$ communication for a single party and an overall cost of $\mathcal{O}(n^5 \log n + Cn \log n)$ for all the parties together.

2. **Agreeing on a core set (ACS):** All triplet generations of honest dealers eventually terminate, whereas those of corrupted dealers might never terminate. However, if one honest party sees that the triplet generation of some player $P_i$ terminates, then all the honest parties will eventually receive output in that triplet generation.
   As $t$ instances might never terminate, the parties proceed with the protocol once at least $n-t$ parties successfully complete the triplet generation. Since parties might receive messages in different orders, they have to agree on the set of parties for which their triplet generation was successful.
   The set Core of at least $n - t$ parties (who have successfully completed the triple sharing) will be chosen using the agreement on core set (ACS) protocol. The communication cost of ACS from [17] is $\mathcal{O}(n^7 \log n)$, and its run-time is $\log n$. In [3], the communication is improved to $\mathcal{O}(n^5 \log n)$, and run-time is expected constant time. We use the latter to quote our complexity. [6]
3. **Triplets with no dealer:** Once agreed which triplets to consider (Core), using triplet extraction of [22], we can extract from a total of $\frac{C(n-t)}{n}$ triplets generated by the dealers in Core, $\mathcal{O}(C)$ triplets where no party knows the underlying values. This step costs $\mathcal{O}(n^2 \log n + Cn \log n)$.

In summary, for generating $C$ triplets we pay a total of $\mathcal{O}(n^5 \log n + Cn \log n)$.

The MPC protocol follows the standard structure where each party shares its input, and the parties evaluate the circuit gate-by-gate, or more exactly, layer-by-layer. In each multiplication gate, the parties have to consume one multiplication triple. Using the method of [22], if the $i$th layer of the circuit contains $C_i$ multiplications (for $i \in [D]$, where $D$ is the depth of the circuit), the evaluation costs $\mathcal{O}(n^2 \log n + C_i \cdot n \log n)$. Summing over all layers, this is $\sum_{i\in[D]}(n^2 + nC_i) \log n = (Dn^2 + Cn) \log n$. Together with the generation of the triplets, we get the claimed $\mathcal{O}((Cn+Dn^2+n^5) \log n)$ cost as in Theorem 1.1. We refer the reader to Section 8 for further details on our MPC protocol.

One point worth addressing is that we parallelize the input-sharing phase to the triplet generation. In that case, the ACS protocol selects the parties for which their triplet generation and input sharing were successful. Moreover, the input-sharing phase uses the robust secret-sharing with $\mathcal{O}(n)$ overhead and not the non-robust $\mathcal{O}(1)$. After the parties obtain robust shares (either sharing inputs or of product relations), all other computations are merely reconstructions and linear combinations of shared values. Since messages of honest parties are guaranteed to be delivered, and we have at least $2t + 1$ honest parties in Core with at most $t$ corruptions, all reconstructions are guaranteed to terminate successfully and asynchrony has no effect.

### 2.4 Multiplication Triplets with a Dealer

The goal is that a dealer wishes to distribute shares of secret values $\vec{a}, \vec{b}, \vec{c}$ such that for every $i$ it holds that $c_i = a_i b_i$. Towards this end, the dealer plants $\vec{a}$ into some bivariate polynomial $A(\mathbf{x}, \mathbf{y})$ using the asynchronous VSS scheme that employs a bivariate polynomial of degree-$(t + t/2, t)$. Such a VSS is given in [22], which we slightly simplify and give it for completeness in Section 4. Specifically, $\vec{a}$ is placed at $(A(-\beta, 0))_{\beta\in 0,\ldots,t/2}$. Similarly, the dealer plants $\vec{b}$ into $B(\mathbf{x}, \mathbf{y})$ and $\vec{c}$ into

[6] Without [3], the cost of our entire MPC is $\mathcal{O}((Cn + Dn^2 + n^7) \log n)$ and $\mathcal{O}(D + \log n)$ expected-time.

$C(\mathbf{x}, \mathbf{y})$. It is important to note that we deploy robust AVSS here, to ensure that the triplets are shared via degree-$t$ polynomials (which is utilized by our MPC protocol). So we can plant only $\mathcal{O}(n)$ values in each bivariate polynomial. Specifically, the $i$th secret in $\vec{a}$ is shared via the $t$-degree polynomial $A(-i, \mathbf{y})$.

Next, the dealer has to prove, using a distributed zero-knowledge protocol, that indeed $c_i = a_i b_i$ for every $i$ (i.e., that $C(-\beta, 0) = A(-\beta, 0) \cdot B(-\beta, 0)$ for every $\beta \in \{0, \ldots, t/2\}$). The input of each party $P_j$ is a point on the univariate polynomials $A(-\beta, \mathbf{y}), B(-\beta, \mathbf{y})$ and $C(-\beta, \mathbf{y})$. The zero-knowledge proof shares and operates on the coefficients of the polynomials used for sharing $\vec{a}, \vec{b}, \vec{c}$. If the dealer shared $\mathcal{O}(M)$ triplets, then the zero-knowledge involves sharing of $\mathcal{O}(Mn)$ values. For simplicity, assume hereafter that $M = \mathcal{O}(n^2)$.

**Verifying product relation.** We now provide a detailed disposition of the zero-knowledge for product relations via the asynchronous weak-binding secret sharing. For simplicity of notation, the dealer has already (verifiably) shared, for every $u \in U = \{0, \ldots, t/2\}^2$, degree-$t$ polynomial $A^u(\mathbf{x}), B^u(\mathbf{x}), C^u(\mathbf{x})$. Each party $P_j$ holds shares $A^u(j), B^u(j), C^u(j)$. The goal of the dealer is to prove that $A^u(0) \cdot B^u(0) = C^u(0)$. The zero-knowledge proof shares and operates on the coefficients of the polynomials used for sharing. Since $|U| \in \mathcal{O}(n^2)$ and each product polynomial has $\mathcal{O}(n)$ coefficients, we have a total of $\mathcal{O}(n^3)$ coefficients to pack. We now need $\mathcal{O}(1)$ trivariate polynomials to pack all needed coefficients. This sharing does not need to produce $t$-sharing of the secrets. Rather the mere confirmation that the dealer commits to a unique set of secrets is good enough. Hence, as discussed in the previous section, we can utilize the asynchronous weak-binding secret sharing with light-enhanced features.

**Constructing the trivariate polynomials.** For every $u \in U$, define:

$$E^u(\mathbf{x}) := A^u(\mathbf{x}) \cdot B^u(\mathbf{x}) - C^u(\mathbf{x}) \ .$$

We redefine $U = V \times V$ where $V = \{0, \ldots, t/2\}$. Then, we need to verify that for every $(\beta, \gamma) \in V \times V$ it holds that

$$E^{(\beta,\gamma)}(0) = A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) - C^{(\beta,\gamma)}(0) = 0 \ ,$$

which implies that $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$. Since $A^{(\beta,\gamma)}(\mathbf{x})$, $B^{(\beta,\gamma)}(\mathbf{x})$, and $C^{(\beta,\gamma)}(\mathbf{x})$ are polynomials of degree-$t$, the polynomial $E^{(\beta,\gamma)}(\mathbf{x})$ is of degree-$2t$. We explicitly write

$$E^{(\beta,\gamma)}(\mathbf{x}) = e_1^{(\beta,\gamma)}\mathbf{x} + \ldots + e_{2t}^{(\beta,\gamma)}\mathbf{x}^{2t} \ .$$

Since the constant term of this polynomial is supposed to be 0 (if indeed $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$) we do not specify it. The dealer embeds the $2t$ coefficients of each of those $(t/2+1)^2$ polynomials in four trivariate polynomials, each of degree $t + t/2$ in $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. Note that each trivariate polynomial can pack $(t/2+1)^3$ values, where we have a total of $(t/2+1)^2 \cdot 2t$ values to pack; We therefore need four trivariate polynomial $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4$ (in each we pack $t/2(t/2+1)^2$ while we can pack $(t/2+1)^3$ values; i.e., we do not fully pack it). Specifically (where in all of the following rows we quantify over all $k \in [1, \ldots, t/2]$, and $(\beta, \gamma) \in V \times V$):

| The coefficients | Embedded in the trivariate | The embedding |
|---|---|---|
| $e_1^{(\beta,\gamma)}, \ldots, e_{t/2}^{(\beta,\gamma)}$ | $\mathbf{S}_1(\mathbf{x},\mathbf{y},\mathbf{z})$ | $\mathbf{S}_1(-\beta,-k,-\gamma) = e_k^{(\beta,\gamma)}$ |
| $e_{t/2+1}^{(\beta,\gamma)}, \ldots, e_t^{(\beta,\gamma)}$ | $\mathbf{S}_2(\mathbf{x},\mathbf{y},\mathbf{z})$ | $\mathbf{S}_2(-\beta,-k,-\gamma) = e_{t/2+k}^{(\beta,\gamma)}$ |
| $e_{t+1}^{(\beta,\gamma)}, \ldots, e_{t+t/2}^{(\beta,\gamma)}$ | $\mathbf{S}_3(\mathbf{x},\mathbf{y},\mathbf{z})$ | $\mathbf{S}_3(-\beta,-k,-\gamma) = e_{t+k}^{(\beta,\gamma)}$ |
| $e_{t+t/2+1}^{(\beta,\gamma)}, \ldots, e_{2t}^{(\beta,\gamma)}$ | $\mathbf{S}_4(\mathbf{x},\mathbf{y},\mathbf{z})$ | $\mathbf{S}_4(-\beta,-k,-\gamma) = e_{t+t/2+k}^{(\beta,\gamma)}$ |

This fixes $t/2(t/2+1)^2$ points on each trivariate polynomial. The dealer chooses random trivariate polynomials under the above constraints. It then "secret shares" those trivariate polynomials among the parties. Each party $P_i$ receives the share:

$$\mathsf{share}_i = (\mathbf{S}_r(\mathbf{x},\mathbf{y},i),\ \mathbf{S}_r(\mathbf{x},i,\mathbf{y}),\ \mathbf{S}_r(i,\mathbf{y},\mathbf{z}))_{r\in[4]}\ .$$

Let us assume for simplicity that $\mathsf{share}_i$s are distributed such that *all* the honest parties' $\mathsf{share}_i$s uniquely define four trivariate polynomials of degree at most $t+t/2$ in each variable. We have to perform the following checks:

1. The shares that the dealer distributes uniquely define four trivariate polynomials of degree at most $t+t/2$ in each variable.
2. The trivariate polynomials define coefficients of polynomials $E^{(\beta,\gamma)}(\mathbf{x})$ for every $(\beta,\gamma)\in V\times V$. It should hold that for at least $2t+1$ indices $i$:

$$a_i^{(\beta,\gamma)}\cdot b_i^{(\beta,\gamma)} - c_i^{(\beta,\gamma)} = E^{(\beta,\gamma)}(i) = i\cdot e_1^{(\beta,\gamma)} + i^2\cdot e_2^{(\beta,\gamma)} + \ldots + i^{2t}\cdot e_{2t}^{(\beta,\gamma)} \tag{1}$$

Since each $A^{(\beta,\gamma)}(\mathbf{x}), B^{(\beta,\gamma)}(\mathbf{x}), C^{(\beta,\gamma)}(\mathbf{x})$ is of degree-$t$, the polynomial $A^{(\beta,\gamma)}(\mathbf{x})\cdot B^{(\beta,\gamma)}(\mathbf{x}) - C^{(\beta,\gamma)}(\mathbf{x})$ is of degree-$2t$. If Eq. (1) holds for at least $2t+1$ indices $i$, then this polynomial is exactly $E^{(\beta,\gamma)}(\mathbf{x})$. Since $E^{(\beta,\gamma)}(0)=0$, we have that $A^{(\beta,\gamma)}(0)\cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$. To require that the above verification holds for at least $2t+1$ honest parties, we actually require that it holds for at least $3t+1$ parties.

To check that the trivariate shares pack the correct values, each $P_i$ must be able to reconstruct $E^{(\beta,\gamma)}(i)$, where

$$\begin{aligned} E^{(\beta,\gamma)}(i) = \ & \textstyle\sum_{k=1}^{t/2}\Big(i^k\cdot\underbrace{\mathbf{S}_1(-\beta,-k,-\gamma)}_{e_k^{(\beta,\gamma)}} + i^{t/2+k}\cdot\underbrace{\mathbf{S}_2(-\beta,-k,-\gamma)}_{e_{t/2+k}^{(\beta,\gamma)}} \\ & + i^{t+k}\cdot\underbrace{\mathbf{S}_3(-\beta,-k,-\gamma)}_{e_{t+k}^{(\beta,\gamma)})} + i^{t+t/2+k}\cdot\underbrace{\mathbf{S}_4(-\beta,-k,-\gamma)}_{e_{t+t/2+k}^{(\beta,\gamma)}}\Big) \end{aligned}$$

We therefore let $P_i$ get the bivariate polynomial from the dealer in addition to $\mathsf{share}_i$, which embeds $(E^{(\beta,\gamma)}(i))_{(\beta,\gamma)\in V\times V}$:

$$T_i(\mathbf{x},\mathbf{z}) = \sum_{r=1}^{4}\sum_{k=1}^{t/2} i^{(r-1)\cdot(t/2)+k}\cdot\mathbf{S}_r(\mathbf{x},-k,\mathbf{z})\ . \tag{2}$$

Note that for every $(\beta,\gamma)$, $T_i(-\beta,-\gamma) = E^{(\beta,\gamma)}(i)$.

We now need a mechanism for $P_i$ to verify that the dealer indeed passes on a correct bivariate polynomial $T_i$ consistent with the unique trivariate polynomials $(\mathbf{S}_r)_{r\in\{1,2,3,4\}}$ defined by $\mathsf{share}_j$s of the honest parties. We observe that $T_i(\mathbf{x}, j)$ can be computed by $P_j$ based on $\mathsf{share}_j$. Since the degree of $T_i$ is $t + t/2$ in both variables, it is enough if $t + t/2 + 1$ honest parties or alternatively a total of $2t + t/2 + 1$ parties confirm that their $T_i(\mathbf{x}, j)$ is consistent with $P_i$'s received $T_i$. So, to let $P_i$ verify $T_i(\mathbf{x}, \mathbf{z})$, the parties jointly perform the following:

1. On holding $(\mathbf{S}_r(\mathbf{x}, \mathbf{y}, j))_{r\in[4]}$ as part of $\mathsf{share}_j$, for every $r \in [4]$, $P_j$ evaluates $\mathbf{S}_r(\mathbf{x}, \mathbf{y}, j)$ on $t/2$ values $\mathbf{y} = -1, \ldots, -t/2$ to obtain $\mathbf{S}_r(\mathbf{x}, -1, j), \ldots, \mathbf{S}_r(\mathbf{x}, -t/2, j)$.
2. Define $t_i^j(\mathbf{x}) = \sum_{r=1}^{4} \sum_{k=1}^{t/2} i^{(r-1)\cdot(t/2)+k} \cdot \mathbf{S}_r(\mathbf{x}, -k, j)$.
3. Send $P_i$ the univariate polynomial $t_i^j(\mathbf{x})$.

$P_i$ can then verify if $t_i^j(\mathbf{x}) = T_i(\mathbf{x}, j)$ for at least $2t + t/2 + 1$ $P_j$s.

We require the dealer to find a set of size at least $3t + 1$ such that: (a) the honest parties in it must define four unique trivariate polynomials; (b) every honest party in it holds $T_i$ that is consistent with every honest party in the set; and (c) every honest party in it must have successfully verified Equation (1). We note that an honest dealer will always be able to find eventually such a set.

**Modeling.** We describe our protocol in the Simple UC (SUC) framework due to Canetti, Cohen, and Lindell [19]. This implies standard UC security [18]. We try to avoid over-formalism in the protocol descriptions (e.g., we ignore $\mathsf{sid}$ while those are implicit). As standard secret-shared protocols in the perfect setting, we conjecture that our protocols are also adaptively secure.

**Organization.** The rest of the paper is organized as follows. In Section 3, we provide the preliminaries, mainly overview the SUC framework. In Section 5, we provide full details of our zero-knowledge protocol, i.e., verifying product relation. In Section 6, we provide our rate-1, asynchronous weak-binding secret sharing. We remark again that we do not use this secret sharing directly, and we provide it just for completeness. In Section 7, we show verifiable triple sharing, followed by the MPC protocol in Section 8.

# 3 Preliminaries

**Notations.** Our protocols are defined over a finite field $\mathbb{F}$ where $|\mathbb{F}| > n + t + 1$. We denote the elements by $\{-t, \ldots, 0, 1 \ldots, n\}$. We use $\langle v \rangle$ to denote the degree-$t$ Shamir-sharing of a value $v$ among parties in $\mathcal{P}$, and $\langle v \rangle_i$ to denote the share held by a party $P_i$.

## 3.1 Asynchronous Secure Computation and SUC

We consider an asynchronous network where the parties are $\mathcal{P} = \{P_1, \ldots, P_n\}$. The parties are connected via pairwise ideal private channels. To model asynchrony, messages sent on a channel can be arbitrarily delayed, however, they are guaranteed to be eventually received after some finite number of activations of the adversary. In general, the order in which messages are received might be different from the order in which they were sent. Yet, to simplify notation and improve readability, we assume that the messages that a party receives from a channel are guaranteed to be delivered in the order they were sent. This can be achieved using standard techniques – counters, and acknowledgements, and so we just make this simplification assumption.

We prove our protocols in this simplified universally composable setting (SUC), which is a simplified UC model aimed for modeling secure protocols, formalized by Canetti, Cohen and Lindell [20], and implies UC security. We briefly review the definitions, but many details are left out, see [20] for additional information.

**Main difference from SUC.** The SUC model allows the adversary to also drop messages, and the adversary is not limited to eventually deliver all messages. To model "eventual delivery" (which is the essence of the asynchronous model), we limit the capabilities of the adversary and quantify over adversaries that eventually transmit each message in the network (i.e., they do not drop messages). Formally, any message sent must be delivered after some finite number of activations of the adversary.

As in SUC, the parties are modeled as interactive Turing machines, with code tapes, input tapes, outputs tapes, incoming communication tapes, outgoing communication tape, random tape and work tape.

**Communication.** In each execution there is an *environment* $\mathcal{Z}$, an *adversary* $\mathcal{A}$, participating *parties* $P_1, \ldots, P_n$, and possibly an *ideal functionality* $\mathcal{F}$ and a *simulator* $\mathcal{S}$. The parties, adversary and ideal functionality are connected in a star configuration, where all communication is via an additional *router machine* that takes instructions from the adversary. That is, each entity has one outgoing channel to the router and one incoming channel. When $P_i$ sends a message to $P_j$, it sends it to the router, and the message is stored by the router. The router delivers to the adversary a general information about the message (i.e., "a header" but not the "content". That is, the adversary can know the type of the message and its size, but cannot see its content). When the adversary allows the delivery of the message, the router delivers the message to $P_j$. As mentioned, we quantify only over all adversaries that eventually deliver all messages. In particular, even in an execution with an ideal functionality, communication between the parties and this functionality is done via the router machine and is subject to (finite) delivery delays imposed by the adversary.

Note that the router machine is also part of the ideal model. When the functionality gives for instance, some output to party $P_j$, then this is performed via the router, and the simulator is notified. Thus, if the adversary, for instance, delays the delivery of the output of some party $P_j$, we do not explicitly mention that in the functionality (e.g., "wait to receive $\mathsf{OK}_j$ from the adversary and then deliver the output to $P_j$"), yet it is captured by the model.

Finally, the environment $\mathcal{Z}$ communicates with the adversary directly and not via the router. In particular, the environment can communicate only with the adversary (and it cannot communicate even with the ideal functionality $\mathcal{F}$). In addition, $\mathcal{Z}$ can *write* inputs to the honest parties' input tapes and can *read* their output tapes.

**Execution in the ideal model.** In the ideal model we consider an execution of the environment $\mathcal{Z}$, dummy parties $P_1, \ldots, P_n$, the router, a functionality $\mathcal{F}$ and a simulator $\mathcal{S}$. In the ideal model with a functionality $\mathcal{F}$ the parties follow a fixed ideal-model protocol. The environment is first activated with some input $z$. The environment delivers the inputs to the dummy honest parties, which forward the inputs to the functionality (recall that this is done via the router, which then gives some leakage about the message header to $\mathcal{S}$, which can adaptively delay the delivery by any finite amount). Moreover, $\mathcal{Z}$ can also give some initial inputs to the corrupted parties via $\mathcal{S}$. At a later stage where the dummy parties receive output from the functionality $\mathcal{F}$, they just write the outputs on their output tapes (and $\mathcal{Z}$ can read those outputs). The simulator $\mathcal{S}$ can send messages to $\mathcal{Z}$ and to the functionality $\mathcal{F}$. The simulator cannot directly communicate with the

participating parties. We stress that in the ideal model, the simulator $\mathcal{S}$ interacts with $\mathcal{Z}$ in an online way, and the environment can essentially read the outputs of the honest parties, and query the simulator (i.e., can see the view of the adversary) at any point of the execution. At the end of the interaction, $\mathcal{Z}$ outputs some bit $b$. We denote by $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(z)$ an execution of this ideal model of the functionality $\mathcal{F}$ with a simulator $\mathcal{S}$ and environment $\mathcal{Z}$, which starts with an input $z$.

**Execution in the real model with protocol $\pi$.** In the real model, there is no ideal functionality and the participating parties are $\mathcal{Z}$, the parties $P_1, \ldots, P_n$, the router and the real-world adversary $\mathcal{A}$. The environment is first activated with some input $z$, and it can give inputs to the honest parties, as well as some initial inputs to the corrupted parties controlled by the adversary $\mathcal{A}$. The parties run the protocol $\pi$ as specified, while the corrupted parties are controlled by $\mathcal{A}$. The environment can see at any point the outputs of the honest parties, and communicate directly with the adversary $\mathcal{A}$ (and see, without loss of generality, its partial view). At the end of the execution, the environment outputs some bit $b$. We denote by $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(z)$ an execution of this real model with the protocol $\pi$, the real-world adversary $\mathcal{A}$ and the environment $\mathcal{Z}$, which starts with some input $z$.

**Definition 3.1.** *We say that an adversary $\mathcal{A}$ is an* asynchronous adversary *if any message that it receives from the router, it allows its delivery within some finite number of activations of $\mathcal{A}$.*

**Definition 3.2.** *Let $\pi$ be a protocol and let $\mathcal{F}$ be an ideal functionality. We say that $\pi$ securely computes $\mathcal{F}$ in the asynchronous setting if for every real-model asynchronous adversary $\mathcal{A}$ there exists an ideal-world adversary $\mathcal{S}$ that runs in polynomial time in $\mathcal{A}$'s running time, such that for every $\mathcal{Z}$:*

$$\{\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(z)\}_z \equiv \{\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(z)\}_z$$

### 3.2 Hybrid Model and Composition

We also consider a hybrid model where the parties follow some protocol $\pi$ as in the real model but also have access to some ideal functionality $\mathcal{F}$, and the protocol instructs the parties to send messages to that ideal functionality and how to process its response. As previously, all communication is performed via the router. We denote the output of $\mathcal{Z}$ from a hybrid execution of $\pi$ with ideal calls to $\mathcal{F}$ as $\text{HYBRID}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}(z)$, where $\mathcal{A}, \mathcal{Z}, z$ are as above. We call this as $\mathcal{F}$-hybrid model.

The composition theorem states that if a protocol $\pi$ realizes $\mathcal{F}$ in the $\mathcal{G}$-Hybrid model, and a protocol $\rho$ realizes $\mathcal{G}$ in the plain model, then the protocol $\pi^\rho$ realizes $\mathcal{F}$ in the plain model. To clarify, when the parties in $\pi$ call $\mathcal{G}$ in the protocol $\pi$, in the protocol $\pi^\rho$ the parties will invoke the code of the protocol $\rho$. This is a difference between the SUC and UC model, in which in the UC model, each subprotocol is invoked as a separate interactive Turing machine, which introduces some extra complexity. See [19] for elaborated discussion.

### 3.3 Asynchronous Broadcast and Agreement on a Core Set

**A-Cast.** Bracha's asynchronous broadcast (A-Cast) protocol [13] allows a sender to send a message $m$ identically to all the parties at a cost of $\mathcal{O}(n^2\ell)$ where $\ell$ is the length of the message in bits. If the sender is honest, then all the honest parties eventually terminate with output $m$. If the sender is corrupt, and some honest party terminates with the output $m'$, then all the honest parties eventually terminate with the same output $m'$. In our protocol, the cost of A-Cast can be amortized

over multiple instances of triple generation with the same dealer. For ease of description, we use the following representation in our protocols.

- "$P_i$ broadcasts a message $m$" represents an instance of A-Cast with $P_i$ as the sender.
- "$P_j$ receives a message $m$ from the broadcast of $P_i$" represents that $P_j$ outputs $m$ in the instance of A-Cast with $P_i$ as the sender.

**Agreement on a Core Set (ACS).** The ACS primitive [17] allows parties to agree on a common set of at least $n-t$ parties $\mathsf{Core} \subset \mathcal{P}$, such that each party in $\mathsf{Core}$ satisfies some predefined property $\mathsf{prop}$ which has the following features:

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

We model ACS as a functionality below. In the functionality, each party $i$ receives $(\mathsf{record}, i, k)$ commands with $k \in [n]$ and sends them to the functionality. Each party is guaranteed to receive at least $n-t$ such commands and if some honest $i$ receives a $(\mathsf{record}, i, k)$ command, every honest $j$ is guaranteed to eventually receive a $(\mathsf{record}, j, k)$ command as well. In addition, parties can receive $\mathsf{receiveS}()$ commands which they forward to the functionality. The functionality then returns a set $S \subseteq [n]$ as a response. Note that in either case, all parties eventually receive the same set of indices $S \subseteq [n]$, such that for every $k \in S$ at least one honest party received a $(\mathsf{record}, i, k)$ command.

---

**Functionality 3.3: Agreement on a Core Set**

---

The functionality is parameterized by a set of corrupted parties $I \subseteq [n]$. Initialize sets $S_i \leftarrow \emptyset$ for every $i \in [n]$ and $S \leftarrow \perp$. In addition, initialize $\mathsf{returned} \leftarrow 0$.

1. $(\mathsf{record}, i, k)$: Upon receiving this command from party $i$, add the index $k$ to $S_i$. Forwards $(\mathsf{record}, i, k)$ to the adversary. If $|S_i| \geq n - t$ then set $i$ as $\mathsf{ready}$. If $n - t$ honest parties are $\mathsf{ready}$, then set $S$ to be the set of all indices $k \in [n]$ such that there exists some $\ell \notin I$ for which $(\mathsf{record}, \ell, k)$ was sent.
2. $(\mathsf{set}, S')$: Upon receiving this command from the adversary, check that $S' \subseteq [n]$ and that $|S'| \geq n - t$. Moreover, check that for every $k \in S'$, there exists some $\ell \notin I$ for which $k \in S_\ell$ (i.e., $P_\ell$ has submitted $(\mathsf{record}, \ell, k)$). If all those conditions hold, and $\mathsf{returned} = 0$, then store $S \leftarrow S'$.
3. $\mathsf{receiveS}()$: Upon receiving this command from some party $i$, if $S \neq \perp$, set $\mathsf{returned} \leftarrow 1$. Return $S$.

---

Looking ahead, we use the ACS primitive to identify a common set of parties with the property that a verifiable triple sharing instance and a asynchronous VSS instance initiated by a party in this set must terminate eventually for all the honest parties.

### 3.4 Finding a STAR in a Graph

**Definition 3.4.** *Let $G$ be a graph over the nodes $[n]$. We say that a pair $(\mathsf{C}, \mathsf{D}) \subseteq [n]^2$ such that $\mathsf{C} \subseteq \mathsf{D}$ is an $(n,t)$-$\mathsf{STAR}$ in $G$ of the following holds:*

- $|\mathsf{C}| \geq n - 2t$,
- $|\mathsf{D}| \geq n - t$,
- *For every* $c \in \mathsf{C}$ *and every* $d \in \mathsf{D}$ *the edge* $(c, d)$ *exists in* $G$.

Canetti [15] 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. For completeness, we describe the algorithm below.

**Algorithm 3.5: STAR Algorithm** $(G, n, t)$

- **Input:** undirected graphs $G$ (over the nodes $\{1, \ldots, n\}$), a parameter $t$.
  1. Find a maximum matching $M$ in $\overline{G}$ (i.e., in the complement graph). Let $N$ be the set of matched nodes (namely, the endpoints of the edges in $M$).
  2. Let $T$ be the set of triangle-heads, i.e., all vertices that are not endpoints of the matching in $G$, but they have two neighbors in the matching.

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

     Let $\mathsf{C} := [n] \setminus (N \cup T)$.
  3. Let $\mathsf{D}$ the set of unmatched nodes that have no neighbors in $\mathsf{C}$ in $\overline{G}$. That is, the set:

     $$\mathsf{D} := \{j \notin N \mid \forall i \in \mathsf{C},\ \ (i, j) \notin \overline{G}\} .$$

  4. **Output:** If $|\mathsf{C}| \geq n - 2t$, and $|\mathsf{D}| \geq n - t$ then output $(\mathsf{C}, \mathsf{D})$. Otherwise, output $\perp$.

**Claim 3.6.** *Let* $I$ *be a set of cardinality of size at most* $t$. *Let* $G$ *be an undirected graph over* $[n]$ *such that for every* $j, k \notin I$ *it holds that* $(j, k) \in G$. *Then,* $\mathsf{C} \setminus I$ *contains at least* $n - 2t$ *indices.*

*Proof.* As in [1] we get that the number of parties in $(N \cup T) \setminus I$ is at most $t$. Since $\mathsf{C}$ is defined as $[n] \setminus (N \cup T)$, we get that $\mathsf{C} \setminus I$ is at least $n - 2t$. □

**Claim 3.7** ([16])**.** *Let* $G$ *be a graph over* $n$ *vertices that contains a clique of size* $n - t$. *Then, the algorithm outputs sets* $(\mathsf{C}, \mathsf{D})$.

### 3.5 Bivariate Polynomials

We consider bivariate polynomials of degree $t + q$ in $\mathbf{x}$ and degree $t$ in $\mathbf{y}$. Such polynomials can be written as follows: $S(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{t+q} \sum_{j=0}^{t+q} a_{i,j} \mathbf{x}^i \mathbf{y}^j$. Looking ahead, in Section 4 we use $q = t$. We have:

**Claim 3.8.** *Let* $t$ *be a nonnegative integer, let* $H \subset [n]$ *be a set of cardinality* $t + 1$ *and let* $(f_h(\mathbf{x}))_{h \in H}$ *be* $t + 1$ *univariate polynomials of degree at most* $t + q$. *Then, there exists a unique bivariate polynomial* $S(\mathbf{x}, \mathbf{y})$ *of degree* $t + q$ *in* $\mathbf{x}$ *and degree* $t$ *in* $\mathbf{y}$ *satisfying for every* $h \in H$: $S(\mathbf{x}, h) = f_h(\mathbf{x})$.

### 3.6 Trivariate Polynomials

We consider trivariate polynomial of degree $t+q$ in variables $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Such polynomial can be written as follows:

$$S(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{i=0}^{t+q} \sum_{j=0}^{t+q} \sum_{k=0}^{t+q} a_{i,j,k} \mathbf{x}^i \mathbf{y}^j \mathbf{z}^k \ .$$

Looking ahead, in our protocol we consider $q = t/2$.

**Claim 3.9** (Interpolation)**.** *Let $t$ be a nonnegative integer, let $H \subset [n]$ be a set of cardinality $t+q+1$, and let $(S_h(\mathbf{x}, \mathbf{y}))_{h \in H}$ be $t+q+1$ bivariate polynomials of degree at most $t+q$ (in both $\mathbf{x}, \mathbf{y}$) each. Then, there exists a unique trivariate polynomial $S(\mathbf{x}, \mathbf{y}, \mathbf{z})$ of degree $t+q$ such that for every $h \in H$:*

$$S(\mathbf{x}, \mathbf{y}, h) = S_h(\mathbf{x}, \mathbf{y})$$

*Proof.* Define the trivariate polynomial $S(\mathbf{x}, \mathbf{y}, \mathbf{z})$ via a generalization of the Lagrange interpolation. For every $h \in H$, define the trivariate polynomial $S_h(\mathbf{x}, \mathbf{y}, \mathbf{z})$ as follows:

$$S_h(\mathbf{x}, \mathbf{y}, \mathbf{z}) = S_h(\mathbf{x}, \mathbf{y}) \cdot \frac{\prod_{j \in H \setminus \{h\}} (\mathbf{z} - j)}{\prod_{j \in H \setminus \{h\}} (h - j)} \ .$$

Note that $S_h(\mathbf{x}, \mathbf{y}, h) = S_h(\mathbf{x}, \mathbf{y})$, and for every $j \in H \setminus \{h\}$, $S_h(\mathbf{x}, \mathbf{y}, j) = 0$. Moreover, $S_h(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is of a trivariate polynomial of degree $t+q$. Then, define:

$$S(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{h \in H} S_h(\mathbf{x}, \mathbf{y}, h) \ .$$

Clearly, for every $h \in H$ it holds that $S(\mathbf{x}, \mathbf{y}, h) = S_h(\mathbf{x}, \mathbf{y})$, and $S(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a trivariate polynomial of degree $t+q$. We now show that $S$ is unique. Assume that there exist two different polynomials $S_1(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and $S_2(\mathbf{x}, \mathbf{y}, \mathbf{z})$ that satisfy the conditions in the claim. Consider the polynomial $R(\mathbf{x}, \mathbf{y}, \mathbf{z}) = S_1(\mathbf{x}, \mathbf{y}, \mathbf{z}) - S_2(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Since $S_1$ and $S_2$ are two trivariate polynomial of degree $t+q$ then $R$ is also a trivariate polynomial of degree $t+q$. Moreover, for every $h \in H$ it holds that $R(\mathbf{x}, \mathbf{y}, h) = 0$. For every $\alpha, \beta \in \mathbb{F}$, consider the degree $t+q$ univariate polynomial $R(\alpha, \beta, \mathbf{z})$. It holds that $R(\alpha, \beta, h)$ for all $h \in H$, and thus it equals 0 on $t+q+1$ points, i.e., this is the all-zero univariate polynomial. Moreover, for every $\alpha, \beta, \gamma \in \mathbb{F}$ it holds that $R(\alpha, \beta, \gamma) = 0$, that is, $R(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is the all-zero polynomial and thus $S_1(\mathbf{x}, \mathbf{y}, \mathbf{z}) = S_2(\mathbf{x}, \mathbf{y}, \mathbf{z})$. □

## 4 Verifiable Packed Bivariate Secret Sharing

In this section, we provide a protocol for packing $\mathcal{O}(n)$ secretes in a bivariate polynomial. We model the packed VSS in Functionality 4.1 below. The protocol is a simplification of the protocol of [22]: instead of the dealer re-broadcasting sets, we show how to do it with just a single broadcast message. Moreover, as opposed to [22], we provide full simulation in the SUC framework.

**Functionality 4.1: Packed VSS Functionality**

The functionality is parameterized by the set of corrupted parties $I \subseteq [n]$.

1. The dealer sends to the functionality its input $S(\mathbf{x}, \mathbf{y})$.

2. The functionality verifies that $S$ is of degree at most $3t/2$ in $\mathbf{x}$ and degree at most $t$ in $\mathbf{y}$. If not, it does not terminate.
3. If the above condition does hold, then the functionality sends to the ideal adversary the shares $S(\mathbf{x}, i), S(i, \mathbf{y})$ for every $i \in I$, and for each honest party $P_j$ it sends $S(\mathbf{x}, j), S(j, \mathbf{y})$.

---

**Definition 4.2.** *For an undirected graph $G = (V, E)$ with $V = [n]$, we say that the four sets $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ are* valid *if the following conditions are satisfied:*

1. $|C| \geq 2t + 1$, $\mathsf{C} \subseteq \mathsf{D}$, *and* $|\mathsf{D}|, |\mathsf{G}|, |\mathsf{F}| \geq 3t + 1$.
2. *For every $c \in \mathsf{C}$ and $d \in \mathsf{D}$ it holds that $(c, d)$ is an edge in $G$.*
3. *For every $g \in \mathsf{G}$ it holds that $|\Gamma(g) \cap \mathsf{C}| \geq 2t + 1$.*
4. *For every $f \in \mathsf{F}$ it holds that $|\Gamma(f) \cap \mathsf{G}| \geq 3t + 1$.*

---

**Protocol 4.3: Packed VSS Protocol**

---

- **Input:** The dealer holds a bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ of degree $3t/2$ in $\mathbf{x}$ and degree $t$ in $\mathbf{y}$. Other parties hold no input.
- **The protocol:**
  1. **(Dealing Shares):** The dealer sends $(\mathsf{shares}, D, i, (f_i(\mathbf{x}), g_i(\mathbf{y}))$ with $f_i(\mathbf{x}) = S(\mathbf{x}, i)$ and $g_i(\mathbf{y}) = S(i, \mathbf{y})$ to each party $P_i$.
  2. **(Pairwise Consistency Checks:)** Each party $P_i$ does as follows
     (a) Upon receiving shares from the dealer, $P_i$ verifies that $f_i(\mathbf{x})$ is of degree at most $3t/2$ and $g_i(\mathbf{y})$ is of degree at most $t$. If so, $P_i$ sends $(\mathsf{exchange}, i, j, f_i(j), g_i(j))$ to every $P_j$.
     (b) Upon receiving shares $(\mathsf{exchange}, j, i, u_{j,i}, v_{j,i})$ from $P_j$, check that $u_{j,i} = g_i(j)$ and $v_{j,i} = f_i(j)$. If so, broadcast the message $\mathsf{Good}(i, j)$.
  3. **(Valid $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ finding:)** The dealer does as follows
     (a) Initialize an undirected graph $G$ where $V = [n]$. Upon viewing broadcasted messages $\mathsf{Good}(k, \ell)$ broadcasted by $P_k$ and $\mathsf{Good}(\ell, k)$ broadcasted by $P_\ell$, add the edge $(k, \ell)$ to $G$.
     (b) Run the $\mathsf{STAR}$ algorithm (Algorithm 3.5) to find sets $(\mathsf{C}, \mathsf{D}) \subset [n]^2$ where $|\mathsf{C}| \geq 2t+1$, $|\mathsf{D}| \geq 3t + 1$, and for every $c \in \mathsf{C}$ and $d \in \mathsf{D}$, the edge $(c, d)$ is in $G$.
     (c) Let $\mathsf{G}$ be the set of parties that agree with at least $2t + 1$ parties in $\mathsf{C}$. That is, initialize $\mathsf{G} = \emptyset$ and add $i$ to $\mathsf{G}$ if $|\Gamma(i) \cap \mathsf{C}| \geq 2t + 1$.
     (d) Let $\mathsf{F}$ be the set of parties that agree with at least $3t + 1$ parties in $\mathsf{G}$. That is, initialize $\mathsf{F} = \emptyset$ and add $i$ to $\mathsf{F}$ if $|\Gamma(i) \cap \mathsf{G}| \geq 3t + 1$.
     (e) If $|\mathsf{C}| \geq 2t+1$, $\mathsf{C} \subseteq \mathsf{D}$, and $|\mathsf{D}|, |\mathsf{G}|, |\mathsf{F}| \geq 3t+1$, then broadcast $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$. Otherwise, continue to listen to $\mathsf{Good}$ messages, and repeat.
  4. **(Verifying $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$:)** Each party $P_i$:
     (a) Initialize an undirected graph $G_i$ where $V = [n]$. Upon viewing broadcasted messages $\mathsf{Good}(k, \ell)$ from $P_k$ and $\mathsf{Good}(\ell, k)$ from $P_\ell$, add the edge $(k, \ell)$ to $G_i$.
     (b) Check if $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ is valid for the graph $G_i$. If not, then continue to listen to $\mathsf{Good}$ message, and with each new edge, re-check validation.

5. **(Deciding on output)** Once $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ are valid for $G_i$, $P_i$ outputs its share as follows and terminates subsequently
   - **(Parties $i \in \mathsf{G} \cap \mathsf{F}$)** Output $(f_i(\mathbf{x}), g_i(\mathbf{y}))$.
   - **(Parties $i \notin \mathsf{G}$)**
     (a) Consider the messages $(u_{j,i}, v_{j,i})$ from Step 2b for parties $j \in \mathsf{F}$, and using Reed Solomon decoding, try to decode the unique univariate polynomial $g_i(\mathbf{y})$ of degree-$t$ satisfying $g_i(j) = u_{j,i}$ for all but $t$ values in $\mathsf{F}$. If there is no unique decoding, wait to receive additional points $(u_{j,i}, v_{j,i})$.
     (b) Once decoded, send $(\mathsf{reconstruct}, i, k, g_i(k))$ to each party $P_k$ for $k \notin \mathsf{F}$.
   - **(Parties $i \notin \mathsf{F}$)**:
     (a) Consider all points $(j, v_{j,i})$ obtained from messages $(\cdot, v_{j,i})$ from Step 2b for parties $j \in \mathsf{G}$, or points $(j, v_{j,i})$ obtained from messages $(\mathsf{reconstruct}, j, i, v_{j,i})$ from parties not in $\mathsf{G}$. Use Reed Solomon decoding procedure to decode the unique univariate polynomial $f_i(\mathbf{x})$ of degree $3t/2$ satisfying $f_i(j) = v_{j,i}$ for all but $t$ parties. If there is no unique decoding, wait to receive additional points $(u_{j,i}, v_{j,i})$ or $(\mathsf{reconstruct}, \ell, i, v_{\ell,i})$.
   - Once both $f_i(\mathbf{x})$ (for $i \notin \mathsf{F}$) and $g_i(\mathbf{y})$ (for $i \notin \mathsf{G}$) were reconstructed, then terminate and output $(f_i(\mathbf{x}), g_i(\mathbf{y}))$.

---

**Theorem 4.4.** *Let $n \geq 4t + 1$. Protocol 4.3 securely computes Functionality 4.1 in the presence of a malicious adversary controlling at most $t$ parties. It has a communication complexity of $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits of broadcast for sharing $\mathcal{O}(n)$ values simultaneously. Each party broadcasts at most $\mathcal{O}(n \log n)$ bits.*

Before showing full simulatability, we have the following two claims regarding the protocol: validity and binding. We will use those claims in the proof of theorem 4.4 when we show full simulation.

**Claim 4.5** (Validity)**.** *If the dealer is honest, then the protocol eventually terminates, and each honest party $P_i$ output $S(\mathbf{x}, i), S(i, \mathbf{y})$, where $S(\mathbf{x}, \mathbf{y})$ is the input of the dealer.*

*Proof.* Eventually, each honest $P_j$ broadcast $\mathsf{Good}(j, \ell)$ for all honest parties $\ell \notin I$. We show that the dealer must eventually broadcast $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$. Consider time $T$ in which all $\mathsf{Good}$ messages between pairs of honest parties were received by the dealer. There are two cases to consider:

**Case I – The dealer broadcast $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ before time $T$.** In that case, the dealer terminates before time $T$. Claim 4.6 below shows that all honest parties output shares that lie on the same bivariate polynomial. Moreover, this polynomial is determined by the set of honest parties in $\mathsf{C}$, which is the bivariate polynomial $S(\mathbf{x}, \mathbf{y})$ that the dealer holds.

**Case II – The dealer did not broadcast $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ before time $T$.** At time $T$, the dealer is guaranteed to find an $(n, t)$-star $(\mathsf{C}, \mathsf{D})$ such that $|\mathsf{C}| \geq n - 2t \geq 2t + 1$, and $|\mathsf{D}| \geq n - t \geq 3t + 1$ (see Claim 3.6). Moreover, $\mathsf{C}$ contains at least $2t + 1$ honest parties. Since $\mathsf{G}$ is defined as the set of parties that agree with at least $2t + 1$ in $\mathsf{C}$, we get that the set $\mathsf{G}$ must contain all honest parties. Moreover, the set $\mathsf{F}$ is defined as the set of parties that agree with at least $3t + 1$ parties in $\mathsf{G}$. Since $\mathsf{G}$ contains all honest parties (i.e., at least $3t + 1$ parties), all honest parties are part of $\mathsf{F}$. We conclude that the dealer finds and eventually broadcasts $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$.

All honest parties will eventually receive the shares from the dealer, and each pair of honest parties $P_k, P_\ell$ will eventually broadcast the messages $\mathsf{Good}(k,\ell)$ and $\mathsf{Good}(\ell,k)$. Therefore, the graph of the dealer $G$, eventually contains a clique of all honest parties (i.e., of $3t+1$). Once this occurs, each honest party is both in $\mathsf{G}$ and $\mathsf{F}$, and therefore output the same shares as received from the dealer. Since the dealer chose all polynomials to lie on the same bivariate polynomial $S(\mathbf{x},\mathbf{y})$, we have that $(f_i(\mathbf{x}), g_i(\mathbf{y})) = ((S(\mathbf{x},i), S(i,\mathbf{y}))$. □

**Claim 4.6** (Binding)**.** *When one honest party terminates, all honest parties will eventually terminate with shares on the same bivariate polynomial $S(\mathbf{x},\mathbf{y})$. The bivariate polynomial $S(\mathbf{x},\mathbf{y})$ is interpolated from the shares that the lexicographically first $t+1$ honest parties in $\mathsf{C}$ received from the dealer, where $\mathsf{C}$ is defined as the set that the dealer broadcasts.*

*Proof.* Consider the first honest party that terminates, say some $P_{j^*}$. If $P_{j^*}$ terminates, then the dealer must have broadcasted sets $(\mathsf{C},\mathsf{D},\mathsf{G},\mathsf{F})$, and $P_{j^*}$ must have validated the property in its local graph.

Let $H \subseteq \mathsf{C}$ be a set of some $t+1$ honest parties in $\mathsf{C}$. Since $|\mathsf{C}| \geq 2t+1$, there must exist such a set $H$. Consider the unique bivariate polynomial $S(\mathbf{x},\mathbf{y})$ satisfying $S(\mathbf{x},h) = f_h(x)$ for every $h \in H$, see Claim 3.8. At this point:

1. **For every honest $d \in \mathsf{D}$ it holds that $g_d(\mathbf{y}) = S(d,\mathbf{y})$.** Specifically, $g_d(\mathbf{y}), S(d,\mathbf{y})$ are two univariate polynomials of degree-$t$, and for every $h \in H$ it holds that $g_d(h) = f_h(d) = S(d,h)$, as otherwise the edge $(d,h)$ does not exist in $G$. Since the two polynomials agree on $t+1$ points, they are the same polynomial.
2. **For every honest $c \in \mathsf{C}$ it holds that $f_c(\mathbf{x}) = S(\mathbf{x},c)$.** This trivially holds for the parties in $H \subseteq \mathsf{C}$. We show that this also holds for the other honest parties in $\mathsf{C}$. Specifically, for every honest $c \in \mathsf{C}$ and $d \in \mathsf{D}$, $P_c$ and $P_d$ checked that $f_c(d) = g_d(c)$. Therefore, the two degree-$3t/2$ univariate polynomials, $f_c(\mathbf{x})$ and $S(\mathbf{x},c)$ agree on $2t+1$ points, and therefore it holds that $f_c(\mathbf{x}) = S(\mathbf{x},c)$.
3. **For every honest $j \in \mathsf{G}$ it holds that $g_j(\mathbf{y}) = S(\mathbf{y},j)$.** Each party in $\mathsf{G}$ agrees with at least $2t+1$ parties in $\mathsf{C}$. Therefore, $P_j$ agrees with at least $t+1$ honest parties in $\mathsf{C}$. Thus, the two polynomials $g_j(\mathbf{y})$ and $S(\mathbf{y},j)$ must agree on $t+1$ points (i.e., each such honest $c \in \mathsf{C}$ that agrees with $P_j$ exchanged $g_j(c)$ and $f_c(j)$ and it holds that $g_j(c) = f_c(j) = S(c,j)$). As a result, $g_j(\mathbf{y}) = S(\mathbf{y},j)$.
4. **For every honest $j \in \mathsf{F}$ it holds that $f_j(\mathbf{x}) = S(\mathbf{x},j)$.** Each party in $\mathsf{F}$ agrees with at least $3t+1$ parties in $\mathsf{G}$. Since $|\mathsf{G}| \geq 3t+1$, we get that $P_j$ agrees with at least $2t+1$ honest parties in $\mathsf{G}$. For each such $P_j$ and an honest $k \in \mathsf{G}$ it holds that $f_j(k) = g_k(j) = S(j,k)$. We get that two degree-$3t/2$ polynomials $f_j(\mathbf{x})$ and $S(\mathbf{x},j)$ must agree on $2t+1$ points, and therefore $f_j(\mathbf{x}) = S(\mathbf{x},j)$.

Since all $\mathsf{Good}$ messages are broadcasted, each honest party will eventually see the same edges as the first honest party that terminated, $P_{j^*}$, and the set $(\mathsf{C},\mathsf{D},\mathsf{G},\mathsf{F})$ will also be validated by each one of the honest parties.

We get that all honest parties $j \in \mathsf{F}$ hold a polynomial $f_j(\mathbf{x}) = S(\mathbf{x},j)$. Moreover, all honest parties $j \in \mathsf{G}$ holds a polynomial $g_j(\mathbf{y}) = S(j,\mathbf{y})$. If $j \in \mathsf{G} \cap \mathsf{F}$, then it has both $(f_j(\mathbf{x}), g_j(\mathbf{y}))$ and it can just terminate. For parties that do not hold both polynomials:

1. If $j \notin \mathsf{G}$, then it considers all the points that it received from parties in $\mathsf{F}$ in Step 2b. Those points lie on the polynomial $S(j,\mathbf{y})$. Since $\mathsf{G}$ contains $3t+1$ parties, it contains at least

$2t+1$ honest parties. Moreover, $P_j$ might receive at most $t$ incorrect points. Therefore, the decoding algorithm must eventually have a unique decoding, to the polynomial $S(j,\mathbf{y})$. Therefore, $P_j$ eventually obtains $g_j(y) = S(j,\mathbf{y})$. At that point it sends $(\mathsf{reconstruct}, j, g_j(k))$ for every $k \notin \mathsf{F}$, where $g_j(k) = S(j,k)$.

2. If $j \notin \mathsf{F}$, then it considers all the points from parties in $\mathsf{G}$ as received from Step 2b, and in addition it considers the points from parties not in $\mathsf{G}$, after they reconstruct their correct $g$ polynomial. As a result, $P_j$ is guaranteed to eventually receive $3t+1$ correct points on the polynomial $S(\mathbf{x}, j)$, and it would have at most $t$ incorrect points. Reed Solomon decoding is guaranteed to succeed.

We conclude that eventually, all honest parties terminate and output shares $(f_i(\mathbf{x}), g_i(\mathbf{y})) = (S(\mathbf{x},i), S(i,\mathbf{y}))$. □

*Proof.* We separate the analysis to the case of an honest dealer and a corrupted dealer.

**Case I – the case of an honest dealer.** We present the simulator $\mathcal{S}$:

1. Upon activation, the simulator invokes the adversary $\mathcal{A}$.
2. The simulator receives from the functionality the shares $(f_i(\mathbf{x}), g_i(\mathbf{y})) = S(\mathbf{x},i), S(i,\mathbf{y})$ for every $i \in I$. Moreover, it receives requests from the router to deliver the outputs of the functionality to the honest parties. The simulator will allow to deliver the outputs as the protocol proceed.
3. $\mathcal{S}$ sends to the adversary $\mathcal{A}$ the message $(\mathsf{shares}, D, i, f_i(\mathbf{x}), g_i(\mathbf{y}))$ for every $i \in I$ as coming from the dealer.
4. Send to the adversary $\mathcal{A}$ all delivery requests $(\mathsf{shares}, D, j)$ for $j \notin I$ as coming from the router.
5. Once $\mathcal{A}$ delivers $(\mathsf{shares}, D, j)$, simulate $P_j$ sending to each corrupted party $P_i$ the message $(\mathsf{exchange}, j, i, f_j(i), g_j(i)))$ where $(f_j(i), g_j(i)) = (S(j,i), S(i,j)) = (g_i(j), f_i(j))$. Moreover, simulate $P_j$ sending $(\mathsf{exchange}, j, k)$ for every $k \notin I$.
6. Once $\mathcal{A}$ delivers the message $(\mathsf{exchange}, j, k)$, simulate $P_k$ broadcasting $\mathsf{Good}(k, j)$.
7. Once the adversary sends $(\mathsf{exchange}, i, j, u_{i,j}, v_{i,j})$, verify that $u_{i,j} = f_i(j)$ and $v_{i,j} = g_i(j)$. If so (and $P_j$ already received its shares from the dealer), simulate $P_j$ broadcasting $\mathsf{Good}(j, i)$.
8. Initialize an undirected graph $G$ over nodes $[n]$. For each message $\mathsf{Good}(j,k)$ broadcasted (by the adversary or a simulated honest party), if the message $\mathsf{Good}(k,j)$ was also delivered to the dealer, then add the edge $(j,k)$ to $G$. Check whether the graph $G$ contains sets $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ as in the protocol. If so, simulate the dealer broadcasting $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$.
9. For each party $P_j$, simulate the following:
   (a) $P_j$ initializes a graph $G_i$. Once $\mathcal{A}$ delivers the broadcasted messages $\mathsf{Good}(j,k)$ and $\mathsf{Good}(k,j)$, add the edge $(k,j)$ to $G_i$. Moreover, once the adversary delivers the broadcasted message $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ to $P_j$, check that $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ is valid in $G_i$. If not, continue to listen to addition $\mathsf{Good}$ messages.
   (b) Once $(\mathsf{C}, \mathsf{D}, \mathsf{G}, \mathsf{F})$ is valid for $P_j$, if $j \in \mathsf{G} \cap \mathsf{F}$, then allow the router to deliver the output of party $P_j$ from the ideal functionality the the dummy party.
   (c) If $j \notin \mathsf{G}$, then wait until the adversary delivers additional $(\mathsf{exchange}, k, j)$ messages for honest $k \notin I$. Moreover, the simulator can identify the number of incorrect shares that the adversary sent to $P_j$ via $(\mathsf{exchange}, i, j, \cdot, \cdot)$. Let $t' \leq t$ be that number of incorrect

shares. Once the adversary allows receiving $t' + t + 1+$ shares from $\mathsf{F} \setminus I$, then $P_j$ can reconstruct its polynomial $g_j(\mathbf{y})$. Simulate $P_j$ sending $(\mathsf{reconstruct}, j, k, g_j(k))$ for each honest party $P_k$ for $k \notin \mathsf{F}$ by giving the adversary $\mathcal{A}$ the messages $(\mathsf{reconstruct}, j, k)$ as coming from the router.

(d) If $j \notin \mathsf{F}$, then continue to listen to all messages $(\mathsf{exchange}, k, i)$ delivered from the adversary, and all $(\mathsf{reconstruct}, \ell, i)$. Moreover, the simulate can identify the number of incorrect sub-shares that the adversary had sent $P_j$ via $(\mathsf{exchange}, i, j)$ messages. Let $t'$ be the number of incorrect shares. Once the adversary delivers to $P_j$ exactly $t' + 3t/2 + 1$ shares from honest parties, then $P_j$ can reconstruct $f_j(\mathbf{x})$.

(e) Once $P_j$ can reconstruct (if needed) $f_j(\mathbf{x})$ and $g_j(\mathbf{y})$, the simulator $\mathcal{S}$ allows the router in the ideal model to deliver the output of $P_j$ from the functionality to $P_j$.

10. The adversary must eventually allow delivering all messages, in then each simulated honest party $P_j$ eventually delivers all messages sent from the functionality to the honest parties in the ideal execution.

We now show that the view of the environment $\mathcal{Z}$ is the same in both executions. From inspection, it is easy to see that the view of the adversary $\mathcal{A}$ is the same in both execution. In particular, this is due to the fact that the protocol and the simulation are deterministic, and that the messages of the honest parties to the corrupted parties can be simulated from the shares of the corrupted parties. Our goal is to show now that the outputs of the parties is the same between the two executions (and in particular, each party receives its output at the same "time" in both executions). Regarding the stage in which each party receives its output, we remark that this is determined by the adversary and how it instructs the router in the real world, or the simulated router in the ideal world (in which the simulator then forwards the request to deliver to the real router in the ideal world).

Therefore, it suffices to show that the outputs of the honest parties are the same in both executions. Claim 4.5 shows that in the real world, each honest party $P_j$ eventually receives shares $S(\mathbf{x}, j), S(j, \mathbf{y})$, while the scheduling is determined by the adversary. In the ideal, the trusted party delivers to each honest party $P_j$ the shares $S(\mathbf{x}, j), S(j, \mathbf{y})$. The adversary $\mathcal{A}$ decides on the scheduling (via $\mathcal{S}$ and the router), but since the view of $\mathcal{A}$ is exactly the same in the real and ideal, the scheduling is exactly the same.

**Case II - the case of a corrupted dealer.** We provide the simulator $\mathcal{S}$:

1. Upon activation, the simulator activates the adversary $\mathcal{A}$.
2. Since the protocol is deterministic and the honest parties have no inputs, the simulator can simulate all honest parties (and the router) in an execution with $\mathcal{A}$. This means that it listens to the messages $\mathcal{A}$ sends to the honest parties, and it can also simulate the messages between honest parties (where $\mathcal{A}$ receives notifications from the simulated router). Note that the simulation might not terminate.
3. When the first honest party $P_{j^*}$ terminates, then all honest parties will eventually terminate (see Claim 4.6). Consider the set $\mathsf{C}$ that dealer has broadcasted in the simulated execution. Interpolate the polynomial $S(\mathbf{x}, \mathbf{y})$ according to the lexicographically first $t+1$ honest parties in $\mathsf{C}$, similarly to the binding property in the proof of Claim 4.6. Send to the functionality the polynomial $S(\mathbf{x}, \mathbf{y})$. It is guaranteed that $S$ has degree at most $2t$ in $\mathbf{x}$ and degree at most $t$ in $\mathbf{y}$.

4. The functionality delivers to each honest party $P_j$. The simulator instructs the router to deliver the output to $P_{j^*}$.
5. It continues to simulate the protocol to the adversary, where whenever a simulated honest party $P_j$ obtains an output in the simulated protocol, then the simulator $\mathcal{S}$ delivers to the router to send the output to the dummy party $P_j$ in the ideal.
6. Eventually, all honest parties will receive output in the simulated execution. The simulator then terminates.

Clearly, the view of the adversary is exactly the same in the real and ideal executions, as the honest parties have no inputs and are deterministic. If some honest party terminates, then as follows from the proof of Claim 4.6, all honest parties eventually output shares that all lie on a bivariate polynomial that is defined from the lexicographically first $t+1$ honest parties in $\mathsf{C}$. We note that the environment $\mathcal{Z}$ sees that the outputs at the same activation in the real and ideal. Specifically, upon the activation in which the first honest party $P_{j^*}$ terminates, in the ideal execution the simulator extracts the input to send to the trusted party, and it delivers the output to $P_{j^*}$ in the ideal. Since the protocol proceeds in the same way in both executions, parties receive their outputs at the same activations. □

# 5 Verifying Product Relation

In this section, we show how to realize the product-relation verification functionality. Assume that a dealer owns and preshares $\mathcal{O}(n^2)$ Shamir-sharings of triples, such that each of the triples are supposed to satisfy a product relation. That is, for a triple $(a, b, c)$, $c$ must be equal to $ab$. The parties input shares of those shared triples to the functionality, and the functionality checks that shares define triples satisfying product relations.

**Functionality 5.1: Verifying Product Relation**

The functionality is parameterized by a set of corrupted parties $I \subseteq [n]$.

1. Let $U = [(t/2+1)^2]$. Each party $P_j$ sends to the functionality a set of points $(a_j^u, b_j^u, c_j^u)_{u \in U}$.
2. For every $u \in U$, the functionality reconstructs the unique degree-$t$ univariate polynomials $A^u(\mathbf{x}), B^u(\mathbf{x}), C^u(\mathbf{x})$ satisfying

$$A^u(j) = a_j^u, \qquad B^u(j) = b_j^u, \qquad C^u(j) = c_j^u \ .$$

   If the dealer is honest, then the dealer also sends $A^u(\mathbf{x}), B^u(\mathbf{x}), C^u(\mathbf{x})$.
3. If the dealer is honest, then give the adversary the shares $A^u(i), B^u(i)$ and $C^u(i)$ for every $i \in I$. If the dealer is corrupted, then give the adversary the reconstructed $A^u(\mathbf{x}), B^u(\mathbf{x}), C^u(\mathbf{x})$.
4. The functionality verifies that for every $u \in U$ it holds that

$$A^u(0) \cdot B^u(0) = C^u(0).$$

   If yes, then it sends $\mathsf{OK}$ to all parties and halts. Otherwise, the functionality never terminates.

### 5.1 Trivariate Polynomial Verification – Functionality

We overviewed the ZK property in Section 2.4. Towards realizing Functionality 5.1, we introduce an aiding functionality. To recall, we write $U = V \times V$ where $V = \{0, \ldots, t/2\}$, and the dealer embeds the coefficients of the polynomials $A^{(\beta,\gamma)}, B^{(\beta,\gamma)}, C^{(\beta,\gamma)}$ in four trivariate polynomials. We abstract some computations that the parties perform to improve readability by considering general predicates. Specifically:

1. Each party $P_i$ also receives from the dealer the bivariate polynomial $T_i(\mathbf{x}, \mathbf{z})$ as part of its share; each party $P_j$ computes from its share $\mathsf{share}_j$ a univariate polynomial that is supposed to be $T_i(\mathbf{x}, j)$ by applying some linear combination over its shares. We abstract this linear combination as "linear circuit", formally defined below. Note that the same computation that the dealer performs on the trivariate shares to obtain $T_i(\mathbf{x}, \mathbf{y})$, each party performs on its share $\mathsf{share}_j$ to obtain $T_i(\mathbf{x}, j)$.
2. Each party $P_i$ also checks that for every $(\beta, \gamma)$ it holds that $T_i(-\beta, -\gamma) = a_i^{(\beta,\gamma)} \cdot b_i^{(\beta,\gamma)} - c_i^{(\beta,\gamma)}$. We abstract this check as an "external validity" predicate that $P_i$ enters as input.

**Linear circuits.** We consider a circuit $L_j$ that receives as an input a bivariate polynomial $F(\mathbf{x}, \mathbf{y})$ and it has the following structure:

1. Evaluate $F(\mathbf{x}, \mathbf{y})$ on several constants $\mathbf{y} = \alpha_1, \ldots, \alpha_k$ in the field. The results are univariate polynomials $f_1(\mathbf{x}) = F(\mathbf{x}, \alpha_1), \ldots, f_k(\mathbf{x}) = F(\mathbf{x}, \alpha_k)$.
2. Output the univariate polynomial $f^{L_j}(\mathbf{x}) = \sum_{\ell=1}^{k} \lambda_\ell \cdot f_\ell(\mathbf{x})$ for some constants $\lambda_1, \ldots, \lambda_k$. That is, output a fixed linear combination of $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$.

We write $f^{L_j(F)}(\mathbf{x}) = L_j(F(\mathbf{x}, \mathbf{y}))$. We also evaluate the circuit $L_j$ on a trivariate polynomial $\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})$. In that case:

1. Evaluate $\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ on the same constants $\mathbf{y} = \alpha_1, \ldots, \alpha_k$. The results are bivariate polynomials $(F_1(\mathbf{x}, \mathbf{z}), \ldots, F_k(\mathbf{x}, \mathbf{z})) = (\mathbf{F}(\mathbf{x}, \alpha_1, \mathbf{z}), \ldots, \mathbf{F}(\mathbf{x}, \alpha_k, \mathbf{z}))$.
2. Output the bivariate polynomial which is a fixed linear combination $F^{L_j}(\mathbf{x}, \mathbf{z}) := \sum_{\ell=1}^{k} \lambda_\ell \cdot F_\ell(\mathbf{x}, \mathbf{z})$.

We write $F^{L_j(\mathbf{F})}(\mathbf{x}, \mathbf{y}) = L_j(\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z}))$. For every $i \in [n]$, consider $F_i(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x}, \mathbf{y}, i)$, and let $f^{L_j(F_i)}(\mathbf{x}) = L_j(F_i(\mathbf{x}, \mathbf{y}))$. Then clearly it holds that

$$f^{L_j(F_i)}(\mathbf{x}) = L_j(F_i(\mathbf{x}, \mathbf{y})) = L_j(\mathbf{F}(\mathbf{x}, \mathbf{y}, i)) = F^{L_j(\mathbf{F})}(\mathbf{x}, i) \ .$$

The specific linear circuit that we use is given below in Circuit 5.2.

**External validity.** The predicate $\mathsf{ExternalValidity}_j$ receives as input the share of $P_j$ and outputs 0 or 1. The exact predicate that we use is given in Algorithm 5.3.

**Linear Circuit 5.2** (The circuit $L_i$:)**.**

- **Input:** *Trivariate polynomials* $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4$.
  1. *For* $r \in [1, \ldots, 4]$*, evaluate* $\mathbf{S}_r(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *on the constants* $\mathbf{y} = -1, \ldots, -t/2$.
  2. *Obtain* $\mathbf{S}_r(\mathbf{x}, -k, \mathbf{z})$ *for* $k \in [1, \ldots, t/2]$ *and* $r \in [1, \ldots, 4]$.
  3. *Define* $T_i(\mathbf{x}, \mathbf{z}) := \sum_{r=1}^{4} \sum_{k=1}^{t/2} i^{(r-1)\cdot(t/2)+k} \cdot \mathbf{S}_r(\mathbf{x}, -k, \mathbf{z})$.

- **Output:** *The bivariate polynomial* $T_i(\mathbf{x}, \mathbf{z})$.

**Algorithm 5.3: The predicate** $\mathsf{ExternalValidity}_i$

- **Input:** The share $\mathsf{share}_i$ of $P_i$, which consists of shares on each one of the polynomials $\mathbf{S}_1, \ldots, \mathbf{S}_4$ and bivariate polynomial $T_i(\mathbf{x}, \mathbf{z})$.
- **Parameters:** For every $(\beta, \gamma) \in V \times V$ the algorithm is hardwired with values $a_i^{(\beta,\gamma)}, b_i^{(\beta,\gamma)}, c_i^{(\beta,\gamma)} \in \mathbb{F}$.
- **The algorithm:** Output 1 iff for every $\beta, \gamma \in V$ it holds that:

$$T_i(-\beta, -\gamma) \;=\; a_i^{(\beta,\gamma)} \cdot b_i^{(\beta,\gamma)} - c_i^{(\beta,\gamma)} \;.$$

We are now ready to provide the functionality which the product-relation verification protocol uses as its main building block:

**Functionality 5.4: Trivariate Polynomial Verification**

The functionality is parameterized by (1) The set of corrupted parties $I \subseteq [n]$; (2) Some linear circuits $L_1, \ldots, L_n$ as defined above. The functionality works as follows:

1. The dealer sends four trivariate polynomials $\mathbf{S}_1(\mathbf{x}, \mathbf{y}, \mathbf{z}), \ldots, \mathbf{S}_4(\mathbf{x}, \mathbf{y}, \mathbf{z})$.
2. Define the share of each party $P_i$ to be

$$\mathsf{share}_i = \Big( (\mathbf{S}_r(\mathbf{x}, \mathbf{y}, i), \mathbf{S}_r(\mathbf{x}, i, \mathbf{z}), \mathbf{S}_r(\mathbf{x}, \mathbf{y}, i))_{r \in [4]}, L_i(\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4) \Big) \;.$$

   If the dealer is honest, then for every $i \in I$ send $P_i$ the share $\mathsf{share}_i$.
3. The honest parties send to the functionality their external validity predicates $\mathsf{ExternalValidity}_j$ to the functionality. Each $\mathsf{ExternalValidity}_j$ takes as input $\mathsf{share}_j$ and outputs 0 or 1.
4. If the dealer is corrupted, then send $(\mathsf{ExternalValidity}_j)_{j \notin I}$ to the adversary.
5. If each one of the four trivariate polynomials $\mathbf{S}_1, \ldots, \mathbf{S}_4$ is of degree $t + t/2$ in each one of the three variables, and $\mathsf{ExternalValidity}_j(\mathsf{share}_j) = 1$ holds for at least $2t + 1$ honest parties, then send $\mathsf{OK}$ to all parties and halt. Otherwise, the functionality does not terminate.

## 5.2 Verifying Product Relation using Trivariate Polynomial

We now proceed to show how to implement Functionality 5.1, using the trivariate sharing as a building block (i.e., Functionality 5.4). We then provide the theorem statement which we prove subsequently.

**Protocol 5.5: Verifying the Product Relation**

- **Input:** The dealer holds $A^{(\beta,\gamma)}(\mathbf{x}), B^{(\beta,\gamma)}(\mathbf{x}), C^{(\beta,\gamma)}(\mathbf{x})$ for every $(\beta, \gamma) \in V \times V$. Each party $P_i$ holds the points $A^{(\beta,\gamma)}(i), B^{(\beta,\gamma)}(i), C^{(\beta,\gamma)}(i)$ for every $(\beta, \gamma) \in V \times V$.
- **The protocol:**
  1. The dealer computes $E^{(\beta,\gamma)}(\mathbf{x}) = A^{(\beta,\gamma)}(\mathbf{x}) \cdot B^{(\beta,\gamma)}(\mathbf{x}) - C^{(\beta,\gamma)}(\mathbf{x})$ for every $(\beta, \gamma) \in V \times V$ and define the coefficients $e_1^{(\beta,\gamma)}, \ldots, e_{2t}^{(\beta,\gamma)}$.

2. The dealer chooses four trivariate polynomials $\mathbf{S}_1, \ldots, \mathbf{S}_4$ of degree $t + t/2$ in all three variables uniformly at random while embedding the coefficients $e_1^{(\beta,\gamma)}, \ldots, e_{2t}^{(\beta,\gamma)}$ as described in the text in Section 2.4.
3. The parties invoke Functionality 5.4 where the dealer inputs $\mathbf{S}_1, \ldots, \mathbf{S}_4$ and each party $P_i$ (eventually) inputs its private $\mathsf{ExternalValidity}_i$ as defined in Algorithm 5.3. The functionality is parameterized by the linear circuits $L_1, \ldots, L_n$, each is defined as in Circuit 5.2.
4. Each party $P_i$: upon receiving an output $\mathsf{OK}$ from Functionality 5.4, then terminate and output $\mathsf{OK}$.

---

**Theorem 5.6.** *Let $n \geq 4t + 1$. Protocol 5.5 securely computes Functionality 5.1 in the presence of a malicious adversary controlling at most $t$ parties. It requires communication of $\mathcal{O}(n^3 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits of broadcast. Each party broadcasts at most $\mathcal{O}(n \log n)$ bits.*

*Proof.* We show separately the case of an honest dealer and of a corrupted dealer.

**The case of an honest dealer.**

1. The simulator invokes the adversary $\mathcal{A}$.
2. The simulator receives from the functionality the shares $A^u(i), B^u(i), C^u(i)$ for every $i \in I$.
3. It chooses trivariate polynomials $\mathbf{S}_1(\mathbf{x}, \mathbf{y}, \mathbf{z}), \ldots, \mathbf{S}_4(\mathbf{x}, \mathbf{y}, \mathbf{z})$ uniformly at random under the constraints that for every $i \in I$:

$$\sum_{r=1}^{4} \sum_{k=1}^{t/2} i^{(r-1)\cdot(t/2)+k} \cdot \mathbf{S}_r(-\beta, -k, -\gamma) = A^{(\beta,\gamma)}(i) \cdot B^{(\beta,\gamma)}(i) - C^{(\beta,\gamma)}(i) \ .$$

   Note that this imposes $(t/2+1)^2 \cdot |I| \leq (t/2+1)^2 \cdot t$ total constraints, while in each polynomial we pack up to $(t/2+1)^3$ secrets.
4. Simulate the invocation of Functionality 5.4: for every $i \in I$, give the adversary

$$\mathsf{share}_i = \big((\mathbf{S}_r(\mathbf{x}, \mathbf{y}, i), \mathbf{S}_r(\mathbf{x}, i, \mathbf{z}), \mathbf{S}_r(\mathbf{x}, \mathbf{y}, i))_{r\in[4]}, L_i(\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4)\big)$$

5. It simulates the functionality returning $\mathsf{OK}$. Whenever the adversary delivers the message $\mathsf{OK}$ to party $P_j$ in the simulated protocol, the simulator delivers the output of Functionality 5.1 to party $P_j$ in the ideal world.

We first show that the output of the honest parties is identical in the real and ideal executions. In the real execution, since the dealer is honest, it always holds valid polynomials $A^{(\beta,\gamma)}(\mathbf{x}), B^{(\beta,\gamma)}(\mathbf{x}), C^{(\beta,\gamma)}(\mathbf{x})$ each of degree-$t$ such that $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$ holds for every $(\beta, \gamma) \in V \times V$. Moreover, it chooses the the trivariate polynomials $\mathbf{S}_1, \ldots, \mathbf{S}_4$ such that they satisfy the conditions of Functionality 5.4. Specifically, $\mathbf{S}_1, \ldots, \mathbf{S}_4$ are of the degree $t + t/2$ in each of the three variables and $\mathsf{ExternalValidity}_j(\mathsf{share}_j) = 1$ holds for all the honest parties. Hence, the functionality always returns $\mathsf{OK}$ to all the parties. Consequently, honest parties always output $\mathsf{OK}$ in the real execution. In the ideal execution, all honest parties hold shares on the valid degree-$t$ polynomials $A^{(\beta,\gamma)}(\mathbf{x}), B^{(\beta,\gamma)}(\mathbf{x}), C^{(\beta,\gamma)}(\mathbf{x})$ for every $(\beta, \gamma) \in V \times V$ shared by the dealer, with which

they invoke the Functionality 5.1. Since the dealer is honest, for each $(\beta,\gamma) \in V \times V$ it holds that $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$. The output of the functionality is fully determined by the honest parties' inputs and hence it always returns $\mathsf{OK}$ to all the honest parties. It thus remains to show that the view of the adversary is identically distributed in the real and ideal executions.

Below, we prove the claim for a single trivariate polynomial to ease the notations. However, the claim easily extends to the case of four trivariate polynomials as used in the actual protocol.

**Claim 5.7.** *Let $E_1$ and $E_2$ be any arbitrary sets of $(t/2+1) \times t/2 \times (t/2+1)$ field elements each and let $I \subset [n]$ be a set of cardinality at most $t$. Then, under the constraint that for each $i \in I$ and $(\beta,\gamma) \in V \times V$, $\sum_{k=1}^{t/2} i^k \cdot E_1(\beta,k,\gamma) = \sum_{k=1}^{t/2} i^k \cdot E_2(\beta,k,\gamma)$, the following two distributions are identical:*

**Process I:** *Choose a random trivariate polynomial $\mathbf{F}(\mathbf{x},\mathbf{y},\mathbf{z})$ such that $\mathbf{F}(-\beta,-k,-\gamma) = E_1(\beta,k,\gamma)$ for every $\beta,\gamma \in \{0,\ldots,t/2\}$ and every $k \in \{1,\ldots,t/2\}$. Output $(i, \mathbf{F}(\mathbf{x},\mathbf{y},i), \mathbf{F}(\mathbf{x},i,\mathbf{z}), \mathbf{F}(i,\mathbf{y},\mathbf{z}), \sum_{k=1}^{t/2} i^k \cdot \mathbf{F}(\mathbf{x},-k,\mathbf{z}))$ for every $i \in I$.*

**Process II:** *Choose a random trivariate polynomial $\mathbf{F}'(\mathbf{x},\mathbf{y},\mathbf{z})$ such that $\mathbf{F}'(-\beta,-k,-\gamma) = E_2(\beta,k,\gamma)$ for every $\beta,\gamma \in \{0,\ldots,t/2\}$ and every $k \in \{1,\ldots,t/2\}$. Output $(i, \mathbf{F}'(\mathbf{x},\mathbf{y},i), \mathbf{F}'(\mathbf{x},i,\mathbf{z}), \mathbf{F}'(i,\mathbf{y},\mathbf{z}), \sum_{k=1}^{t/2} i^k \cdot \mathbf{F}'(\mathbf{x},-k,\mathbf{z}))$ for every $i \in I$.*

*Proof.* We show that the probability distributions $\{\{(i, \mathbf{F}(\mathbf{x},\mathbf{y},i), \mathbf{F}(\mathbf{x},i,\mathbf{z}), \mathbf{F}(i,\mathbf{y},\mathbf{z}), T_i(\mathbf{x},\mathbf{z}))\}_{i\in I}\}$ corresponding to **Process I** and $\{\{(i, \mathbf{F}'(\mathbf{x},\mathbf{y},i), \mathbf{F}'(\mathbf{x},i,\mathbf{z}), \mathbf{F}'(i,\mathbf{y},\mathbf{z}), T'_i(\mathbf{x},\mathbf{z}))\}_{i\in I}\}$ corresponding to **Process II** are identical. Towards that, let $\mathbb{S}$ and $\mathbb{S}'$ denote the probability ensembles corresponding to **Process I** and **Process II** respectively. We thus show that $\mathbb{S} \equiv \mathbb{S}'$. For this, we show that given any set of tuple of bivariate polynomials with degree $3t/2$ in both variables, say $Z = \{Q_i(\mathbf{x},\mathbf{y}), W_i(\mathbf{x},\mathbf{z}), R_i(\mathbf{y},\mathbf{z}), T_i(\mathbf{x},\mathbf{z})\}_{i\in I}$ that satisfy Definition 5.8, the number of trivariate polynomials in support of $\mathbb{S}$ that are consistent with $Z$ are the same as the number of polynomials in support of $\mathbb{S}'$.

Note that if the set $Z$ does not satisfy Definition 5.8, then there does not exist any trivariate polynomial that is in support of $\mathbb{S}$ or $\mathbb{S}'$. Now, consider a set $Z$ that satisfies Definition 5.8. For simplicity, consider the case when $|I| = t$. Choose a set, say $E$, of $(t/2+1)^2$ elements selected uniformly at random from $\mathbb{F}$. Note that $E_1 \cup E$ together with $Z$ defines a unique trivariate polynomial $S(\mathbf{x},\mathbf{y},\mathbf{z})$ as follows. For each $k \in \{0,\ldots,t/2\}$, construct $W_{-k}(\mathbf{x},\mathbf{z})$ of degree-$3t/2$ in each variable such that $W_0(-\beta,-\gamma) = E(\beta,\gamma)$ and $W_{-k}(-\beta,-\gamma) = E_1(\beta,k,\gamma)$. Note that this defines $(t/2+1)^2$ points on each $W_{-k}(\mathbf{x},\mathbf{y})$. Moreover, for every $i \in I$, we set $W_{-k}(\mathbf{x},i) = Q_i(\mathbf{x},-k)$ and $W_{-k}(i,\mathbf{z}) = R_i(-k,\mathbf{z})$. This defines $2t(t+1)$ more points on each $W_{-k}(\mathbf{x},\mathbf{z})$. Thus, in total we have $(t+t/2+1)^2$ points defined on each $W_{-k}(\mathbf{x},\mathbf{z})$ which defines the polynomial completely. Given the $t$ polynomials $W_i(\mathbf{x},\mathbf{z})$ for every $i \in I$ and $t/2+1$ polynomials $W_{-k}(\mathbf{x},\mathbf{y})$ for every $k \in \{0,\ldots,t/2\}$, these define a unique polynomial $\mathbf{F}(\mathbf{x},\mathbf{y},\mathbf{z})$ due to Claim 3.9. The same argument holds true for the case of the set $E_2 \cup E$ and the corresponding unique polynomial $\mathbf{F}'(\mathbf{x},\mathbf{y},\mathbf{z})$. Hence, the support for $\mathbb{S}$ and $\mathbb{S}'$ is equal. For the case when $|I| < t$, we can view process I (respectively process II) as first choosing $t - |I|$ polynomials $W_j(\mathbf{x},\mathbf{z})$ (respectively $W'_j(\mathbf{x},\mathbf{z})$) for some $j \notin I$ uniformly at random under the constraint that $Z$ together with $\{W_j(\mathbf{x},\mathbf{z})\}$ (respectively $\{W'_j(\mathbf{x},\mathbf{z})\}$) satisfy Definition 5.8. Following this, the analysis is same as the case with $|I| = t$. □

**The case of a corrupted dealer.**

1. The simulator invokes the adversary $\mathcal{A}$.

2. The simulator receives from the functionality the polynomials $A^{(\beta,\gamma)}(\mathbf{x}), B^{(\beta,\gamma)}(\mathbf{x})$ and $C^{(\beta,\gamma)}(\mathbf{x})$ for every $(\beta,\gamma) \in V \times V$.
3. It simulates the invocation of Functionality 5.4: It receives from the adversary four trivariate polynomials $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4$.
4. It generates the external validity functions $\mathsf{ExternalValidity}_j$ for every $j \notin I$ with $a_j^{(\beta,\gamma)} = A^{(\beta,\gamma)}(j)$, $b_j^{(\beta,\gamma)} = B^{(\beta,\gamma)}(j)$ and $c_j^{(\beta,\gamma)} = C^{(\beta,\gamma)}(j)$. It sends to the adversary the predicates $\mathsf{ExternalValidity}_j$ for every $j \notin I$ as in Functionality 5.4.
5. Define $T_i(\mathbf{x},\mathbf{z}) = L_i(\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4)$ where $L_i$ is defined as in Circuit 5.2.
6. Check that each polynomial $\mathbf{S}_r$ for $r \in [4]$ is of degree $t + t/2$ in each one of the three variables. Moreover, if $\mathsf{ExternalValidity}_j$ holds for at least $2t+1$ honest parties, then simulate Functionality 5.4 sending output $\mathsf{OK}$. Otherwise, do not terminate.
7. Whenever the adversary delivers the message $\mathsf{OK}$ to party $P_j$ in the simulated protocol, the simulator delivers the output of Functionality 5.1 to party $P_j$ in the ideal world.

Clearly the view of the adversary is the same in both executions. We now prove that if Functionality 5.4 returns $\mathsf{OK}$, then Functionality 5.1 also return $\mathsf{OK}$. That is, we claim that once $\mathcal{A}$ sent polynomials $\mathbf{S}_r$ for $r \in [4]$ for which $\mathsf{ExternalValidity}_j$ holds for at least $2t+1$ parties, then it holds that $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$ for every $(\beta,\gamma) \in V \times V$. To see that, fix some $(\beta,\gamma)$. Consider the following polynomial:

$$\begin{aligned} Y_{(\beta,\gamma)}(\mathbf{x}) &:= \sum_{r=1}^{4} \sum_{k=1}^{t/2} \mathbf{x}^{(r-1)\cdot(t/2)+k} \cdot \mathbf{S}_r(-\beta, -k, -\gamma) \\ &= e_1\mathbf{x} + \ldots + e_{2t}\mathbf{x}^{2t} \end{aligned}$$

where each $e_\ell$ directly corresponds to some $\mathbf{S}_r(-\beta,-k,-\gamma)$. This is a univariate polynomial where the only variable is $\mathbf{x}$ and is of of degree $2t$. Moreover, $Y_{(\beta,\gamma)}(j) = T_j(-\beta,-\gamma)$, where $T_j(\mathbf{x},\mathbf{z}) = L_j(\mathbf{S}_1,\ldots,\mathbf{S}_4)$ as defined in Step 5 of the simulation. Furthermore, consider the polynomial:

$$E^{(\beta,\gamma)}(\mathbf{x}) = A^{(\beta,\gamma)}(\mathbf{x}) \cdot B^{(\beta,\gamma)}(\mathbf{x}) - C^{(\beta,\gamma)}(\mathbf{x}) \ .$$

This is also a univariate polynomial of degree-$2t$. From the external validity property, for at least $2t+1$ honest parties $J \subseteq [n]$ it holds that

$$E^{(\beta,\gamma)}(j) = a_j^{(\beta,\gamma)} \cdot b_j^{(\beta,\gamma)} - c_j^{(\beta,\gamma)} = T_j(-\beta,-\gamma) = Y_{(\beta,\gamma)}(j) \ .$$

Therefore, the two degree-$2t$ polynomials $E^{(\beta,\gamma)}(\mathbf{x}), Y_{(\beta,\gamma)}(\mathbf{x})$ agree. Since the constant term of the polynomial $Y_{(\beta,\gamma)}(\mathbf{x})$ is 0, we get that the constant term of $E^{(\beta,\gamma)}(\mathbf{x})$ is 0. Therefore it holds that

$$E^{(\beta,\gamma)}(0) = A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) - C^{(\beta,\gamma)}(0) = 0 \ ,$$

and therefore $A^{(\beta,\gamma)}(0) \cdot B^{(\beta,\gamma)}(0) = C^{(\beta,\gamma)}(0)$ for every $(\beta,\gamma) \in V \times V$. □

### 5.3 Trivariate Polynomial Verification – Protocol

In the remainder of this sub-section, we show how to implement Functionality 5.4. To ease notations, we show how to implement the functionality with general linear functions $L_1,\ldots,L_n$ and general

$\mathsf{ExternalValidity}_j$. Moreover, we assume that the dealer sends just one trivariate polynomial $\mathbf{S}$ instead of four; generalizing for the case of four polynomial is straightforward. This construction is essentially our asynchronous weak-binding trivariate secret sharing, for which we provided an extensive overview in Section 2.2.

**Definition 5.8.** *We say that the share that $P_i$ received from the dealer*

$$\begin{aligned}\mathsf{share}_i \;=\; & (Q_i(\mathbf{x},\mathbf{y}), \quad W_i(\mathbf{x},\mathbf{z}), \quad R_i(\mathbf{y},\mathbf{z}), \quad T_i(\mathbf{x},\mathbf{z})\ ) \\ & (=(\mathbf{S}(\mathbf{x},\mathbf{y},i), \quad \mathbf{S}(\mathbf{x},i,\mathbf{z}), \quad \mathbf{S}(i,\mathbf{y},\mathbf{z}), \quad L_i(\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})))\end{aligned}$$

*is consistent with an exchange sub-share message $m_{j\to i}$ that $P_j$ sends to $P_i$,*

$$m_{j\to i} := \left(\mathsf{exchange}, j, i, f_i^{Q_j}(\mathbf{x}), g_i^{Q_j}(\mathbf{y}), f_i^{W_j}(\mathbf{x}), g_i^{W_j}(\mathbf{z}), f_i^{R_j}(\mathbf{y}), g_i^{R_j}(\mathbf{z}), t^{L_i(Q_j)}(\mathbf{x})\right)\ ,$$

*denoted as* $\mathsf{consistent}(\mathsf{share}_i, m_{j\to i}) = 1$*, if the following conditions hold:*

$$\begin{aligned} f_i^{Q_j}(\mathbf{x}) &= W_i(\mathbf{x},j) \quad (=\mathbf{S}(\mathbf{x},i,j)), & g_i^{Q_j}(\mathbf{y}) &= R_i(\mathbf{y},j) \quad (=\mathbf{S}(i,\mathbf{y},j)),\\ f_i^{W_j}(\mathbf{x}) &= Q_i(\mathbf{x},j) \quad (=\mathbf{S}(\mathbf{x},j,i)), & g_i^{W_j}(\mathbf{z}) &= R_i(j,\mathbf{z}) \quad (=\mathbf{S}(i,j,\mathbf{z})),\\ f_i^{R_j}(\mathbf{y}) &= Q_i(j,\mathbf{y}) \quad (=\mathbf{S}(j,\mathbf{y},i)), & g_i^{R_j}(\mathbf{z}) &= W_i(j,\mathbf{z}) \quad (=\mathbf{S}(j,i,\mathbf{z}))\ , \end{aligned}$$

*and,*

$$t^{L_i(Q_j)}(\mathbf{x}) = T_i(\mathbf{x},j)$$

---

**Protocol 5.9: Trivariate Polynomial Verification**

---

**Input:** The input of the dealer is some trivariate polynomial $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$. The input of each party $P_j$ is some predicate $\mathsf{ExternalValidity}_j$.

**Public parameters:** The protocol is parameterized by linear circuits $L_1,\ldots,L_n$.

**The protocol:**

1. **(Share Distribution)** For each party $P_i$, the dealer sends the share:

$$(\mathsf{share},\ i,\ \mathbf{S}(\mathbf{x},\mathbf{y},i),\ \mathbf{S}(\mathbf{x},i,\mathbf{z}),\ \mathbf{S}(i,\mathbf{y},\mathbf{z}),\ L_i(\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z}))$$

2. **(Exchange sub-share)** Each party $P_i$:
   (a) Upon receiving $(\mathsf{share}, i, Q_i(\mathbf{x},\mathbf{y}), W_i(\mathbf{x},\mathbf{z}), R_i(\mathbf{y},\mathbf{z}), T_i(\mathbf{x},\mathbf{z}))$ from the dealer, check if: $\mathsf{ExternalValidity}_i\,(Q_i(\mathbf{x},\mathbf{y}), W_i(\mathbf{x},\mathbf{z}), R_i(\mathbf{y},\mathbf{z}), T_i(\mathbf{x},\mathbf{z})) = 1$.
   (b) If the above condition holds, then for every $P_j$ define the following seven polynomials:

$$\begin{aligned} f_j^{Q_i}(\mathbf{x}) &\stackrel{\text{def}}{=} Q_i(\mathbf{x},j) \quad (=\mathbf{S}(\mathbf{x},j,i))\ , & g_j^{Q_i}(\mathbf{y}) &\stackrel{\text{def}}{=} Q_i(j,\mathbf{y}) \quad (=\mathbf{S}(j,\mathbf{y},i))\ ,\\ f_j^{W_i}(\mathbf{x}) &\stackrel{\text{def}}{=} W_i(\mathbf{x},j) \quad (=\mathbf{S}(\mathbf{x},i,j))\ , & g_j^{W_i}(\mathbf{z}) &\stackrel{\text{def}}{=} W_i(j,\mathbf{z}) \quad (=\mathbf{S}(j,i,\mathbf{z}))\ ,\\ f_j^{R_i}(\mathbf{y}) &\stackrel{\text{def}}{=} R_i(\mathbf{y},j) \quad (=\mathbf{S}(i,\mathbf{y},j))\ , & g_j^{R_i}(\mathbf{z}) &\stackrel{\text{def}}{=} R_i(j,\mathbf{z}) \quad (=\mathbf{S}(i,j,\mathbf{z}))\ , \end{aligned}$$

   and

$$t^{L_j(Q_i)}(\mathbf{x}) = L_j(Q_i(\mathbf{x},\mathbf{y}))\ .$$

   Then, define the message:

$$m_{i\to j} := \left(\mathsf{exchange}, i, j, f_j^{Q_i}(\mathbf{x}), g_j^{Q_i}(\mathbf{y}), f_j^{R_i}(\mathbf{x}), g_j^{R_i}(\mathbf{z}), f_j^{W_i}(\mathbf{y}), g_j^{W_i}(\mathbf{z}), t^{L_j(Q_i)}(\mathbf{x})\right)\ .$$

(c) Verify that $\mathsf{consistent}(\mathsf{share}_i, m_{i\to i}) = 1$ (as per Definition 5.8), i.e., the share that $P_i$ received from the dealer is consistent with itself.

(d) If all the above conditions hold, then $P_i$ sends to each $P_j$ its sub-share $m_{i\to j}$.

(e) Upon receiving a message $m_{j\to i}$ from $P_j$, verify that it is consistent with $\mathsf{share}_i$ (i.e., $\mathsf{consistent}(\mathsf{share}_i, m_{j\to i}) = 1$), where $\mathsf{share}_i$ received from the dealer and $\mathsf{consistent}$ is as Definition 5.8. If so, then broadcast $\mathsf{Good}(i, j)$.

3. **(Identifying Star or Clique)** The dealer does as follows. Initalize a dynamic undirected graph $G = (V, E)$ with $V = [n]$. Upon receiving broadcasted messages $\mathsf{Good}(i, j)$ from $P_i$ and $\mathsf{Good}(j, i)$ from $P_j$, add the edge $(i, j)$ to $E$. Run Algorithm 5.10 and if the output is not $\perp$ then broadcast the output and output $\mathsf{OK}$. Otherwise, continue to listen to $\mathsf{Good}$ messages and repeat.
4. **(Verifying Star or Clique)** Each party $P_i$:
   (a) Initialize an undirected graph $G_i = (V_i, E_i)$ with $V_i = [n]$. Upon receiving broadcasted messages $\mathsf{Good}(i, j)$ from $P_i$ and $\mathsf{Good}(j, i)$ from $P_j$, add the edge $(i, j)$ to $E_i$.
      i. If $(\mathsf{Dense}, \mathsf{C})$ is received from the broadcast of the dealer, validate that $|\mathsf{C}| \geq n - t$, and that each node $i \in \mathsf{C}$ has a degree at least $3t + t/2 + 1$ in $G_i$. If the conditions hold, output $\mathsf{OK}$.
      ii. If $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$ is received from the broadcast of the dealer, $P_i$ verifies that $\mathsf{C} \subset \mathsf{D}$, $|\mathsf{C}| \geq 2t + t/2 + 1$, $|\mathsf{D}| \geq n - t$ and that for every $c \in \mathsf{C}$ and $d \in \mathsf{D}$ the edge $(c, d)$ is in $G_i$. If the conditions hold, then output $\mathsf{OK}$.
   (b) Otherwise, continue to listen to $\mathsf{Good}$ messages, and with each message it updates the graph $G_i$ and repeats the above checks.

---

**Algorithm 5.10: Finding a $\mathsf{BigStar}$ or a Clique**

---

- **Input:** An undirected graph $G$ over $[n]$.
  1. Initialize a set $\mathsf{C} = \emptyset$.
  2. For each node $i$ that has degree higher than $3t + t/2 + 1$, add $i$ to $\mathsf{C}$.
  3. If $|\mathsf{C}| \geq 3t + 1$ then output $(\mathsf{Dense}, \mathsf{C})$.
  4. Otherwise, let $\overline{\mathsf{C}} = [n] \setminus \mathsf{C}$, i.e., the set of all nodes with degree less than $3t + t/2 + 1$. For each node $i \in \overline{\mathsf{C}}$:
     (a) Consider the graph $G[\Gamma(i)]$ which consists of all vertices in $G$ that have an edge to $i$ (including $i$). If $G[\Gamma(i)]$ consists of less than $3t + t/2 + 1$ vertices, then add arbitrary vertices in $G$ (say the lexicographically first one) to have a graph with exactly $3t + t/2 + 1$ vertices.
     (b) Run $\mathsf{STAR}$ algorithm on input $(G[\Gamma(i)], n', t/2)$ where $n'$ is the number of vertices in $G[\Gamma(i)]$ (i.e., at least $3t+t/2+1$). If the output is $(\mathsf{C}, \mathsf{D})$, then output $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$.
  5. Otherwise, output $\perp$.

---

**Theorem 5.11.** *Let $n \geq 4t + 1$. Protocol 5.9 securely computes Functionality 5.4 in the presence of a malicious adversary controlling at most $t$ parties. It has a communication complexity of $\mathcal{O}(n^3 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits of broadcast. Each party broadcasts at most $\mathcal{O}(n \log n)$ bits.*

To show the full simulatability, We prove the following two claims regarding the protocol: validity and binding.

**Claim 5.12** (Validity)**.** *If the dealer is honest and starts with* $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$*, and the inputs of the honest parties* $\mathsf{ExternalValidity}_j$ *are such that* $\mathsf{ExternalValidity}(\mathsf{share}_j) = 1$ *where* $\mathsf{share}_j$ *is defined as in the protocol, then the protocol eventually terminates and each honest party outputs* $\mathsf{OK}$*.*

*Proof.* Since for each honest party it holds that $\mathsf{ExternalValidity}(\mathsf{share}_j) = 1$, and all shares of honest parties agree with each other, we have that eventually, each honest $P_j$ will broadcast $\mathsf{Good}(j, \ell)$ for all honest parties $\ell \notin I$. We show that the dealer must eventually broadcast either $\mathsf{Dense}$ or $\mathsf{BigStar}$ messages. Consider time $T$ in which all $\mathsf{Good}$ messages between pairs of honest parties were received by the dealer. There are two cases to consider:

- The dealer broadcasted $(\mathsf{Dense}, \mathsf{C})$ or $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$ before time $T$;
- At time $T$, if each honest node has degree at least $3t + t/2 + 1$ then we have a total of $n - t$ nodes that have high degree, and thus the dealer broadcasts $(\mathsf{Dense}, \mathsf{C})$ where $\mathsf{C}$ contains (at least) all honest parties.
- Otherwise, there exists an honest party $j \notin I$ that has degree smaller than $3t + t/2 + 1$. Since we are at time $T$, all the missing edges are of corrupted parties. That is, when considering the graph $G[\Gamma(j)]$, the vertices that are removed correspond to $t/2$ corrupted parties. Thus, the graph $G[\Gamma(j)]$ contains $n' \geq 3t + t/2 + 1$ vertices, and contains a clique of size $3t + 1$ (i.e., all honest parties). According to Claim 3.7, the STAR algorithm finds a $(\mathsf{C}, \mathsf{D})$-star with $|\mathsf{C}| \geq n' - 2 \cdot (t/2) \geq 2t + t/2 + 1$ and $|\mathsf{D}| \geq n' - t/2 \geq 3t + 1$.

The dealer thus eventually broadcasts one of the messages $(\mathsf{Dense}, \mathsf{C})$ or $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$, and all honest parties eventually receive this message. Moreover, since all $\mathsf{Good}$ messages are broadcasted, eventually all honest parties will see the same edges as the honest dealer, and validates the $(\mathsf{Dense}, \mathsf{C})$ or $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$ messages. Once the broadcasted message of the dealer is validated by an honest $P_j$, it halts and output $\mathsf{OK}$. □

**Claim 5.13** (Termination)**.** *If one honest parties terminate, then all honest parties eventually terminates.*

*Proof.* This follows immediately from the guarantees of the broadcast (aka. A-cast): If the dealer broadcasts $(\mathsf{Dense}, \mathsf{C})$ or $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$ then all honest parties will eventually see this message. Moreover, if the property holds in the graph of one honest party, then all honest parties eventually see those edges and validate the property in their respective graphs. This is because the graph is defined by the $\mathsf{Good}$ messages that were also broadcasted. □

**Claim 5.14** (Binding)**.** *When the first honest party terminates, there exists a unique trivariate polynomial* $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ *of degree* $t + t/2$ *in each one of the variables* $\mathbf{x}, \mathbf{y}, \mathbf{z}$ *that can be extracted from the views of the honest parties. Moreover, for at least* $2t + 1$ *honest parties* $J \subset [n] \setminus I$ *it holds that* $\mathsf{ExternalValidity}_j(\mathsf{share}_j) = 1$*.*

*Proof.* Consider the first honest party that terminates, say some $P_{j^*}$. If $P_{j^*}$ terminates then the dealer must have broadcasted $\mathsf{Dense}$ or $\mathsf{BigStar}$ messages, and $P_{j^*}$ validated the respective property in its local graph. There are two cases to consider:

**Case I: The property is (Dense, C).** In that case, the graph that the honest party $P_{j^*}$ sees contains $n-t$ vertices $\mathsf{C}$, where each has degree at least $3t+t/2+1$. Consider the set of the lexicographically first $t+t/2+1$ honest parties $H \subseteq \mathsf{C} \setminus I$. Each such party inputs some $Q_h(\mathbf{x},\mathbf{y}), W_h(\mathbf{x},\mathbf{z})$ and $R_h(\mathbf{y},\mathbf{z})$. Consider the unique trivariate polynomial $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$ that satisfies $\mathbf{S}(\mathbf{x},\mathbf{y},h) = Q_h(\mathbf{x},\mathbf{y})$ for every $h \in H$. Such a trivariate polynomial is guaranteed to exist by Claim 3.9.

We now show that all honest parties in $\mathsf{C}$ agree with $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$.

**Claim 5.15.** *For every honest party $j \in \mathsf{C}$ it holds that $R_j(\mathbf{y},\mathbf{z}) = \mathbf{S}(j,\mathbf{y},\mathbf{z})$.*

*Proof.* Fix some $j \in \mathsf{C}$ and $h \in H$. Since $P_j$ and $P_h$ both have bivariate polynomials with degrees $3t + t/2 + 1$ (otherwise, they would never send $\mathsf{Good}$), and so they have at least $3t+1$ vertices in common, in which at least $2t+1$ honest parties are in their intersection. That is, there is a set $K$ of honest parties of cardinality at least $2t+1$ in which $\mathsf{Good}(j,k)$ and $\mathsf{Good}(h,k)$ was broadcasted for every $k \in K$. In particular, $P_h$ and $P_k$ verified that (among other things):

$$(\mathbf{S}(\mathbf{x},k,h) =) \quad Q_h(\mathbf{x},k) = f_k^{Q_h}(\mathbf{x}) = f_h^{W_k}(\mathbf{x}) = W_k(\mathbf{x},h) \ .$$

$P_k$ and $P_j$ also exchanged shares, and verified that $g_j^{W_k}(\mathbf{z}) = g_k^{R_j}(\mathbf{z})$, that is, it holds that

$$W_k(j,\mathbf{z}) = g_j^{W_k}(\mathbf{z}) = g_k^{R_j}(\mathbf{z}) = R_j(k,\mathbf{z}) \ ,$$

and in particular on $\mathbf{z} = h$:

$$\mathbf{S}(j,k,h) = W_k(j,h) = R_j(k,h) \ .$$

Since this holds for every $k \in K$, we get that two degree $t+t/2$ polynomials $\mathbf{S}(j,\mathbf{y},h)$ and $R_j(\mathbf{y},h)$ agree on $2t+1$ points, and therefore $\mathbf{S}(j,\mathbf{y},h) = R_j(\mathbf{y},h)$. Moreover, since this holds for every $h \in H$, we have that two degree-$(t+t/2)$ bivariate polynomials $\mathbf{S}(j,\mathbf{y},\mathbf{z})$ and $R_j(\mathbf{y},\mathbf{z})$ must agree, i.e., $\mathbf{S}(j,\mathbf{y},\mathbf{z}) = R_j(\mathbf{y},\mathbf{z})$. □

**Claim 5.16.** *For every honest party $j \in \mathsf{C}$ it holds that $W_j(\mathbf{x},\mathbf{z}) = \mathbf{S}(\mathbf{x},j,\mathbf{z})$.*

*Proof.* As before, fix some $j \in \mathsf{C}$ and $h \in H$, and consider $K$ of cardinality $2t+1$ that agree both with $P_j$ and $P_h$. For every $k \in K$, $P_h$ and $P_k$ verified that

$$(\mathbf{S}(k,\mathbf{y},h) =) \quad Q_h(k,\mathbf{y}) = g_k^{Q_h}(\mathbf{y}) = f_h^{R_k}(\mathbf{y}) = R_k(\mathbf{y},h) \ .$$

Moreover, $P_j$ and $P_k$ verified that

$$W_j(k,\mathbf{z}) = g_k^{W_j}(\mathbf{z}) = g_j^{R_k}(\mathbf{z}) = R_k(j,\mathbf{z}) \ .$$

In particular, for $\mathbf{z} = h$ it holds that $W_j(k,h) = R_k(j,h) = \mathbf{S}(k,j,h)$. Since this holds for every $k \in K$, we have that two univariate polynomials $W_j(\mathbf{x},h)$ and $\mathbf{S}(\mathbf{x},j,h)$ must agree. Since this holds for every $h \in H$, we get that the two polynomials $W_j(\mathbf{x},\mathbf{y})$ and $\mathbf{S}(\mathbf{x},j,\mathbf{y})$ must agree, and so $W_j(\mathbf{x},\mathbf{y}) = \mathbf{S}(\mathbf{x},j,\mathbf{y})$. □

**Claim 5.17.** *For every honest party $j \in \mathsf{C}$ it holds that $Q_j(\mathbf{x},\mathbf{y}) = \mathbf{S}(\mathbf{x},\mathbf{y},j)$.*

*Proof.* As before, $P_h$ and $P_k$ exchanged the shares

$$(\mathbf{S}(h,k,\mathbf{z}) =) \quad R_h(k,\mathbf{z}) = g_k^{R_h}(\mathbf{z}) = g_h^{W_k}(\mathbf{z}) = W_k(h,\mathbf{z}) \ ,$$

where $\mathbf{S}(h,k,\mathbf{z}) = R_h(k,\mathbf{z})$ follows from Claim 5.15. Moreover, $P_k$ and $P_j$ verified that

$$Q_j(\mathbf{x},k) = f_k^{Q_j}(\mathbf{x}) = f_j^{W_k}(\mathbf{x}) = W_k(\mathbf{x},j) \ ,$$

and in particular for $\mathbf{x} = h$ it holds that $Q_j(h,k) = W_k(h,j) = \mathbf{S}(h,k,j)$. Since this holds for every $k \in K$, we get that the two univariate polynomials $Q_j(h,\mathbf{y})$ and $\mathbf{S}(h,\mathbf{y},j)$ agree. Since it also holds for every $h \in H$, this means that $Q_j(\mathbf{x},\mathbf{y}) = \mathbf{S}(\mathbf{x},\mathbf{y},j)$. □

**Case II: The property is (BigStar, C, D).** In that case, the graph that the honest party $P_j^*$ sees contains a clique $\mathsf{C}$ of size $2t + t/2 + 1$. This implies that there is a set of at least $t + t/2 + 1$ honest parties for which for every $j,k \in \mathsf{C}$, the party $P_{j^*}$ viewed $\mathsf{Good}(j,k)$ and $\mathsf{Good}(k,j)$, and thus the shares that $P_k$ and $P_j$ agree with each other. Let $H$ be the lexicographically first $t + t/2 + 1$ honest parties in $\mathsf{C}$. Consider the unique trivariate polynomial $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$ that satisfies $\mathbf{S}(\mathbf{x},\mathbf{y},h) = Q_h(\mathbf{x},\mathbf{y})$ for every $h \in H$. Such a trivariate polynomial is guaranteed to exist by Claim 3.9. We now show that all honest parties in $\mathsf{D}$ hold shares on the polynomial $\mathbf{S}(x,y,z)$.

**All honest parties in $H$ hold shares on $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$.** Clearly, for every $h \in H$ it holds that $Q_h(\mathbf{x},\mathbf{y}) = \mathbf{S}(\mathbf{x},\mathbf{y},h)$. For every $i,j \in H$ we have that $P_i$ and $P_j$ verified their shares, and thus we have that

$$\mathbf{S}(\mathbf{x},i,j) = Q_j(\mathbf{x},i) = f_i^{Q_j}(\mathbf{x}) = f_j^{W_i}(\mathbf{x}) = W_i(\mathbf{x},j) \ .$$

Since this holds for every $j \in H$, we get that that the two $t + t/2 + 1$ bivariate polynomials $\mathbf{S}(\mathbf{x},i,\mathbf{z})$ and $W_i(\mathbf{x},\mathbf{z})$ agree, and so for every $i \in H$, $\mathbf{S}(\mathbf{x},i,\mathbf{z}) = W_i(\mathbf{x},\mathbf{z})$.

Similarly, for every $i,j \in H$ we have that $P_i$ and $P_j$ verified that

$$(\mathbf{S}(i,\mathbf{y},j) =) \quad Q_j(i,\mathbf{y}) = g_i^{Q_j}(\mathbf{y}) = f_j^{R_i}(\mathbf{y}) = R_i(\mathbf{y},j)$$

Since this holds for every $j \in H$, we get that the two $t + t/2 + 1$ bivariate polynomials $\mathbf{S}(i,\mathbf{y},\mathbf{z})$ and $R_i(\mathbf{y},\mathbf{z})$ agree on $t + t/2 + 1$ univariate polynomials, and so for every $i \in H$ it holds that $\mathbf{S}_i(\mathbf{y},\mathbf{z}) = R_i(\mathbf{y},\mathbf{z})$.

**All honest parties in $\mathsf{D}$ hold shares on $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$.** Similarly to above, each party in $i \in \mathsf{D}$ agrees with each party in $H$. Thus, for every $i \in \mathsf{D}$ and $h \in H$ we have that

$$(\mathbf{S}(\mathbf{x},h,i) =) \quad W_h(\mathbf{x},i) = f_i^{W_h}(\mathbf{x}) = f_h^{Q_i}(\mathbf{x}) = Q_i(\mathbf{x},h) \ .$$

Since this holds for every $h \in H$, we have that $\mathbf{S}(\mathbf{x},\mathbf{y},i) = Q_i(\mathbf{x},\mathbf{y})$. Likewise,

$$\mathbf{S}(\mathbf{x},i,h) = Q_h(\mathbf{x},i) = f_i^{Q_h}(\mathbf{x}) = f_h^{W_i}(\mathbf{x}) = W_i(\mathbf{x},h) \ .$$

Since this holds for every $h \in H$, we get that $W_i(\mathbf{x},\mathbf{z}) = \mathbf{S}(\mathbf{x},i,\mathbf{z})$. Finally,

$$(\mathbf{S}(i,\mathbf{y},h) =) \quad Q_h(i,\mathbf{y}) = g_i^{Q_h}(\mathbf{y}) = f_h^{R_i}(\mathbf{y}) = R_i(\mathbf{y},h) \ ,$$

and since this holds for every $h \in H$, we have that $\mathbf{S}(i,\mathbf{y},\mathbf{z}) = R_i(\mathbf{y},\mathbf{z})$.

**External validity.** An honest party $P_j$ does not send shares to other parties, and in particular does not broadcast $\mathsf{Good}(j,k)$ for every party $P_k$ if its external validity was not 1. In the case

of $(\mathsf{Dense}, \mathsf{C})$, we are guaranteed to have $|\mathsf{C}| \geq n - t \geq 3t + 1$ and therefore the set contains at least $2t + 1$ honest parties. Since those parties broadcasted $\mathsf{Good}$, we have $2t + 1$ honest parties with validated external validity predicate. Likewise, in the case of $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$, we have that $|\mathsf{D}| \geq n - t \geq 3t + 1$, honest parties in $\mathsf{D}$ validated their external validity, and therefore we have at least $2t + 1$ honest parties with validated external validity predicate. □

*Proof.* We are now ready to show the simulation.

**The case of an honest dealer.** In the honest dealer case, we assume that the honest parties input $\mathsf{ExternalValidity}_j$ which on the shares of the honest parties output 1. We then have:

1. The simulator receives from the functionality the shares of the corrupted parties $\mathsf{share}_i$ for every $i \in I$.
2. Invoke the adversary $\mathcal{A}$. Simulate $\mathcal{A}$ receiving headers $(\mathsf{shares}, j)$ for every $j \in [n]$. Moreover, simulate $\mathcal{A}$ receiving the message $(\mathsf{share}, i, \mathsf{share}_i)$ where $\mathsf{share}_i$ is as received from the functionality.
3. Once $\mathcal{A}$ delivers the message $(\mathsf{shares}, j)$ for $j \notin I$, simulate $P_j$ sending $\mathsf{exchange}(j, k)$ for every $k \notin I$. Moreover, for each $i \in I$, simulate $P_j$ sending the message $m_{j \to i}$ to $P_i$, where
$$m_{j \to i} := (\mathsf{exchange}, j, i, Q_i(\mathbf{x}, j), R_i(\mathbf{y}, j), Q_i(\mathbf{x}, j), R_i(j, \mathbf{z}), Q_i(j, \mathbf{y}), W_i(j, \mathbf{z}), T_i(\mathbf{x}, j)) \ .$$
4. For every message $\mathcal{A}$ sends in the name of $P_i$ to some honest $P_j$ a message $m_{i \to j}$, check that $m_{i \to j}$ was sent correctly according to the protocol given $\mathsf{share}_i$. Once $\mathcal{A}$ delivers the message to $P_j$ and the message is correct, simulate $P_j$ broadcasting $\mathsf{Good}(j, i)$.
5. Once $\mathcal{A}$ delivers the message $\mathsf{exchange}(j, k)$ from $P_j$ to $P_k$ (and it already delivered the message $\mathsf{share}(k)$ from the dealer to $P_k$), then simulate $P_k$ broadcasting $\mathsf{Good}(k, j)$.
6. Simulate the honest dealer as in the protocol, running Algorithm 5.10. Since all honest parties eventually broadcast $\mathsf{Good}(k, \ell)$ for every pair $k, \ell \notin I$, eventually (as shown in Claim 5.12) the dealer will ask to broadcast either $(\mathsf{Dense}, \mathsf{C})$ or $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$. Simulate the dealer broadcasting this message.
7. Simulate each honest party $P_j$ verifying the validity of the broadcasted messages by the dealer. When the simulated $P_j$ terminates (this occurs when the adversary delivers some message to $P_j$, either a message broadcasted by the dealer or some other party), then the simulator allows the deliver of the output of the functionality (which is $\mathsf{OK}$) to the honest party $P_j$ in the ideal world.

We now show that the view of the environment $\mathcal{Z}$ is the same in both executions. From inspection, it is easy to see that the view of the adversary $\mathcal{A}$ is the same in both executions. In particular, this is due to the fact that the protocol and the simulation are deterministic, and that the messages of the honest parties to the corrupted parties can be simulated from the shares of the corrupted parties. As follows from Claim 5.12, the output of the honest parties in the case of an honest dealer is always $\mathsf{OK}$ (assuming that the external validity predicates that the honest parties input to the functionality give 1 on the shares provided by the dealer). Therefore, the output of the honest parties in the real world is always $\mathsf{OK}$, the same as in the ideal world. The scheduling in which parties receive outputs is determined by the adversary (i.e., the controlling of the router), and since the view of the adversary is exactly the same in both executions, its control over the router is exactly the same.

**The case of a corrupted dealer.**

1. The simulator invokes the adversary $\mathcal{A}$.
2. The simulator receives from the functionality the external validity predicates, i.e., $\mathsf{ExternalValidity}_j$, for every $j \notin I$.
3. The simulator simulates the honest parties in an execution of the protocol where the input of each $P_j$ is $\mathsf{ExternalValidity}_j$. The simulator also simulates the router of the real world. Note that the simulation might not terminate.
4. When the first honest party $P_{j^*}$ terminates, all honest parties will eventually terminate, see Claim 5.14. The Claim also shows how to extract a trivariate polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ from the views of the honest parties. The simulator sends $\mathbf{S}$ to the trusted party. For this particular $\mathbf{S}$, the external validity property holds for at least $2t + 1$ honest parties. Moreover, $\mathbf{S}$ is of degree at most $t + t/2$ in all three variables $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Thus, the functionality will accept this polynomial, and it would send $\mathsf{OK}$ to all parties.
5. The simulator continues to simulate the protocol with the adversary. Whenever a simulated honest party $P_j$ obtains an output in the simulated protocol, the simulator $\mathcal{S}$ delivers to the router to allow sending the dummy party $P_j$ in the ideal its output, $\mathsf{OK}$.
6. Eventually, all honest parties will receive output in the simulated execution. The simulator then terminates.

The view of the adversary is exactly the same in the real and ideal executions, as the simulated honest parties are deterministic and use the exact same inputs as in the real world. If some honest party terminates, then as follows from Claim 5.14 all honest parties would eventually terminate and with their shares lie on the same polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$. The environment $\mathcal{Z}$ sees the outputs at the same activation in the real and ideal: Upon the activation in which the first honest party $P_{j^*}$ terminates, the simulator extracts the trivariate polynomial and sends it to the trusted party, and then deliver the output to $P_{j^*}$. Since the view of the adversary is exactly the same, whenever a real honest party $P_k$ receives an output, a simulated $P_k$ receives an output in the simulation, and then the simulator extracts the router to deliver the output of the functionality to $P_k$ in the ideal world. □

# 6 Rate-1 Asynchronous Weak-Binding Secret Sharing

Protocol 5.9 provides a secret sharing of a trivariate polynomial with $\mathcal{O}(1)$ overhead. Along the way, it also allows some external verification ($\mathsf{ExternalValidity}$) and some computation on the trivariate shares. This section describes our weak-binding secret-sharing protocol with a shunning reconstruction. We remark again that we do not use this protocol in the paper. Nevertheless, we provide it as an independent primitive for completeness and as it might be useful as an independent primitive.

**Definition 6.1** (Asynchronous Weak-binding Secret Sharing with Shunning Reconstruction)**.** *Let $S$ be a finite domain, $|S| \geq 2$, and let $[n]$ be a set of parties that includes a distinguished* dealer*. An asynchronous weak-binding secret sharing with shunning reconstruction consists of two phases, a* sharing phase *and a* reconstruction phase*, with the following syntax.*

- Sharing: *At the beginning, the dealer holds a* secret *$s \in S$ and each party including the dealer holds an independent random input $r_i$. The parties may communicate in several time in sequence. Each time, each party can privately send messages to the other parties and it can also broadcast a message. Each message sent or broadcasted by $P_i$ is determined by the view*

*of $P_i$, consists of its input (if any), its random input and messages received from other parties in previous rounds.*

- Reconstruction: *At the beginning of the reconstruction, the parties are holding their view from the sharing phase and in addition the dealer maintains a list $L$ which is initialized to $\emptyset$. The reconstruction phase may involve several interactions, and at each time the parties send messages based on their view. At the end of the reconstruction, each party either outputs a value or never terminates. When the parties do not terminate, the dealer will have at least $t/2+1$ parties in its list $L$.*

*We should have the following properties for any adversary $A = (A_s, A_r)$ corrupting at most $t$ parties:*

- Termination: *If the dealer is honest then each honest party eventually terminates the sharing phase. If some honest party terminates the sharing phase, then every honest party must terminate it eventually. For the reconstruction phase one of these must hold: (a) If some honest party terminates the reconstruction phase, then all other honest parties will eventually terminate or (b) the dealer will have at least $t/2+1$ parties in $L$.*

- Privacy: *If $D$ is honest then the adversary's view during the sharing phase reveals almost no information on $s$. Formally, let $D_s$ is the view $A$ in the sharing phase on secret $s$. Then, for any $s \neq s'$, the random variables $D_s$ and $D_{s'}$ are identical.*

- Weak-binding: *At the end of the sharing phase there is a value $s^* \in S$ such that at the end of the reconstruction phase, if the parties terminate, then the output will be $s^*$. If the dealer is honest, then $s^* = s$.*

---

**Protocol 6.2: Asynchronous Weak-binding Secret Sharing**

---

**Input:** The input of the dealer is some trivariate polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Each other party has no input.

**Sharing phase:**

1. Each party $P_i$ and the dealer: Run Protocol 5.9 with $\mathsf{ExternalValidity}_i(\cdot)$ as the predicate that always returns 1, and each $L_j(\cdot) = \perp$ for every $\mathsf{share}_j$.
2. If the protocol terminates with output $\mathsf{OK}$, then:
   (a) If $(\mathsf{Dense}, \mathsf{C})$ was received as the broadcasted message from the dealer: if $i \in \mathsf{C}$ then store $X = \mathsf{C}$ and $\mathsf{share}_i$. (excluding the last element – which is $\perp$.)
   (b) If $(\mathsf{BigStar}, \mathsf{C}, \mathsf{D})$ was received as the broadcasted message from the dealer: if $i \in \mathsf{D}$ then store $X = \mathsf{D}$ and $\mathsf{share}_i$. (excluding the last element.)
   (c) Otherwise, store $X = \mathsf{C}$ if $\mathsf{Dense}$ and $X = \mathsf{D}$ if $\mathsf{BigStar}$.

---

**Shunning Reconstruction phase:**

1. **(Broadcasting the Polynomial)** The dealer:
   (a) Initialise a shunning list $\mathsf{Shun} = X$.
   (b) Broadcast a trivariate polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$.
2. **(Verifying the Dealer's Polynomial)** Each party $P_i$:
   (a) Upon receiving a polynomial $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ from the dealer, verify that the polynomial is of degree at most $t + t/2$ in each variable. If not, then discard the dealer and terminate.

(b) If $i \in X$, then verify that $\mathbf{S}(\mathbf{x}, \mathbf{y}, i) = Q_i(\mathbf{x}, \mathbf{y})$, $\mathbf{S}(\mathbf{x}, i, \mathbf{z}) = W_i(\mathbf{x}, \mathbf{z})$ and $\mathbf{S}(i, \mathbf{y}, \mathbf{z}) = R_i(\mathbf{y}, \mathbf{z})$ holds. If all the conditions hold, then $P_i$ broadcasts $\mathsf{OK}$.

3. **(Output)**
   (a) Upon receiving the $\mathsf{OK}$ from the broadcast of $P_i$, the dealer updates $\mathsf{Shun} = \mathsf{Shun} \setminus \{i\}$.
   (b) Upon receiving $\mathsf{OK}$ from at least $2t + t/2 + 1$ parties in $X$, party $P_i$ outputs $\mathbf{S}(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and terminates. Otherwise, it continues to wait for $\mathsf{OK}$ messages.

---

We use the term "shunning" for reconstruction to indicate that the reconstruction phase offers the following guarantees: either the reconstruction succeeds, or the dealer can identify at least $t/2 + 1$ parties thereafter.

Essentially, in an honest dealer case, termination of the sharing phase is guaranteed. However, since the set $X$ does not necessarily contain $2t + t/2 + 1$ honest parties, for reconstruction, we might need the adversary's help. This is why the reconstruction is either guaranteed, or the dealer shuns at least $t/2 + 1$ parties. In the case of a corrupted dealer, once the sharing terminates, reconstruction must be to the same polynomial (or discard the dealer, or not terminate, but cannot be ended successfully with a different polynomial). We formalize and prove the properties of this protocol below.

**Claim 6.3** (Sharing Termination)**.** *If the dealer is honest, then each honest party terminates the sharing phase. If some honest party terminates the sharing phase then every honest party must terminate it eventually.*

*Proof.* Since for each honest party $P_j$, we have $\mathsf{ExternalValidity}_j(\cdot) = 1$, for an honest dealer, by Claim 5.12 we have that Protocol 5.9 terminates with the output $\mathsf{OK}$. Hence, sharing phase always completes successfully. The latter follows immediately from Claim 5.13. □

**Claim 6.4** (Reconstruction Termination)**.** *Either all the honest parties terminate the reconstruction, or the dealer shuns at least $t/2 + 1$ parties.*

*Proof.* We have the following two cases to consider:

1. **There exists some honest party $P_{j^*}$ which terminates:** This implies that $P_{j^*}$ received at least $2t + t/2 + 1$ broadcasts of $\mathsf{OK}$ messages from the parties in set $X$ identified during the sharing phase. These messages will be eventually received by all the honest parties (including the dealer), ensuring that all the honest parties terminate.
2. **No honest party has terminated:** This implies that less than $2t + t/2 + 1$ $\mathsf{OK}$ messages are received by the honest parties, which includes the dealer. In that case, the dealer's shunning set consists of all the parties from $X$ from whom the dealer has not received a broadcast of $\mathsf{OK}$. Since $|X| \geq n - t$, we have that $|\mathsf{Shun}| \geq t/2 + 1$.

□

The following claim follows from Theorem 5.11:

**Claim 6.5** (Privacy)**.** *If the dealer is honest, then the adversary's view during the sharing phase reveals no information on the dealer's input.*

**Claim 6.6** (Weak Binding)**.** *At the termination of the sharing phase, there is a unique trivariate polynomial* $\mathbf{S}'(\mathbf{x},\mathbf{y},\mathbf{z})$ *with degree* $3t/2$ *that might be reconstructed in the reconstruction phase. Moreover, if the dealer is honest then* $\mathbf{S}'(\mathbf{x},\mathbf{y},\mathbf{z}) = \mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$ *where* $\mathbf{S}(\mathbf{x},\mathbf{y},\mathbf{z})$ *is the input of the dealer.*

*Proof.* If the sharing phase terminates for the honest parties, it implies that Protocol 5.9 terminates with OK and all the parties hold a set $X$ of size at least $n-t$. By Claim 5.14, we have that all the honest parties in $X$ hold shares on some unique trivariate polynomial $\mathbf{S}'(\mathbf{x},\mathbf{y},\mathbf{z})$.

Further, if the reconstruction terminates for some honest party, it implies that the dealer broadcasted a trivariate polynomial, say $\mathbf{S}^*$, with degree $3t/2$ in each variable. Moreover, at least $2t+t/2+1$ parties from $X$ broadcasted OK after verifying the consistency of their shares with the dealer's broadcasted trivariate polynomial. Of these, at least $t+t/2+1$ parties are guaranteed to be honest. We have that the two polynomials $\mathbf{S}^*$ and $\mathbf{S}'$ agree in at least $(t+t/2+1)^3$ points, and hence $\mathbf{S}^* = \mathbf{S}'$. Moreover, an honest dealer always broadcasts $\mathbf{S}^* = \mathbf{S}$ and hence parties output the dealer's polynomial. □

# 7 Verifiable Triple Sharing

In this section, we build upon packed VSS (Functionality 4.1) and Functionality 5.1 to show how a dealer can verifiably share $\mathcal{O}(n^2)$ triples simultaneously.

The functionality for verifiable triple sharing appears below, followed by the protocol. The Shamir-shares of party $P_j$ for the $(t/2+1)^2$ multiplication triples are as follows, following the invocation of the functionality or the protocol: $(A^u(-\beta,j), B^u(-\beta,j), C^u(-\beta,j))$ for every $u,\beta \in \{0,\ldots,t/2\}$.

**Functionality 7.1: Verifiable Triple Secret Sharing**

The functionality is parameterized by a set of corrupted parties $I \subseteq [n]$.

1. The dealer sends to the functionality 3 sets of $t/2+1$ polynomials $\{A^u(\mathbf{x},\mathbf{y})\}$, $\{B^u(\mathbf{x},\mathbf{y})\}$ and $\{C^u(\mathbf{x},\mathbf{y})\}$ for each $u \in \{0,\ldots,t/2\}$.
2. The functionality verifies that each polynomial is of degree at most $t+t/2$ in $\mathbf{x}$ and $t$ in $\mathbf{y}$. If not, it does not terminate.
3. If the dealer is honest, then for each $u \in \{0,\ldots,t/2\}$, give adversary the shares $(A^u(\mathbf{x},i), A^u(i,\mathbf{y}))$, $(B^u(\mathbf{x},i), B^u(i,\mathbf{y}))$ and $(C^u(\mathbf{x},i), C^u(i,\mathbf{y}))$ for every $i \in I$.
4. The functionality verifies that for every $u,\beta \in \{0,\ldots,t/2\}$ it holds that

$$A^u(-\beta,0)\cdot B^u(-\beta,0) = C^u(-\beta,0)\ .$$

   If yes, then it sends $(A^u(\mathbf{x},j), A^u(j,\mathbf{y}))$, $(B^u(\mathbf{x},j), B^u(j,\mathbf{y}))$ and $(C^u(\mathbf{x},j), C^u(j,\mathbf{y}))$ for every $u \in \{0,\ldots,t/2\}$ to each party $P_j$ and halts. Otherwise, the functionality never terminates.

**Protocol 7.2: Verifiable Triple Secret Sharing Protocol**

- **Input:** The dealer holds the polynomials $A^u(\mathbf{x},\mathbf{y})$, $B^u(\mathbf{x},\mathbf{y})$ and $C^u(\mathbf{x},\mathbf{y})$ of degree $t+t/2$ in $\mathbf{x}$ and $t$ in $\mathbf{y}$ for every $u \in \{0,\ldots,t/2\}$ such that $A^u(-\beta,0)\cdot B^u(-\beta,0) = C^u(-\beta,0)$ holds for every $\beta \in \{0,\ldots,t/2\}$.

- **The protocol:**
  1. The dealer invokes Functionality 4.1 with its polynomials $A^u(\mathbf{x},\mathbf{y})$, $B^u(\mathbf{x},\mathbf{y})$ and $C^u(\mathbf{x},\mathbf{y})$ for every $u \in \{0,\ldots,t/2\}$ in a batched manner.
  2. The dealer invokes Functionality 5.1 with the input $A^u(-\beta,\mathbf{y})$, $B^u(-\beta,\mathbf{y})$ and $C^u(-\beta,\mathbf{y})$ for every $u,\beta \in \{0,\ldots,t/2\}$.
  3. Upon receiving an output $(A^u(\mathbf{x},j),A^u(j,\mathbf{y}))$, $(B^u(\mathbf{x},j),B^u(j,\mathbf{y}))$ and $(C^u(\mathbf{x},j),C^u(j,\mathbf{y}))$ from the functionality, each $P_j$ invokes the Functionality 5.1 with the inputs $(A^u(-\beta,j),$ $B^u(-\beta,j),C^u(-\beta,j))$ for every $u,\beta \in \{0,\ldots,t/2\}$.
  4. Upon receiving an output $\mathsf{OK}$ from the Functionality 5.1, $P_j$ outputs $(A^u(\mathbf{x},j),A^u(j,\mathbf{y}))$, $(B^u(\mathbf{x},j),B^u(j,\mathbf{y}))$ and $(C^u(\mathbf{x},j),C^u(j,\mathbf{y}))$, where $(A^u(-\beta,j),B^u(-\beta,j),C^u(-\beta,j))$ for every $u,\beta \in \{0,\ldots,t/2\}$ defines $P_j$'s degree-$t$ Shamir-share of the $(t/2+1)^2$ multiplication triples.

---

**Theorem 7.3.** *Let $n \geq 4t+1$. Protocol 7.2 securely computes Functionality 7.1 in the presence of a malicious adversary controlling at most $t$ parties. It has a communication complexity of $\mathcal{O}(n^3 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits of broadcast for sharing $\mathcal{O}(n^2)$ triples simultaneously. Each party broadcasts at most $\mathcal{O}(n \log n)$ bits.*

*Proof.* We show the case of an honest dealer and of a corrupted dealer separately.

**Case I – the case of an honest dealer.** The simulator $\mathcal{S}$ is as follows:

1. Upon activation, invoke the adversary $\mathcal{A}$.
2. The simulator receives from the functionality the shares $(f_i^{A^u}(\mathbf{x}),g_i^{A^u}(\mathbf{y}))$, $(f_i^{B^u}(\mathbf{x}),g_i^{B^u}(\mathbf{y}))$ and $(f_i^{C^u}(\mathbf{x}),g_i^{C^u}(\mathbf{y}))$ for every $u \in \{0,\ldots,t/2\}$ and every $i \in I$.
3. Simulate the invocation of the inner Functionality 4.1 for the adversary: for every $i \in I$, send to the adversary the shares $(f_i^{A^u}(\mathbf{x}),g_i^{A^u}(\mathbf{y}))$, $(f_i^{B^u}(\mathbf{x}),g_i^{B^u}(\mathbf{y}))$ and $(f_i^{C^u}(\mathbf{x}),g_i^{C^u}(\mathbf{y}))$ for every $u \in \{0,\ldots,t/2\}$.
4. Simulate the invocation of the inner Functionality 5.1 for the adversary: for every $i \in I$, give the adversary $(f_i^{A^u}(-\beta),f_i^{B^u}(-\beta),f_i^{C^u}(-\beta))$ for every $u,\beta \in \{0,\ldots,t/2\}$.
5. Simulate the Functionality 5.1 returning $\mathsf{OK}$. When the adversary delivers $\mathsf{OK}$ to a party $P_j$ in the simulated protocol, the simulator delivers the output of Functionality 7.1 to $P_j$ in the ideal world.

Clearly, since the protocol and the simulation are deterministic, the view of the adversary $\mathcal{A}$ is identical in both the real and ideal executions. It thus remains to show that the output of honest parties is the same in both the executions.

In the ideal world, an honest dealer always invokes the functionality with valid polynomials. Hence, the functionality delivers the shares on the dealer's polynomials to each honest party. In the real world, an honest dealer's polynomials always satisfy the conditions of Functionality 4.1 and from its guarantees (Claim 4.5) we have that each honest party receives its share. Moreover, the dealer's polynomials also satisfy the conditions of Functionality 5.1. Hence the functionality returns $\mathsf{OK}$ to all the honest parties, which in turn output their respective shares on the dealer's polynomials. Hence, the output of honest parties is identical in the real and ideal executions. Moreover, although the scheduling of message delivery is determined by the adversary $\mathcal{A}$, its view is identical in both the executions. Hence, the scheduling is also identical.

**Case II – the case of a corrupt dealer.** The simulator $\mathcal{S}$ is as follows:

1. The simulator invokes the adversary $\mathcal{A}$.
2. Since the protocol is deterministic and the honest parties do not have any input to the protocol, the simulator can simulate all honest parties in an execution with $\mathcal{A}$. That is, the simulator knows all the messages $\mathcal{A}$ sends to the honest parties and hence can also simulate the communication among honest parties. This includes simulating Functionalities 4.1, 5.1. Note that the simulation may not terminate.
3. If there exists some honest party $P_{j^*}$ that terminates, then it implies that $P_{j^*}$ terminates in the simulation of Functionality 4.1. By Claim 4.6 we have that all the honest parties eventually terminate.
4. The simulator interpolates the dealer's polynomials from the shares of the lexicographically first $t+1$ simulated honest parties. It is guaranteed that the polynomials have degree $t+t/2$ in $\mathbf{x}$ and $t$ in $\mathbf{y}$.
5. Moreover, since $P_{j^*}$ terminated in the protocol, it also implies that the simulation of Functionality 5.1 terminated with the output $\mathsf{OK}$.
6. The simulator sends to the functionality the interpolated polynomials ensuring that all the honest parties will eventually receive the output. When the adversary delivers $\mathsf{OK}$ to a party $P_j$ in the simulated protocol, the simulator delivers the output of Functionality 7.1 to $P_j$ in the ideal world.

It is easy to see that the since the honest parties do not have inputs and the simulator emulates the honest parties as in the real world, the view of the adversary in the ideal and real execution is the same. Moreover, if an honest party $P_{j^*}$ terminates in the real execution, then the same holds true in the simulated execution. The simulator extracts the input polynomials of the dealer from the view of these simulated honest parties, and sends it to the functionality ensuring that the output of honest parties is identical in the ideal model. □

## 7.1 Batching for Linear overhead per triple

We note that the overall communication of one instance of verifiable triple sharing protocol is $\mathcal{O}(n^3 \log n)$ over point-to-point channels and $\mathcal{O}(n^2 \log n)$ using broadcast. Using the broadcast of [14], the total cost turns out to be $\mathcal{O}(n^4 \log n)$ for sharing $\mathcal{O}(n^2)$ triples. We make the cost linear per triple by simply batching $n$ instances of the triple sharing protocol under the same dealer. Since all the instances have the same dealer, the broadcasts communication can be common for all. For instance, $P_i$ can send a single broadcast of $\mathsf{Good}(i,j)$ after checking consistency with $P_j$ in all the instances of AVSS and triple sharing. Similarly, the dealer can run the $\mathsf{Star}$ algorithm just once for all the AVSSs and broadcast one common $\mathsf{Star}$. Likewise, Algorithm 5.10 also is run for all the trivariate sharings together and the output is broadcast once for all.

This batching allows us to keep the broadcast communication the same as before i.e $\mathcal{O}(n^4 \log n)$. The point-to-point communication increases by a factor of $n$ and now becomes $\mathcal{O}(n^4 \log n)$. However, we are now able to share $n^3$ triplets, and thus achieve a linear overhead.

# 8 Linear Perfectly Secure AMPC

In this section, we give the details of the building blocks required for the complete MPC protocol such as reconstruction of degree-$t$ polynomials, and Beaver triple generation. We conclude with the complete MPC protocol which relies on these building blocks, the packed VSS (Section 4) and the verifiable triple secret sharing (Section 7) protocol.

## 8.1 Secret Reconstruction

At the termination of our packed VSS, the secrets are available in Shamir-shared format. We discuss how such sharing can be reconstructed efficiently. We use two standard ways of reconstruction:

**Private reconstruction.** Here, we describe the private reconstruction of a degree-$t$ shared secret to a specified party, say $P^*$. For this, all the parties disclose their shares to $P^*$, who tries to recover the secret as follows. $P^*$ waits for $2t+1$ shares, all of which lie on the same degree-$t$ polynomial. This requires $P^*$ to apply the Reed Solomon (RS) error correction repeatedly in an "online" manner, also known as online error correction (OEC) [17]. If $P^*$ obtains such a polynomial, it is guaranteed to be the correct degree-$t$ polynomial since it agrees with the shares of at least $t+1$ honest parties. The protocol $\Pi_{\mathsf{Rec}}$ appears below.

---

**Protocol 8.1: Private Reconstruction Protocol** – $\Pi_{\mathsf{Rec}}$

---

**Common input:** The description of a field $\mathbb{F}$, $n$ non-zero distinct elements $1, \ldots, n$, identity of a party $P^*$.
**Input:** Parties hold the univariate degree-$t$ sharing $\langle v \rangle$.

1. Each $P_i$ sends its share $\langle v \rangle_i$ to $P^*$ and terminates.
2. For $r = 0, \ldots, t$:
   (a) Upon receiving the first $2t+1+r$ values, $P^*$ looks for a codeword of a polynomial of degree-$t$ with a distance of at most $r$ from the values it received. If there is no such unique codeword, $P^*$ proceeds to the next iteration. Otherwise, it sets $p_r(x)$ to be the unique RS reconstruction.
   (b) If $p_r(i) = \langle v \rangle_i$ holds for at least $2t+1$ parties whose shares were considered during RS reconstruction, then $P^*$ outputs $p_r(0)$ and terminates. Otherwise, it proceeds to the next iteration.

---

**Lemma 8.2.** *Protocol 8.1, $\Pi_{\mathsf{Rec}}$, has a communication complexity of $\mathcal{O}(n \log n)$ bits over point-to-point channels and no broadcast for privately reconstructing a value (i.e., $\mathcal{O}(\log n)$ bits) in constant runtime.*

**Batched public reconstruction.** Naïvely, reconstructing $t+1$ secrets that are Shamir-shared requires $(t+1)n$ private reconstructions (via $\Pi_{\mathsf{Rec}}$), resulting in $\mathcal{O}(n^3 \log n)$ communication[7]. On the other hand the batch reconstruction protocol, first presented in [24], allows parties to robustly

[7] Alternatively, $(t+1)n$ elements of broadcasts. With each party broadcasting $(t+1)$ elements, this results in a cost of $\mathcal{O}(n^4 \log n)$ bits of communication.

reconstruct $t+1$ Shamir-shared values at a cost of communicating $\mathcal{O}(n^2 \log n)$ bits, ensuring an amortized cost of $\mathcal{O}(n \log n)$ bits per reconstruction.

In particular, given $\langle v_0 \rangle, \ldots, \langle v_t \rangle$, parties translate them to $n$ sharings *non-interactively*, say $\langle v'_1 \rangle, \ldots, \langle v'_n \rangle$, using a linear error correcting code, such as Reed-Solomon code which tolerates up to $t$ errors. To be specific, $(v'_1, \ldots, v'_n)$ can be thought of as $n$ points on a $t$-degree polynomial $p(x) = \sum_{i=0}^{t} v_i x^i$. Following this, of the $n$ sharings, one sharing $\langle v'_i \rangle$ is reconstructed towards each party $P_i$ via private reconstruction protocol $\Pi_{\mathsf{Rec}}$ who obtains $v'_i$. At this stage, the parties essentially hold $\langle v_0 \rangle$. Therefore, $n$ instances of private reconstruction enables every party to recover $p(x)$, the polynomial used to share $v_0$, whose coefficients are the desired output. This requires a total communication of $\mathcal{O}(n^2 \log n)$ bits. The protocol $\Pi_{\mathsf{bPubRec}}$ appears below for completeness.

**Protocol 8.3: Batched Public Reconstruction Protocol** – $\Pi_{\mathsf{bPubRec}}$

**Common input:** The description of a field $\mathbb{F}$, $n$ non-zero distinct elements $1, \ldots, n$.
**Input:** Parties hold the univariate degree-$t$ sharings $\langle v_0 \rangle, \ldots, \langle v_t \rangle$.

1. Let $p(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_t x^t$.
2. For each $P_i$, parties locally compute $\langle v'_i \rangle = \langle p(i) \rangle = \langle v_0 \rangle + \langle v_1 \rangle \cdot i + \langle v_2 \rangle \cdot i^2 + \ldots + \langle v_t \rangle \cdot i^t$.
3. For each party $P_i$, parties invoke $\Pi_{\mathsf{Rec}}$ with $\langle v'_i \rangle$ as input to enable $P_i$ to privately reconstruct $v'_i = p(i)$. Note that parties now hold $\langle p(0) \rangle$.
4. For each party $P_i$, parties invoke $\Pi_{\mathsf{Rec}}$ with $\langle p(0) \rangle$ as input to enable $P_i$ to privately reconstruct the polynomial $p(x)$. Upon reconstructing, each $P_i$ outputs the $t+1$ coefficients $v_0, v_1, \ldots, v_t$ of $p(x)$.

**Lemma 8.4.** *Protocol 8.3, $\Pi_{\mathsf{bPubRec}}$, has a communication complexity of $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels and no broadcast for publicly reconstructing $\mathcal{O}(n)$ values (i.e., $\mathcal{O}(n \log n)$ bits) simultaneously and has constant runtime.*

## 8.2 The complete MPC protocol

We now describe our MPC protocol using the packed VSS (Section 4), the verifiable triple sharing and the building blocks which are taken from [22]. Our MPC protocol relies on Beaver's circuit randomization trick [6] and has two phases: (i) Preparing the Beaver triples and input sharing, and (ii) Evaluation using the batched Beaver multiplication.

### 8.2.1 Preparing the Beaver Triples and Input Sharing

This phase is further divided into three tasks. First, using the verifiable triple sharing protocol, each party acting as a dealer is made to Shamir-share the required number of triples $(a, b, c)$ such that $c = ab$ holds. Each party also uses the packed VSS described in Section 4 to share its inputs. Note however that due to the asynchronous nature of the network, parties cannot afford to wait for the triple sharing and input sharing instances of all the parties to terminate. Doing so might result in an endless wait since the corrupt parties might remain silent and not initiate an instance of sharing. Given this, the parties are required to agree on a common set, say $\mathsf{Core}$ of (at least) $n - t$ dealers whose triple sharing as well as input sharing instances will eventually terminate for all the parties. This forms the second task, wherein parties execute an instance of ACS [17]

(Functionality 3.3) to agree on a set of parties whose shared triples and inputs will be considered for subsequent computation. For the parties outside this set, a default sharing of 0 is considered as the input. Finally, once the common set $\mathsf{Core}$ is decided upon and the triple sharing instances of all the dealers in $\mathsf{Core}$ terminate, parties execute the triple extraction protocol which uses the triples shared by these parties and gives as output random triples, not known to any party.

**Verifiable Triple Sharing and Input Sharing.** In this phase, each party shares verified multiplication triples, which will be used in the subsequent phases for extracting random triples unknown to any party. The exact number of triples to be shared by each party depends on the size of the circuit to be computed. We provide the detailed analysis of this in the proof of Theorem 8.11. Towards that end, each party invokes the triple sharing functionality (Functionality 7.1) in parallel to share the required number of triples. Each party invokes the Functionality 4.1 in parallel to share its inputs. However, to ensure termination while accounting for the asynchronous network, parties cannot afford to wait for the triple sharing and input sharing instances of all the parties to terminate. Moreover, waiting for at least $n - t$ parties' instances to terminate before proceeding to triple extraction does not offer a solution. Honest parties might terminate instances in different sequence leading to inconsistency in the subsequent phase. This issue is tackled by the second phase described below.

**Agreement on a Core Set (ACS).** Here, parties execute an instance of ACS [17] (Functionality 3.3) to agree on a common set of at least $n - t$ parties whose triple sharing and input sharing instances are guaranteed to terminate eventually for all the parties. Having agreed on the set, parties proceed to the final task which consumes the triples shared by each party in this set for extracting random triples unknown to any party.

**Triple Extraction.** Our last component is a triple extraction protocol that consumes one (verified) multiplication triple, say $(\langle a_i \rangle, \langle b_i \rangle, \langle c_i \rangle)$, shared by each party $P_i \in \mathsf{Core}$ in the prior stage and extracts $h + 1 - t$ random triples not known to any party, where $h = \lfloor \frac{|\mathsf{Core}|-1}{2} \rfloor$. For simplicity, let $m = |\mathsf{Core}|$ and without loss of generality, we assume $\mathsf{Core} = \{P_1, \ldots, P_m\}$. The protocol incurs a cost of $\mathcal{O}(n^2)$ point to point communication and at a high level, proceeds as follows. First, the parties "transform" the $m$ random shared triples $(\langle a_i \rangle, \langle b_i \rangle, \langle c_i \rangle)$ for each $i \in [m]$ into $m$ correlated triples $(\langle x_i \rangle, \langle y_i \rangle, \langle z_i \rangle)$ for every $i \in [m]$ such that the values $\{x_i, y_i, z_i\}_{i \in [m]}$ 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 $i \in [m]$, 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, $\langle x_i \rangle$ and $\langle y_i \rangle$ for every $i \in \{h+2, \ldots, m\}$ can be computed *non-interactively* by taking linear combination of $\{x_i, y_i\}_{i \in [h+1]}$. Following this, $\langle z_i \rangle$ for every $i \in \{h+2, \ldots, m\}$ is computed using Beaver's trick where the inputs are $\langle x_i \rangle$ and $\langle y_i \rangle$ and the triple $(\langle a_i \rangle, \langle b_i \rangle, \langle c_i \rangle)$. Clearly, if $P_i$ is corrupt then $x_i, y_i, z_i$ is known to the adversary as claimed. To conclude, we note that triple extraction reduces to running a batch of $\mathcal{O}(h)$ Beaver multiplications which requires $\mathcal{O}((nh + n^2) \log n)$ bits communication using $\Pi_{\mathsf{bPubRec}}$. The formal description

appears in Protocol 8.5.

**Protocol 8.5: Triple Extraction** – $\Pi_{\mathsf{tripleExt}}$

**Common input:** The description of a field $\mathbb{F}$, a set $\mathsf{Core} \subseteq \mathcal{P}$ such that $m = |\mathsf{Core}|$, $m = 2h + 1$ non-zero distinct elements $1, \ldots, m$ and $h+1-t$ non-zero distinct elements $\beta_1, \ldots, \beta_{h+1-t}$. Without loss of generality, assume $\mathsf{Core} = \{P_1, \ldots, P_m\}$.
**Input:** Parties hold the degree-$t$ shared triples $(\langle a_i \rangle, \langle b_i \rangle, \langle c_i \rangle)$ for every $i \in [m]$ such that $(a_i, b_i, c_i)$ is known to party $P_i$.

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

**Lemma 8.6.** *Protocol 8.5, $\Pi_{\mathsf{tripleExt}}$, has a communication complexity of $\mathcal{O}((nh + n^2) \log n)$ bits over point-to-point channels and no broadcast for sharing $h+1-t$ random multiplication triples in constant runtime.*

### 8.2.2 Batched Beaver Multiplication

This corresponds to the second phase of our MPC protocol, which uses the degree-$t$ shared multiplication triples computed in the prior phase to evaluate the multiplication gates in the circuit via Beaver multiplication in a batched manner. This protocol relies on the well known technique of Beaver's circuit randomization [6], which, given a pre-computed $t$-shared random and private multiplication triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, reduces the computation of $\langle xy \rangle$ from $\langle x \rangle$ and $\langle y \rangle$ to two public reconstructions. Towards this, parties first locally compute $\langle d \rangle = \langle x \rangle - \langle a \rangle$ and $\langle e \rangle = \langle y \rangle - \langle b \rangle$, followed by public reconstruction of $d$ and $e$. Since $z = xy = ((x-a)+a)((y-b)+b) = (d+a)(e+b) = de + db + ea + ab$, parties can locally compute $\langle z \rangle = \langle xy \rangle$ using the shared multiplication triple and the publicly reconstructed values $d$ and $e$. Specifically, parties locally compute $\langle xy \rangle = de + d\langle b \rangle + e\langle a \rangle + \langle c \rangle$.

To leverage the efficiency benefits offered by the batch public reconstruction protocol, the protocol handles a batch of $l$ multiplications together, each requiring 2 reconstructions. The $2l$ public reconstructions are thus batched together in groups of $t+1$ to invoke $\Pi_{\mathsf{bPubRec}}$ and ensure an amortized communication complexity of $\mathcal{O}(n \log n)$ bits per reconstruction. The resultant communication complexity of $\Pi_{\mathsf{bBeaver}}$ for handling $l$ multiplications is $\mathcal{O}((n^2 + nl) \log n)$. The formal description appears in Protocol 8.7.

**Protocol 8.7: Batched Beaver Multiplication** – $\Pi_{\mathsf{bBeaver}}$

**Input:** Parties hold $l$ degree-$t$ shared triples $(\langle a_i \rangle, \langle b_i \rangle, \langle c_i \rangle)$ for every $i \in [l]$ and $l$ degree-$t$ shared pairs of values $(\langle x_i \rangle, \langle y_i \rangle)$ to be multiplied.

1. For each $i \in [l]$, parties locally compute $\langle d_i \rangle = \langle x_i \rangle - \langle a_i \rangle$ and $\langle e_i \rangle = \langle y_i \rangle - \langle b_i \rangle$.
2. Let $2l = k(t+1)$. Parties execute $k$ parallel instances of $\Pi_{\mathsf{bPubRec}}$ and publicly reconstruct $\{d_i, e_i\}$ for every $i \in [l]$.
3. For each $i \in [l]$, parties locally compute $\langle z_i \rangle = \langle x_i y_i \rangle = d_i e_i + d_i \langle b_i \rangle + e_i \langle a_i \rangle + \langle c_i \rangle$.

---

**Lemma 8.8.** *Protocol 8.7, $\Pi_{\mathsf{bBeaver}}$, has a communication complexity of $\mathcal{O}((ln+n^2)\log n)$ bits over point-to-point channels and no broadcast for the multiplication of $l$ pairs of shared values and has constant runtime.*

## 8.3 The MPC Protocol

We first describe the MPC functionality, followed by the complete protocol using the building blocks described above. We subsequently provide the proof of security and the communication complexity analysis of the protocol.

---

**Functionality 8.9: AMPC – $\mathcal{F}_{\mathsf{AMPC}}$**

---

The functionality is parameterized by a set of corrupted parties $I \subseteq [n]$. Initialize the sets $S, H, I' = \phi$. Initialize $x_i = 0$ for every $i \in I$.
**Input:** Each $P_i$ holds input $x_i \in \mathbb{F} \cup \{\perp\}$.
**Common Input:** An $n$-party function $f(x_1, \ldots, x_n)$.

1. Upon receiving $(\mathsf{Input}, j, x_j)$ from an honest party $P_j$, if $j \notin H$ then add $j$ to $S$.
2. Receive from the adversary, the sets $H \subset [n] \setminus I$ and $I' \subseteq I$ such that $|H| \leq |I'| \leq t$. Also receive a set of inputs $\{(\mathsf{Input}, i, x_i)\}_{i \in I'}$.
3. If $|S| < n - t$, then for each $P_j$ with $j \in H$, the functionality sets $x_j = 0$ and updates $S = S \setminus H$.
4. If $|S \cup I'| \geq n - t$, then compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and send $y_i$ to $P_i$ for every $i \in [n]$ and terminate.

---

**Protocol 8.10: AMPC – $\Pi_{\mathsf{AMPC}}$**

---

**Common input:** The description of a circuit, the field $\mathbb{F}$, $n$ non-zero distinct elements $1, \ldots, n$ and a parameter $h$ where $n - t = 2h + 1$. Let $m = \lceil \frac{C}{h+1-t} \rceil$.
**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 $\lceil \frac{m}{(t/2+1)^2} \rceil$ instances of Protocol 7.2 (Section 7) in a batched manner each with $(t/2+1)^2$ triples.
2. **(Input sharing)** Each party $P_i$ holding $k_i$ inputs to the circuit executes the VSS protocol (Functionality 4.1) in a batched manner, packing $\lceil \frac{k_i}{t/2+1} \rceil$ inputs in one instance.

3. **(ACS Execution)** Parties invoke ACS protocol (Functionality 3.3) to agree on a set Core of at least $n - t$ parties whose instances of triple sharing and input sharing will terminate eventually all the honest parties. Let $(\langle a_i^j \rangle, \langle b_i^j \rangle, \langle c_i^j \rangle)$ for $j \in [m]$ denote the triples shared by $P_i \in$ Core. The input sharing for the parties outside Core is take as default sharing of 0.
4. **(Beaver Triple Extraction)** Parties execute $m$ instances of the triple extraction protocol, $\Pi_{\mathsf{tripleExt}}$ (Protocol 8.5), with Core as the common input and additionally $(\langle a_i^j \rangle, \langle b_i^j \rangle, \langle c_i^j \rangle)$ for every $P_i \in$ Core as the input for the $j^{\text{th}}$ instance. Let $(\langle \mathbf{a}_i \rangle, \langle \mathbf{b}_i \rangle, \langle \mathbf{c}_i \rangle)$ for $i \in [C]$ denote the random multiplication triples generated.

**(Circuit computation:)**

1. **(Linear Gates)** Parties locally apply the linear operation on their respective shares of the inputs.
2. **(Multiplication Gates)** Let $(\langle \mathbf{a}_i \rangle, \langle \mathbf{b}_i \rangle, \langle \mathbf{c}_i \rangle)$ be the multiplication triple associated with the $i^{\text{th}}$ multiplication gate with shared inputs $(\langle x_i \rangle, \langle y_i \rangle)$. Parties invoke the batched Beaver protocol, $\Pi_{\mathsf{bBeaver}}$ (Protocol 8.7), with $\{\langle x_i \rangle, \langle y_i \rangle, \langle \mathbf{a}_i \rangle, \langle \mathbf{b}_i \rangle, \langle \mathbf{c}_i \rangle\}$ for all gates $i$ at the same layer of the circuit and obtain the corresponding $\langle z_i \rangle$ as the output sharing for every gate $i$.
3. **(Output)** For each output gate $j$ with the associated sharing $\langle v_j \rangle$, parties execute private reconstruction protocol, $\Pi_{\mathsf{Rec}}$ (Protocol 8.1), towards every party $P_i$ who is supposed to receive the output $v_j$.

---

**Theorem 8.11.** *Let $n \geq 4t + 1$. Protocol 8.10 securely computes Functionality 8.9 in the Functionality (4.1,7.1, 3.3)-hybrid in the presence of a malicious adversary controlling at most t parties. It has a communication complexity of $\mathcal{O}((Cn + Dn^2 + n^5) \log n)$ bits over point-to-point channels and $\mathcal{O}(n^3 \log n)$ bits of broadcast for evaluating a circuit with $C$ gates and depth $D$. Each party broadcasts at most $\mathcal{O}(n^2 \log n)$ bits.*

*Proof.* The circuit evaluation requires $C$ random shared multiplication triples. We analyze the cost of the two phases of the MPC protocol separately.

*Beaver triple generation and Input sharing.* Note that one instance of the triple extraction protocol (Protocol 8.5) extracts $h+1-t$ random triples simultaneously, where $h = \lceil \frac{|\mathsf{Core}|-1}{2} \rceil$. Given that Core is of size at least $n-t$, we have that $h+1-t = \mathcal{O}(n)$. Hence, we require $\mathcal{O}(C/n)$ instances of the triple extraction, which incurs a cost of $\mathcal{O}((Cn + n^2) \log n)$ bits of communication. Further, each instance of the triple extraction protocol consumes one verified triple shared by each party. Hence, each party is required to share $\mathcal{O}(C/n)$ multiplication triples. Since our verifiable triple sharing protocol packs $\mathcal{O}(n^2)$ triples in one instance, it requires each party to execute $\mathcal{O}(C/n^3)$ instances of the protocol in a batched manner, which results in a cost of $\mathcal{O}((C + n^3) \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits of broadcast per dealer. That is, the total cost incurred is $\mathcal{O}((Cn + n^5) \log n)$ using the protocol of [13] to instantiate broadcast. Moreover, the ACS protocol has a cost of $\mathcal{O}(n^5 \log n)$ (as discussed in Section 2.3), resulting in a communication complexity of $\mathcal{O}((Cn + n^5) \log n)$ for this phase.

*Circuit evaluation.* Here, parties evaluate batches of multiplication gates at the same level in the circuit by invoking the batched Beaver multiplication protocol (Protocol 8.7). Given $C_i$ is the number of gates per level of the circuit, this stage incurs a cost of $\mathcal{O}(C_i \cdot n \log n + n^2 \log n)$ bits communication over point-to-point channels. Thus we have that this phase requires a total of

$\sum_{i=1}^{D} \mathcal{O}(C_i \cdot n \log n + n^2 \log n) = \mathcal{O}(Cn \log n + Dn^2 \log n)$ bits communication over point-to-point channels.

We now show the simulation. The simulator $\mathcal{S}$ is as follows:

1. Upon activation, invoke the adversary $\mathcal{A}$.
2. *Beaver triple generation and Input Sharing*
   (a) On behalf of each honest party $P_j$, the simulator chooses $m$ random multiplication triples and simulates the Functionality 7.1 for the case of an honest dealer by choosing the appropriate bivariate polynomials.
   (b) On behalf of each honest party $P_j$ holding an input to the circuit, the simulator chooses a value uniformly at random and simulates the Functionality 4.1 for the case of an honest dealer.
   (c) For every $i \in I$, simulate Functionality 7.1 and 4.1 as a functionality would run it. If the simulation of Functionality 4.1 terminates, then from the shares of the honest parties, the simulator reconstructs the input $x_i$.
   (d) Simulate the Functionality 3.3: When the adversary delivers the shares to an honest party $P_j$ in the simulation of instance of Functionality 7.1 and Functionality 4.1 (if any) corresponding to some party $P_k$, simulate $P_j$ sending $(\mathsf{record}, j, k)$. Listen to the $\mathsf{record}$ messages sent by the adversary. Simulate the functionality sending a set $S$ upon receiving the command $\mathsf{receiveS}()$ from the adversary. When $S \neq \perp$, let $\mathsf{Core} = S$.
   (e) The simulator has the shares of honest parties corresponding to the triples shared by each party in $S$ when $S \neq \perp$, and hence can run Protocol 8.5 as honest parties would. Specifically, it can simulate all the communication from honest parties including that among honest parties as in the protocol.
3. *Circuit evaluation*
   (a) The simulator holds shares of the honest parties corresponding to the inputs of each party in $\mathsf{Core}$, as well as the shares corresponding to the default input 0 of the remaining parties.
   (b) On behalf of each honest party, the simulator computes the linear operations on the shares of honest parties as in the protocol.
   (c) For each multiplication gate, the simulator holds the shares of honest parties and can run Protocol 8.7 as honest parties would run it.
   (d) The simulator constructs a set $H$ of all the simulated honest parties not in $\mathsf{Core}$. It also constructs a set $I' = \mathsf{Core} \cap I$. It invokes Functionality 8.9 with the sets $H, I'$ and the extracted inputs $\{(\mathsf{Input}, i, x_i)\}_{i \in I'}$.
   (e) It receives from the functionality the outputs $y_i$ for every $i \in I$.
   (f) The simulator computes the shares of the honest parties to be used in the simulation of Protocol 8.1 using the output $y_i$ and the shares of the corrupt parties (which can be computed from the view of the simulated honest parties). The simulator runs Protocol 8.1 as honest parties would using these shares.
   (g) When the adversary allows successful reconstruction of the output towards an honest party $P_j$ in the simulated protocol, the simulator delivers the output of Functionality 8.9 in the ideal protocol.

We now show that the view of the adversary is indistinguishable in both the executions. We do

this using a sequence of hybrids.

**Hybrid$_0$:** Execution of $\Pi_{\mathsf{AMPC}}$ in the real world.

**Hybrid$_1$:** In this hybrid, the simulator simulates the execution of the output gates as described. The only change is that the simulator computes the inputs of the corrupt parties in Core, and the sets $H$ and $I'$ and invokes the Functionality 8.9 with these inputs. Note that for the output $y_i$ towards a corrupt $P_i$, the shares of honest parties in the reconstruction are completely determined by the shares of the corrupt parties and the output $y_i$. Hence, the shares computed by $\mathcal{S}$ are identical to the real shares of the honest parties. The distributions of **Hybrid$_0$** and **Hybrid$_1$** are identical.

**Hybrid$_2$:** Here, the simulator simulates the execution of the addition and multiplication gates. Since $\mathcal{S}$ knows the shares held by honest parties, it can simulate the addition and multiplication gates as honest parties would in the real protocol. The distributions of **Hybrid$_1$** and **Hybrid$_2$** are identical.

**Hybrid$_3$:** In this hybrid, $\mathcal{S}$ simulates the execution of triple extraction as described. Since the shares of honest parties are held by the simulator, the two distributions are identical.

**Hybrid$_4$:** In this hybrid, the simulator simulates the Functionality 3.3. By the security of ACS, we have that the two distributions are indistinguishable.

**Hybrid$_5$:** In this hybrid, $\mathcal{S}$ simulates the invocations of Functionality 7.1 and Functionality 4.1 by choosing triples and inputs respectively uniformly at random on behalf of the honest parties. The two hybrids differ only in the manner that $\mathcal{S}$ uses the random triples and inputs of honest parties in **Hybrid$_4$**, whereas it samples random values in **Hybrid$_5$**. The shares of the corrupt parties are distributed uniformly at random in both the hybrids. Due to Theorem 7.3 and Theorem 4.4, we have that the distributions in **Hybrid$_4$** and **Hybrid$_5$** are indistinguishable.

Note that **Hybrid$_5$** is the execution between $\mathcal{S}$ and $\mathcal{A}$ in the ideal world. Therefore, we conclude that the distributions in **Hybrid$_0$** and **Hybrid$_5$** corresponding to the executions in the real and ideal worlds respectively are indistinguishable.

Note that in the simulated execution, the circuit evaluation is performed considering the inputs shared by parties in Core and with default inputs for the remaining parties. The simulator invokes the functionality in the ideal world with the sets $H$ and $I'$ such that the inputs of the same set of parties in Core are considered during computation of the function, ensuring that the output of the honest parties is the same in both the executions. □

## References


[1] Abraham, I., Asharov, G., Patil, S., Patra, A.: Asymptotically free broadcast in constant expected time via packed vss. In: Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I. pp. 384–414. Springer (2023) 17

[2] Abraham, I., Asharov, G., Patil, S., Patra, A.: Detect, pack and batch: Perfectly-secure mpc with linear communication and constant expected time. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 251–281. Springer (2023) 1, 2, 5

[3] Abraham, I., Asharov, G., Patra, A., Stern, G.: Perfectly secure asynchronous agreement on a core set in constant expected time. IACR Cryptol. ePrint Arch. p. 1130 (2023), https://eprint.iacr.org/2023/1130 10

[4] Abraham, I., Dolev, D., Stern, G.: Revisiting asynchronous fault tolerant computation with optimal resilience. Distributed Computing **35**(4), 333–355 (2022) 1

[5] Applebaum, B., Kachlon, E.: Conflict checkable and decodable codes and their applications. IACR Cryptol. ePrint Arch. p. 627 (2023) 2, 3, 8, 9

[6] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference (1991) 9, 43, 45

[7] Beerliová-Trubíniová, Z., Hirt, M.: Simple and efficient perfectly-secure asynchronous mpc. In: Advances in Cryptology – ASIACRYPT 2007. pp. 376–392. Berlin, Heidelberg (2007) 1

[8] Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Proceedings of the 5th Conference on Theory of Cryptography. p. 213–230. TCC'08, Springer-Verlag, Berlin, Heidelberg (2008) 1

[9] Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Theory of Cryptography Conference (2008) 2

[10] Ben-Or, M.: Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In: PODC (1983) 2

[11] Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: ACM symposium on Theory of computing (1993) 1, 3, 4

[12] Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing. p. 183–192. PODC '94, Association for Computing Machinery, New York, NY, USA (1994) 1, 3

[13] Bracha, G.: An asynchronous [(n-1)/3]-resilient consensus protocol. In: PODC. pp. 154–162 (1984) 3, 5, 15, 47

[14] Bracha, G.: Asynchronous byzantine agreement protocols. Inf. Comput. **75**(2), 130–143 (1987) 41

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

[16] Canetti, R.: Studies in secure multiparty computation and applications (1996), https://www.wisdom.weizmann.ac.il/~oded/PSX/ran-phd.pdf 1, 17

[17] Canetti, R.: Studies in secure multiparty computation and applications. Ph.D. thesis, Citeseer (1996) 4, 5, 10, 16, 42, 43, 44

[18] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS (2001) 13

[19] Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II 35. pp. 3–22. Springer (2015) 13, 15

[20] Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Advances in Cryptology–CRYPTO 2015. pp. 3–22 (2015) 14

[21] Choudhury, A., Hirt, M., Patra, A.: Unconditionally secure asynchronous multiparty computation with linear communication complexity. In: DISC (2013) 3

[22] Choudhury, A., Patra, A.: An efficient framework for unconditionally secure multiparty computation. IEEE Transactions on Information Theory (2016) 3, 6, 9, 10, 18, 43

[23] Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Advances in Cryptology - EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 445–465. Springer (2010) 1

[24] Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Annual International Cryptology Conference (2007) 42

[25] Damgård, I., Schwartzbach, N.I.: Communication lower bounds for perfect maliciously secure mpc. Cryptology ePrint Archive (2020) 2

[26] Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional mpc with guaranteed output delivery. In: CRYPTO (2019) 2

[27] Patra, A., Choudhary, A., Rangan, C.P.: Communication efficient perfectly secure VSS and MPC in asynchronous networks with optimal resilience. In: Progress in Cryptology - AFRICACRYPT 2010. vol. 6055, pp. 184–202. Springer (2010) 1, 2

[28] Patra, A., Choudhury, A., Pandu Rangan, C.: Efficient asynchronous verifiable secret sharing and multiparty computation. Journal of Cryptology **28**(1), 49–109 (2015) 1

[29] Prabhu, B., Srinathan, K., Rangan, C.P.: Asynchronous unconditionally secure computation: An efficiency improvement. In: Progress in Cryptology — INDOCRYPT 2002. pp. 93–107. Berlin, Heidelberg (2002) 1

[30] Srinathan, K., Pandu Rangan, C.: Efficient asynchronous secure multiparty distributed computation. In: INDOCRYPT 2000. pp. 117–129 (2000) 1