Statistical MPC with a Constant Communication Overhead

Ashish Choudhury[1], Ivan Damgård[2], Shravani Patil[3], and Arpita Patra[3]

[1] IIIT Bangalore, India;
[2] Aarhus University, Denmark;
[3] IISc Bangalore, India

Abstract. We study the feasibility of constant communication-overhead statistically-secure MPC tolerating malicious adversaries in the synchronous setting and guaranteed output delivery (GOD). We propose a protocol with $n = 3t + 1$ for SIMD circuits with a communication overhead of $\mathcal{O}(1)$ and round complexity $\mathcal{O}(D)$ for circuit depth D, where n is the number of parties and t is the maximum number of corruptions. Moreover, our protocol achieves GOD. This is the *first* MPC protocol with above characteristics.
To design our protocol, we give two major contributions. Our first technical contribution is a verifiable secret sharing (VSS) protocol that achieves constant per-secret overhead via two-dimensional packing. Our second contribution is a constant-overhead degree-reduction protocol that converts a high-degree packed sharing into a lower-degree packed sharing.

1 Introduction

The communication complexity of interactive protocols is one of their most important efficiency measures. Consequently, a huge amount of research has been devoted to characterizing the communication complexity of various distributed tasks (e.g., Byzantine agreement [26,10], and general secure multiparty computation [39,40,29,14,28,25,22,21,19,24]) under different security models.

We focus on the communication complexity of protocols for multiparty computation that achieve information-theoretic security with negligible error probability (aka. statistical security) in the presence of an active (aka. Byzantine or malicious), adaptive, computationally unbounded, rushing adversary, in *synchronous* communication setting. Moreover, we aim for the highest notion of security, namely *guaranteed output delivery* (GOD), where the honest parties (namely the parties not under the control of the adversary) are guaranteed to obtain the output of the computation, irrespective of the adversary's strategy.

It is known that the optimal resilience for perfect and statistical security is $t < n/3$ [8,12] and $t < n/2$ [43], where n is the number of protocol participants and t is the maximum number of corruptions. The traditional approach for designing information-theoretic protocols is to perform secret-shared evaluation of the arithmetic circuit, representing the underlying computation, as pioneered by [8]. The secret-sharing used is linear, allowing the parties to non-interactively

evaluate the linear gates, while interaction is needed to evaluate the non-linear (multiplication) gates and so the focus is rightly put on efficiently evaluating the non-linear gates. There are two families of protocols which have emerged. The first category consists of protocols which are communication-efficient, but slow and requires $\Omega(D + \text{poly}(n))$ rounds of communication [6,33,35], where D is the multiplicative depth of the underlying circuit. The second category of protocols are fast, requiring only $\mathcal{O}(D)$ communication rounds, but are less communication efficient [8,4,16]. The former category of protocols are based on *non-robust* primitives, coupled with player-elimination [36] and dispute control [7] techniques. On the other hand, the latter category deploy *robust* primitives, typically based on *verifiable secret sharing* (VSS) [13]. As elaborated in [1], it is important to optimize both communication efficiency as well as round complexity, especially if the parties are evaluating circuits with a "shallow" depth. Hence, we aim to simultaneously achieve the best of **both** families, namely communication efficiency, as well as $\mathcal{O}(D)$ communication rounds.

For protocols computing an arithmetic circuit C of size $|C|$, [23] shows that perfectly-secure MPC with $n = 3t + s$, with $s > 0$ requires communication complexity $\Omega(\frac{n|C|}{s})$ bits. Thus, the per-gate communication is $\Omega(n/s)$ bits, and we refer to this as the *overhead*. For $s = 1$, the overhead is $\Omega(n)$ and the protocols of [6,34,1] provide the matching upper bound. When $s = \Theta(n)$, the optimal overhead is a constant. In this case, a matching upper bound protocol with GOD and $\mathcal{O}(D)$ rounds can be obtained by modifying the protocol of [1]. These observations almost settle the communication complexity status for perfect security with GOD and $\mathcal{O}(D)$ rounds (intermediate cases like $s = \sqrt{n}$ are not covered).

The goal of this work is to consider the corresponding question for *statistical* security. On the lower bound side, [19] shows that for $n = 2t+s$ with $s > 0$, there exists a function f with circuit C such that in any statistically-secure protocol for computing f against t passive corruptions, the average total communication complexity is $\Omega(n|C|/s)$ bits.[4] So, the overhead must be $\Omega(n)$ for $s = 1$ and matching upper bounds are presented in [9,35] against Byzantine corruptions. So the next natural and important question is:

> can we design a constant-overhead statistical MPC protocol with GOD **and** $\mathcal{O}(D)$ rounds with $n = 2t + s$ for $s = \Theta(n)$?

1.1 Our Results

We answer the above question affirmatively for $s = t+1$ and show the following.

Theorem 1.1 (Informal). *Let $n = 3t + 1$. Then there exists a statistically-secure MPC with GOD for securely evaluating any arithmetic circuit in an SIMD*

[4] The bound in [19] is in terms of the size of the input to f. However, since f has linear size circuits, we get the result cited. Moreover, since all known MPC constructions do "the same" for any circuit, we expect the overhead to be $\Theta(n/s)$ in general.

fashion $\Theta(n^7)$ times in parallel, with a constant communication overhead per gate. The protocol requires $\mathcal{O}(D)$ expected rounds.[5]

We assume that C has an SIMD structure, i.e., it computes the same function $\Theta(n^7)$ number of times in parallel. The need for this stems from the fact that we aim to design *robust* primitives for our MPC protocol (such as VSS, secret-shared multiplication, etc) with $\mathcal{O}(1)$ communication overhead. The constant overhead is achieved via amortization, only when we work with large number of inputs. However, we remark that even though our protocol is designed for SIMD structured circuits, it can be applied to a general circuit via the compilation process of [18]. The compilation uses network routing techniques and produces a new SIMD structured circuit computing the original function, of size a factor $\log(|C|)$ larger than $|C|$. In addition there may be an additive term that depends on the depth of C, but it only occurs in very unstructured cases where C is a large depth circuit that sends values from the start of the computation to (almost) all of its layers. For layered circuits or low-depth circuits, for instance, this does not happen. Thus, the cost of the compilation is usually a factor $\log(|C|)$. Our protocol will still have the property that the overhead *does not* grow with n.

Verifiable Secret Sharing (VSS). For our MPC protocol, we first construct a VSS protocol. Informally, a VSS protocol allows a designated dealer to verifiably share its inputs (called secrets) among the parties without revealing any additional information about the inputs (privacy), such that the inputs can be later reconstructed robustly (correctness). The verifiability guarantees that even a potentially corrupt dealer "consistently" shares some well-defined inputs, which can be later reconstructed robustly (commitment).

Our VSS protocol with $n = 3t+1$ incurs a constant communication overhead per secret **and** requires a constant expected number of rounds, provided there are $\Theta(n^5)$ inputs. This is the *first* statistical VSS with the above properties and is of independent interest. To incur a constant communication overhead, our VSS generates polynomial-based packed secret sharings, where $\Theta(n)$ secrets are hidden and secret-shared through a single polynomial. Moreover, for robust reconstruction, shares are authenticated through pair-wise information-theoretic MACs. To maintain a constant overhead, a single tag is generated on "multiple" shares. This means that the packed secret sharings are batched and the authentication tags are computed on a batch of shares instead of a single value.

Robust Degree Reduction Over Authenticated Packed Secret-Sharings. For evaluating multiplication gates, we generate authenticated packed Beaver multiplication triples through our VSS. An important building block for the triple generation and evaluating multiplication gates is *degree-reduction* that allows a bunch of packed secret sharings, shared with a degree d_1, to be securely reshared with a lower degree d_2 (i.e. $d_1 > d_2$). [23] demonstrated a robust method for this, which is based on the properties of RS error correction. Unfortunately, the method *does not* extend to our setting, due to lack of sufficiently many

[5] For $n = 3t+1$, we can securely realize broadcasts over point-to-point channels using constant expected round broadcast protocols.

shares to error-correct up to t errors on polynomials of degree $t + \Theta(n)$. So, the degree-reduction constitutes another technical contribution of this work.

1.2 Other Related Works

Prior works to achieve a constant overhead can be categorized into two classes.

- The first approach is to use player-virtualization technique [11], first used in [18], which achieves a constant overhead per gate for SIMD circuits with statistical security for any $t < (\frac{1}{2} - \epsilon)n$, where $\epsilon > 0$. The idea here is to first design an "outer" protocol involving virtual parties, with a "low" corruption threshold and then emulating each virtual party by running any existing statistically-secure "inner protocol" with honest majority. Each virtual party is emulated by a constant number of real parties and the construction of virtual parties is done deterministically, using expander graphs [17].[6] The round complexity of the resultant protocol is $\Omega(D + \text{poly}(n))$. The follow-up work of [30] improves the underlying constants of [18], but settles for a weaker notion of security with abort. Using the generic protocol transformation of [37], one can boost the security of [30] and get GOD, without affecting the constant overhead per gate. However, the transformed protocol will require $\Omega(D + \text{poly}(n))$ rounds.
- The works of [31,32] present statistically-secure protocols with a constant overhead per gate for any $t < (\frac{1}{2} - \epsilon)n$. Unlike SIMD circuits, the parties need to evaluate a single copy of the circuit, provided the circuit is "non-pathological" in the sense that it can be decomposed into sub-circuits, either in parallel or sequentially. However, the protocols achieve security with abort. One can again boost the security to GOD using the transformation of [37], albeit with $\Omega(D + \text{poly}(n))$ rounds.

2 Technical Overview of Our VSS and Multiplication

We follow the standard paradigm of secret-sharing based MPC in the preprocessing model [5], where the parties generate secret-shared random multiplication-triples in a preprocessing phase, which are later utilized to efficiently evaluate the given circuit. We envision a scenario where the same circuit needs to be evaluated in *parallel* "multiple" times. Let p be the *packing parameter* where $p = \Theta(n)$. The idea is then to use packed-secret-sharing (PSS) [29], where p number of values are secret-shared through a *single* polynomial. Namely, a vector $\mathbf{s} = (s^{(1)}, \ldots, s^{(p)}) \in \mathbb{F}^p$ is said to be degree-ℓ packed-secret-shared where $\ell \geq p-1$, denoted as $\langle \mathbf{s} \rangle_\ell$, if there exists a degree-ℓ polynomial $f(X)$ over $\mathbb{F}$ with $f(-k) = s^{(k)}$ for each $k \in \{1, \ldots, p\}$ and each $P_i \in \mathcal{P}$ has a share $s_i = f(i)$ for $\mathbf{s}$. Note that PSS has the linearity property and $c_1 \cdot \langle \mathbf{a} \rangle_d + c_2 \cdot \langle \mathbf{b} \rangle_\ell = \langle c_1 \cdot \mathbf{a} + c_2 \cdot \mathbf{b} \rangle_\ell$ holds for vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^p$ and publicly-known constants c_1 and c_2; here $c_1 \cdot \mathbf{a}$

[6] Even though [18] is designed with perfect security, one can get statistical security by emulating the virtual parties using a statistically-secure inner protocol.

denotes the vector $\mathbf{a}$ with each component being multiplied by c_1. As each party gets a *single* element from $\mathbb{F}$ as its share for the *entire* $\mathbf{s}$, the "overhead" here is $\mathcal{O}(1)$. If $\ell = t + p - 1$ and if $f(X)$ is chosen randomly (with the constraints that $f(-k) = s^{(k)}$), then $\mathbf{s}$ remains secret from the adversary. So instead of evaluating the circuit *once* as done in the standard secret-sharing based MPC protocols, one can evaluate the circuit p times in parallel in an SIMD fashion over PSS.

2.1 Authenticated Packed Secret Sharing (APSS)

With $n = 3t+1$, the first problem we face is that we *cannot* robustly reconstruct $\mathbf{s}$ from $\langle \mathbf{s} \rangle_\ell$ using error-correction for any $\ell > t$. As a way out, we opt for authenticated-packed-secret-sharing (APSS), where each share is authenticated by pair-wise information-theoretically secure MACs. Namely, let $\mathbf{s} \in \mathbb{F}^p$. Then we augment $\langle \mathbf{s} \rangle_\ell$ to ${}^A\langle \mathbf{s} \rangle_\ell$, where each share s_i is further authenticated as follows: every party P_j will have a MAC-key K_{ji} corresponding to P_i and party P_i will hold a MAC-tag τ_{ij} for the share s_i, computed under the key K_{ji}. Given ${}^A\langle \mathbf{s} \rangle_\ell$, any designated P_j can reconstruct $\mathbf{s}$, provided $\ell \leq n - t - 1$: every P_i sends s_i and τ_{ij} to P_j, who verifies the authenticity of the share s_i, using its key K_{ji} and the received tag. Once P_j receives $\ell + 1$ verified shares, it can reconstruct $\mathbf{s}$.

The standard information-theoretically secure MAC works as follows: a random pair $\mathsf{K} = (u, v)$ is selected as the MAC-key, where $u, v \in \mathbb{F}$. The MAC-tag on a value $s \in \mathbb{F}$ is defined as $\mathsf{MAC}_{\mathsf{K}}(s) = \tau \stackrel{\mathsf{def}}{=} u \cdot s + v$. Instead of authenticating *single* field element, we will be authenticating vectors of field elements, as done in [9,38]. Here to authenticate a vector $\mathbf{s} \in \mathbb{F}^L$, a random pair $\mathsf{K} = (\boldsymbol{\mu}, v)$ is selected as the MAC-key, where $v \in \mathbb{F}$ and $\boldsymbol{\mu} \in \mathbb{F}^L$. The MAC-tag on $\mathbf{s} \in \mathbb{F}^L$ is then defined as $\mathsf{MAC}_{\mathsf{K}}(\mathbf{s}) = \tau \stackrel{\mathsf{def}}{=} \boldsymbol{\mu} \cdot \mathbf{s} + v$, where $\cdot$ here denotes the inner-product operation. Note that we can *reuse* $\boldsymbol{\mu}$ for authenticating multiple vectors by just sampling a random v every time. Namely, if $\mathbf{s}, \mathbf{s}'$ are two vectors to be authenticated, then one can use the keys $\mathsf{K} = (\boldsymbol{\mu}, v)$ and $\mathsf{K}' = (\boldsymbol{\mu}, v')$ for authenticating $\mathbf{s}$ and $\mathbf{s}'$ respectively. Consequently, once $\boldsymbol{\mu}$ is selected, the key-size for subsequent authentications will be *independent* of L. Reusing $\boldsymbol{\mu}$ further permits the MAC (and hence APSS) to be linear. That is, let $\tau = \mathsf{MAC}_{\mathsf{K}}(\mathbf{s})$ and $\tau' = \mathsf{MAC}_{\mathsf{K}'}(\mathbf{s}')$. Then the MAC-tag and the corresponding MAC-key for any publicly-known linear combination of $\mathbf{s}$ and $\mathbf{s}'$ can be computed from τ, τ and K, K' respectively.

Consider two-dimensional vectors $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$, where each $\mathbf{s}^{(k)} \in \mathbb{F}^p$. Then a degree-$\ell$ APSS of $\mathbf{S}$ consists of degree-ℓ PSS $\langle \mathbf{s}^{(k)} \rangle_\ell$ of each vector $\mathbf{s}^{(k)}$ in $\mathbf{S}$. Additionally, the share-vector of every party will be authenticated with information-theoretic MAC. In more detail, let $\mathbf{s}_i \in \mathbb{F}^M$ be the vector of shares of P_i for $\langle \mathbf{s}^{(1)} \rangle_\ell, \ldots, \langle \mathbf{s}^{(M)} \rangle_\ell$. Then every party P_j will hold a MAC-key $\mathsf{K}_{ji} = (\boldsymbol{\mu}_{ji}, v_{ji})$ and P_i will hold the MAC-tag $\tau_{ij} = \boldsymbol{\mu}_{ji} \cdot \mathbf{s}_i + v_{ji}$. The notation ${}^A\langle \mathbf{S} \rangle_\ell$ will denote an APSS of $\mathbf{S}$, as depicted in Fig 1. We will say that ${}^A\langle \mathbf{S} \rangle_\ell$ and ${}^A\langle \mathbf{S}' \rangle_\ell$ are *key-consistent* if for every P_i, P_j, party P_j has the same $\boldsymbol{\mu}_{ji}$ for authenticating P_i's share-vectors for both ${}^A\langle \mathbf{S} \rangle_\ell$ and ${}^A\langle \mathbf{S}' \rangle_\ell$, thus ensuring linearity.

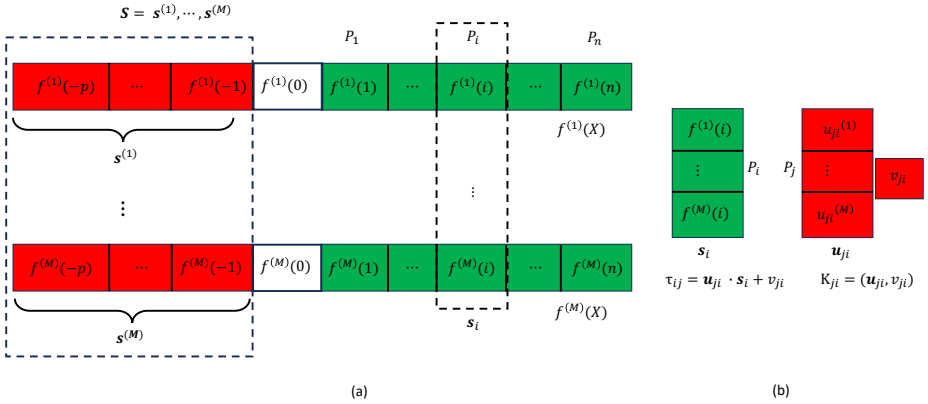


Fig. 1: Pictorial illustration of ${}^{A}\langle \mathbf{S} \rangle_{\ell}$, where $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$. Part (a) shows $\langle \mathbf{s}^{(1)} \rangle_{\ell}, \ldots, \langle \mathbf{s}^{(M)} \rangle_{\ell}$, while part (b) shows the pairwise authentication information

We extend APSS to another dimension. Let $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$ be three-dimensional vectors, where each individual vector consists of p elements from $\mathbb{F}$. Then a degree-ℓ APSS of $\mathbb{S}$ consists of ${}^{A}\langle \mathbf{S}^{(1)} \rangle_{\ell}, \ldots, {}^{A}\langle \mathbf{S}^{(L)} \rangle_{\ell}$. If ${}^{A}\langle \mathbb{S} \rangle_{\ell}$ and ${}^{A}\langle \mathbb{S}' \rangle_{\ell}$ are key-consistent, then an APSS of any publicly-known linear combination of the vectors in $\mathbb{S}$ and $\mathbb{S}'$ can be locally computed. Looking ahead, utilizing amortization, our VSS operates three-dimensional vectors of suitable dimensions L and M to attain constant overhead. We next review our VSS protocol.

2.2 Statistical VSS with a Constant Overhead

Let D be the designated dealer with input $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$, where each $\mathbf{S}^{(k)}$ further consists of Mp vectors each consisting of p elements from $\mathbb{F}$ where $p = \frac{t}{4} + 1$; hence $\mathbb{S}$ has LMp^2 elements from $\mathbb{F}$. Our VSS protocol then generates ${}^{A}\langle \mathbb{S} \rangle_{\ell} = ({}^{A}\langle \mathbf{S}^{(1)} \rangle_{\ell}, \ldots, {}^{A}\langle \mathbf{S}^{(1)} \rangle_{\ell})$ for any given $\ell \in \{p-1, d, d+p-1\}$ with $\mathcal{O}(1)$ communication overhead, provided $L = \Theta(n)$ and $M = \Theta(n^2)$ (hence $\mathbb{S}$ consists of $n^3p^2 = \Theta(n^5)$ field elements); here $d \stackrel{\text{def}}{=} t + p - 1$. Looking ahead, in our MPC protocol, we will encounter situations where a designaed dealer has to generate APSS of its inputs, with degree of sharing being either $p-1, d$ and $d+p-1$. For the ease of understanding, let us now assume $\ell = d$ and $\mathbb{S} = \mathbf{S}$ (i.e. $L = 1$) with $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(Mp)})$, where each vector $\mathbf{s}^{(k)} \in \mathbb{F}^p$. The protocol consists of a sequence of following 4 phases, each with a dedicated purpose.

The first phase is the *sharing phase*, where D distributes shares of its inputs and tries to publicly identify a set Core of $d+t+1 = 2t+p$ parties, who confirm the receipt of shares from D. A candidate Core contains at least $d+1$ *honest* parties, whose shares can define degree-d PSS. This phase *need not* be always successful, as there can be only $n - t = 2t + 1$ *honest* parties and the corrupt

parties may deny to be part of Core, even if D is *honest*. If D fails to identify a candidate Core, the parties restart the protocol, where the parties who failed to make it to Core previously are considered to be in "dispute" with D and included in a set Dp, such that D sets the shares of the parties in Dp to 0 when it restarts the protocol. An honest D is now bound to get a candidate Core, since the parties in Dp will be by default considered to be part of Core and $|\mathsf{Dp}| \geq t - p + 2$ with $p = \frac{t}{4} + 1$. Note that "fixing" the shares of the parties in Dp to 0 will not violate the privacy for an *honest* D since in this case Dp consist of only *corrupt* parties.

The next phase is the *verification phase* where if a candidate Core is identified, the parties verify if the shares of the parties in $\mathsf{Core} \cup \mathsf{Dp}$ are "consistent" and define degree-d PSS. If D is *corrupt* then Core may not have all honest parties. So once the "consistency" of the shares of the parties in $\mathsf{Core} \cup \mathsf{Dp}$ is confirmed, during the third phase, the parties in Core help the parties outside Core to recompute their shares, consistent with the shares of the parties in $\mathsf{Core} \cup \mathsf{Dp}$.

The last, but the most challenging phase, is the *pairwise tag-generation phase*, which consists of two steps. First, the parties try to compute the MAC tags. If a party fails to compute its supposed tags then it publicly complains for corrupt behaviour. All such complaints are then *publicly* resolved during the second step. Here either all the complaints are resolved successfully and tag-computations are completed or the parties publicly identify a set Cr of at least $t - p + 2$ *corrupt* parties and restart the protocol. During rerun, D sets the shares of the parties in Cr to 0. The protocol ensures that the parties in Cr cannot "disrupt" the tag-generation during the rerun. Moreover, every (honest) party will be now able to compute its supposed tags and no more reruns will be needed.

It follows that the parties may restart the protocol at most *twice*, once due to the failure of the first phase and once due to the failure of the last phase. We note that the asynchronous VSS protocol of [38] also has a similar structure as ours. However, in their protocol the parties *need not* have to restart the protocol since they *do not* generate APSS. Dealing with the restarts bring in additional technicalities. We now elaborate upon each individual phase, assuming $L = 1$.

Sharing phase. D on having the input $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(Mp)})$ where $\mathbf{s}^{(v)} = (s^{(v,1)}, \ldots, s^{(v,p)})$, picks M bivariate polynomials $F^{(1)}(X,Y), \ldots, F^{(M)}(X,Y)$ of degree d in both variables, where $d = t + p - 1$. Each $F^{(v)}(X,Y)$ is a random bivariate polynomial, except that the univariate polynomials $F^{(v)}(X,-1), \ldots, F^{(v)}(X,-p)$ constitute degree-d sharing-polynomials for $\langle \mathbf{s}^{((v-1)p+1)} \rangle_d, \ldots, \langle \mathbf{s}^{(vp)} \rangle_d$ respectively. D distributes the i^{th} *row polynomial* $F^{(v)}(X,i)$ and the i^{th} *column-polynomial* $F^{(v)}(i,Y)$ of each bivariate polynomial $F^{(v)}(X,Y)$ to each party P_i. The row and column-polynomial of the parties in $\mathsf{Dp} \cup \mathsf{Cr}$ are implicitly set to 0, if the parties have restarted the protocol due to the failure of sharing or tag-generation phase earlier. The privacy of $\mathbf{S}$ is maintained for an *honest* D, since for every $F^{(v)}(X,Y)$, the adversary learns at most t rows and columns, leaving $(t+p)^2 - t \cdot (t+p) - t \cdot p = p^2$ "degree of freedom". Hence adversary's view will be independent of the p^2 elements from $\mathbf{S}$ embedded in $F^{(v)}(X,Y)$. If D has distributed consistent polynomials, then every $P_i \notin \mathsf{Dp} \cup \mathsf{Cr}$ can compute its

shares corresponding to $\langle \mathbf{S} \rangle_d$ by evaluating its column-polynomials $F^{(v)}(i, Y)$ at $Y = -1, \ldots, -p$. Note that for the parties in $\mathsf{Dp} \cup \mathsf{Cr}$, the shares are set to 0.

In the remaining phases, to prove the consistency of the distributed shares, D will be asked to publicly open random linear combinations of its bivariate polynomials. To prevent it from using different polynomials than what it has used for distributing the rows and column-polynomials, once P_i receives its polynomials, it gives *information-checking signatures* (IC-signatures) on the points of the received column-polynomials to D. Informally, IC-signatures are information-theoretic analogue of digital signatures, which can be generated using an *information-checking protocol* (ICP) [43]. For efficiency, we present a new ICP which also enjoys the linearity property (see Section 4). D checks if it has received back signed column polynomials from a set of parties Core such that $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t+p$. If a Core exists then D announces the same. There will be at least $t+p = d+1$ honest parties in $\mathsf{Core} \cup \mathsf{Dp}$. So, the column polynomials of the *first* $t+p$ *honest* parties in $\mathsf{Core} \cup \mathsf{Dp}$ define valid degree-d bivariate polynomials, with the column polynomials of the parties in Dp being supposedly 0.

During the *first* run of the protocol, D *may not* find a candidate Core, as up to $t - p + 2$ *corrupt* parties many not send back the correct signed column polynomials. In this case, D gets into dispute with them and the parties restart the protocol with the set Dp. Note that Dp contains corrupt parties only for an honest D. If the parties restart, then next time an *honest* D is bound to get a candidate Core set. For a pictorial illustration of this phase, see Fig 2.

Verification phase. To verify if D has distributed consistent rows and column-polynomials with the polynomials of the parties in $\mathsf{Cr} \cup \mathsf{Dp}$ being 0, D is challenged to make public a random linear combination of its bivariate polynomials. To prevent D from making public linear combination of incorrect bivariate polynomials, D is also asked to publicly reveal the IC-signature of every party in Core on the column polynomials of the linear combination of the bivariate polynomials. The parties then verify the signatures and if the signed values lie on the polynomials made public by D. Additionally, it is also verified that if the row and column polynomials of the parties in Dp and Cr in the publicly-revealed polynomials are all 0. If all these checks pass then with a high probability, D has distributed consistent rows and column polynomials to the (honest) parties in $\mathsf{Core} \cup \mathsf{Dp}$. To maintain the privacy of $\mathbf{S}$ while revealing linear combination of the bivariate polynomials, we let D use additional random "masking data" independent of $\mathbf{S}$, which has the same number of vectors as $\mathbf{S}$ and distribute shares of those vectors during the sharing phase, as done for $\mathbf{S}$. Looking ahead, *apart* from this phase, D will be asked to reveal linear combination of its bivariate polynomials for $\mathbf{S}$ *thrice* in the remaining phases and so it is sufficient to pick four batches of random masking vectors.

Non-Core share reconstruction phase. If D is *corrupt*, then there might be honest parties outside $\mathsf{Core} \cup \mathsf{Dp}$, whose row and column polynomials are inconsistent with the bivariate polynomials "committed" by D to the parties in $\mathsf{Core} \cup \mathsf{Dp}$. To help non-$\mathsf{Core}$ parties recompute their supposed row and column polynomials, the parties in $\mathsf{Core} \cup \mathsf{Dp}$ provide the supposedly common points

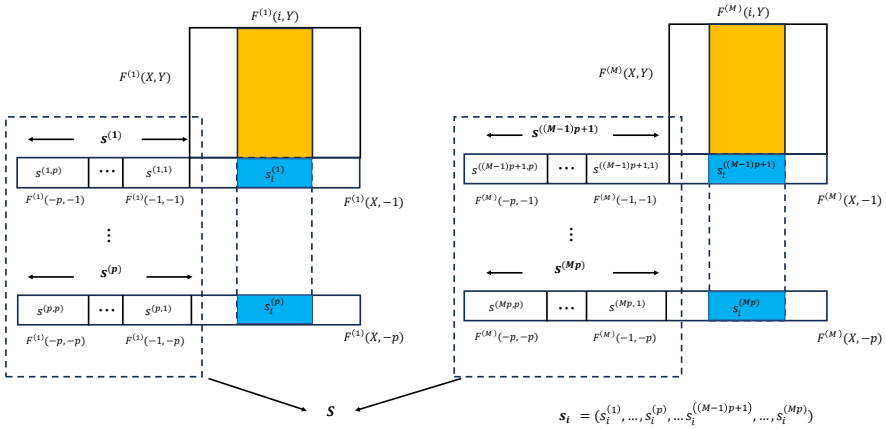


Fig. 2: Pictorial illustration of the sharing phase. D picks bivariate polynomials $F^{(1)}(X,Y),\ldots,F^{(M)}(X,Y)$, each of degree d, where each polynomial hides p vectors from $\mathbf{S}$. For instance, $F^{(1)}(X,Y)$ embeds $\mathbf{s}^{(1)},\ldots,\mathbf{s}^{(p)}$, in the polynomials $F^{(1)}(X,-1),\ldots,F^{(1)}(X,-p)$ respectively. The individual elements of the vectors are embedded at $X=-1,\ldots,-p$. Party P_i gets M column polynomials which are highlighted. Party P_i can compute its share vector $\mathbf{s}_i$ by evaluating its column polynomials at $Y=-1,\ldots,-p$. The elements of $\mathbf{s}_i$ are highlighted in blue color. Party P_i signs M column polynomials (namely distinct points on these polynomials as highlighted in yellow color) through an instance of ICP for D.

on their row and column polynomials to the outside parties (for the parties in $\mathsf{Dp}\cup\mathsf{Cr}$, the supposedly common points are implicitly taken as 0). Note that corrupt parties in Core may send incorrect common points. Moreover, we *cannot* use RS error-correction due to a high value of d. The way out is to ask D to help by asking it to publicly reveal a random linear combination of its committed bivariate polynomials, where we again use the IC-signature trick to prevent a corrupt D from revealing a linear combination of incorrect bivariate polynomials. Every party outside $\mathsf{Core}\cup\mathsf{Dp}\cup\mathsf{Cr}$ then applies the same linear combination on the supposedly common points that it received from the parties in Core and verify if they lie on the bivariate polynomials made public by D. Since corrupt parties in Core are *unaware* of the random linear combination used to verify their sent common points (at the time of sending), all incorrect common points are identified with a high probability. Once every party outside $\mathsf{Core}\cup\mathsf{Dp}\cup\mathsf{Cr}$ verifiably receives $d+1$ correct common points on their supposed row and column polynomials, it interpolates them to recompute its row and column polynomials. Note that at least $d+1$ correct common points are guaranteed to be available, since there are at least $d+1$ honest parties in $\mathsf{Core}\cup\mathsf{Dp}$, as $|\mathsf{Core}\cup\mathsf{Dp}|\geq d+t+1$.

Pairwise tag-generation phase. At the end of the previous phase, each P_i has its share-vector $\mathbf{s}_i=(g_i^{(1)}(-1),\ldots,g_i^{(1)}(-p),\ldots,\ldots g_i^{(M)}(-1),\ldots,g_i^{(M)}(-p))$ for

$\langle \mathbf{S} \rangle_d$, where $g_i^{(1)}(Y), \ldots, g_i^{(M)}(Y)$ are the column polynomials held by P_i. So the previous phase generates $\langle \mathbf{S} \rangle_d$ and this phase transforms $\langle \mathbf{S} \rangle_d$ into ${}^A\langle \mathbf{S} \rangle_d$, with a possibility of failing just once. In the latter case the parties identify a set Cr of at least $t - p + 2$ corrupt parties and restart the protocol in such a way that in the next run this phase is *guaranteed* to be successful. Let us assume that the parties are running this phase for the first time and $\mathsf{Cr} = \emptyset$.

Consider an arbitrary pair (P_i, P_j) and P_i wants to compute the tag $\tau_{ij} \stackrel{\mathsf{def}}{=} \mathbf{s}_i \cdot \boldsymbol{\mu}_{ji} + v_{ji}$, for the key $\mathsf{K}_{ji} \stackrel{\mathsf{def}}{=} (\boldsymbol{\mu}_{ji}, v_{ji})$ held by P_j. This could be done via a "mini-MPC" protocol, where P_i secret-shares $\mathbf{s}_i$ and P_j secret-shares v_{ji}, let the parties compute a secret-sharing of τ_{ij} and reconstruct it towards P_i.[7] However, as observed in [9,38], a degree-d *twisted-sharing* of $\mathbf{s}_i$ with respect to P_i is *already* available to the parties.[8] Namely the parties already hold $\lceil \mathbf{s}_i \rfloor_d^i$ where every party P_l except P_i *already* hold its respective twisted share corresponding to $\lceil \mathbf{s}_i \rfloor_d^i$, and the share of P_l is actually its own share-vector $\mathbf{s}_l$ of $\langle \mathbf{S} \rangle_d$. This is because the polynomials $((F^{(1)}(X,-1), \ldots, F^{(1)}(X,-p)), \ldots, (F^{(M)}(X,-1), \ldots, F^{(M)}(X,-p)))$ constitute degree-d polynomials whose i^{th} points are the elements of $\mathbf{s}_i$ and every party $P_l \neq P_i$ has the l^{th} points on these polynomials which are elements of the vector $\mathbf{s}_l$. Note that P_i *will not* have its twisted-shares corresponding to $\lceil \mathbf{s}_i \rfloor_d^i$, since they are the 0^{th} points on the above polynomials, which are *not* available with P_i. Based on this observation, the idea is to let the parties compute

$$\lceil \tau_{ij} \rfloor_{t+d}^i = \lceil \mathbf{s}_i \rfloor_d^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \lceil v_{ji} \rfloor_t^i,$$

where $\lceil v_{ji} \rfloor_t^i$ is a random twisted t-sharing of v_{ji} held by P_j. By setting the twisted-shares of P_i for $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ to $\mathbf{0}$, we make sure that the twisted-share of P_i for $\lceil \mathbf{s}_i \rfloor_d^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ is $\mathbf{0}$ too. So P_i can *also* compute its share corresponding to $\lceil \tau_{ij} \rfloor_{t+d}^i$. It now follows that all the parties can compute their respective shares of $\lceil \tau_{ij} \rfloor_{t+d}^i$ and send to P_i. As $[\tau_{ij}]_{t+d}^i$ is *not* a random degree-$(t+d)$ twisted-sharing of τ_{ij}, to avoid breach of privacy, we randomize the above sharing by masking it with a random degree-$(t+d)$ twisted sharing $\lceil 0 \rfloor_{t+d}^i$ of 0, before reconstructing it towards P_i. Following [27,38], a random $\lceil 0 \rfloor_{t+d}^i$ can be generated from a *publicly-known* linear combination of several random t-sharings.

Based on the above discussion, the parties can compute their shares of $[\tau_{ij}]_{t+d}^i$ and send to P_i. We need to provide a mechanism for P_i to verify the tag-shares that it received from every party P_l. The verification mechanism is to let the parties compute a twisted t-sharing with respect to P_i, whose l^{th} share is same as the supposed l^{th} tag-share of τ_{ij} sent by P_l to P_i and enable P_i to reconstruct the complete twisted t-sharing, who can then compare the tag-shares received from P_l with the l^{th} share of the reconstructed t-sharing. Such a twisted t-sharing can be computed as follows: consider the random sharing $\lceil 0 \rfloor_{t+d}^i$, for which P_l would

[7] As a one-time setup, a secret-sharing of $\boldsymbol{\mu}_{ji}$ will be available to the parties apriori.

[8] A vector $\mathbf{a}$ is said to be degree-ℓ twisted-shared with respect to P_i, denoted by $\lceil \mathbf{a} \rfloor_\ell^i$, if for every element $a \in \mathbf{a}$, there is a degree-ℓ polynomial, say $f_a(X)$, such that $f_a(i) = a$ and every $P_j \in \mathcal{P} \setminus \{P_i\}$ holds the share $f(j)$ and party P_i holds the share $f(0)$. In essence a twisted-sharing (with respect to P_i) is same as the regular degree-ℓ sharing, where the slots for embedding the secret and the share of P_i are "swapped".

have computed its share by taking a *publicly-known* linear combination of the random t-sharings which are used to compute $\lceil 0 \rfloor^i_{t+d}$. If all the parties apply the same linear combination (as P_l), then it will lead to a t-sharing whose l^{th} share will be the same as share of P_l for $\lceil 0 \rfloor^i_{t+d}$. We next find a t-sharing whose l^{th} share will be the same $\mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}$.

For a moment, let us forget the privacy of $\mathbf{S}$. Then D can make *public* the share $\mathbf{s}_l$ in a verifiable way (using the IC-signature trick). The parties can then compute the degree-0 constant twisted-sharing $\lceil \mathbf{s}_l \rfloor^i_0$ with respect to P_i. Then consider the degree-t twisted-sharing $\lceil \mathbf{s}_l \rfloor^i_0 \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor^i_t + \lceil v_{ji} \rfloor^i_t$. It turns out that l^{th} share of this t-sharing is exactly the same as $\mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}$. Hence we have a t-sharing whose l^{th} share is exactly the same as P_l's supposed tag-share $\tau_{ij,l}$.

To maintain the privacy of $\mathbf{s}$ (and hence $\mathbf{S}$), instead of applying the above idea only for $\mathbf{S}$, we apply it on a random linear combination of $\mathbf{S}$ and the masking-vectors used by D. Namely, we let D to verifiably reveal a random linear combination of its committed bivariate polynomials (as done in the earlier phases). Party P_i applies the same linear combination on all the tag-shares that P_i has received from P_l. The parties then compute a twisted t-sharing whose l^{th} share will be same as the supposed linear combination of tag-shares of P_l. This twisted t-sharing is then reconstructed towards P_i for verifying the tag-shares sent by P_l. The privacy of $\mathbf{S}$ is maintained due to masking vectors. A minor caveat is that directly reconstructing the final twisted t-sharing as above will reveal additional information about the honest parties' shares. The simple fix is to mask the final twisted t-sharing with a random twisted t-sharing whose l^{th} share is 0.

If P_i verifiably gets at least $t + d = 2t + p$ correct tag-shares then it can reconstruct τ_{ij}. Else, it complaints against those parties P_l whom it alleges for incorrect tag-shares. Such complaints are then *publicly* resolved by first letting P_l make public all its tag-shares for P_i. D then verifiably makes public a random linear combination of its committed bivariate polynomials and then parties computing a twisted t-sharing whose l^{th} share will be the corresponding linear combination of all the tag-shares which P_l has made public. This way, either a corrupt P_l is included to Cr or P_i gets correct tag-shares from P_l. Resolving the complaints publicly does not breach privacy, since either P_l or P_i is corrupt.

If every party gets $2t + p$ correct tag-shares for every tag that it is supposed to compute, then the protocol completes here. Otherwise parties restart with a set of corrupt parties Cr. Note that $|\mathsf{Cr}| \geq t - p + 2$. To ensure that the parties in Cr do not "disrupt" the tag-computation phase during rerun, we ensure that for every $P_l \in \mathsf{Cr}$, the tag-shares of P_l are implicitly "set" to 0 during all tag computations. For this, D fixes the row and column polynomials of each $P_l \in \mathsf{Cr}$ to $\mathbf{0}$ while selecting its bivariate polynomials (and the parties verify the same every time D reveals a random linear combination of its bivariate polynomials). This sets the share $\mathbf{s}_l$ of every $P_l \in \mathsf{Cr}$ to $\mathbf{0}$ and thus the twisted-shares of P_l for the twisted-sharings of every party's share is also $\mathbf{0}$. Furthermore, for each pair (P_i, P_j), while generating $\lceil v_{ji} \rfloor^i_t$ and the random t-sharings which are used to compute the random masking $\lceil 0 \rfloor^i_{t+d}$, we ensure that the shares of P_l are 0.

Looking ahead, the cost of the VSS is dominated by the instances of ICP involved and generating the various secret-shared masks during the tag-generation phase. To achieve $\mathcal{O}(1)$ overhead via amortization, D needs to execute the above protocol with n batches $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(n)}$ of secrets, where each $\mathbf{S}^{(u)}$ consists of $n^2 p$ vectors, each consisting of p elements from $\mathbb{F}$, resulting in number of secrets to be secret-shared to be at least n^5; see Section 5 for details.

2.3 Multiplication Over APSS with a Constant Overhead

We next overview our multiplication protocol. For simplicity, we consider one-dimensional APSS of the form ${}^{A}\langle \mathbf{s} \rangle_{\ell}$, but in our actual protocol we need to consider three-dimensional APSS of the form ${}^{A}\langle \mathbb{S} \rangle_{\ell}$ to get $\mathcal{O}(1)$ communication overhead via ammortization. We first review the existing multiplication protocol for PSS based on [23,44,27], which we call Π_{Beaver}. We will then see the challenges in extending the protocol to APSS and our final solution.

Existing Multiplication Protocol for PSS. Protocol Π_{Beaver} takes input PSS $(\langle \boldsymbol{x} \rangle_d, \langle \boldsymbol{y} \rangle_d)$, along with an auxiliary triple $(\langle \mathbf{a} \rangle_d, \langle \mathbf{b} \rangle_d, \langle \mathbf{c} \rangle_d)$ and an auxiliary pair $(\langle \mathbf{r} \rangle_d, \langle \mathbf{r} \rangle_{d+p-1})$, and outputs $\langle \boldsymbol{z} \rangle_d$ where $\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{y}$ iff $\mathbf{c} = \mathbf{a} \times \mathbf{b}$. Moreover, privacy of $\boldsymbol{x}$ and $\boldsymbol{y}$ is maintained if $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $\mathbf{r}$ remains random for the adversary. The protocol assumes the existence of a robust *degree-reduction* protocol Π_{DegRed}, which transforms $\langle \mathbf{s} \rangle_{\ell}$ into $\langle \mathbf{s} \rangle_{p-1}$, where $\ell \in \{d, d+p-1\}$. The steps of Π_{Beaver} are as follows.

1. Parties locally compute $\langle \boldsymbol{x} + \mathbf{a} \rangle_d$ and $\langle \boldsymbol{y} + \mathbf{b} \rangle_d$.
2. Parties do: $\langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \leftarrow \Pi_{\mathsf{DegRed}}(\langle \boldsymbol{x} + \mathbf{a} \rangle_d)$, $\langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} \leftarrow \Pi_{\mathsf{DegRed}}(\langle \boldsymbol{y} + \mathbf{b} \rangle_d)$.
3. Parties locally compute: $\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{d+p-1} = \langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \times \langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} - \langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \langle \mathbf{b} \rangle_d - \langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} \langle \mathbf{a} \rangle_d + \langle \mathbf{c} \rangle_d + \langle \mathbf{r} \rangle_{d+p-1}$.
4. Parties do $\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{p-1} \leftarrow \Pi_{\mathsf{DegRed}}(\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{d+p-1})$.
5. Parties then locally compute the output $\langle \boldsymbol{z} \rangle_d = \langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{p-1} - \langle \mathbf{r} \rangle_d$.

Challenges in Extending Π_{DegRed} from PSS to APSS. Since APSS satisfies the linearity property, the steps in Π_{Beaver} involving only local computation works over APSS as well. So let us review the the steps of Π_{DegRed} for PSS from [23], which transforms $\langle \mathbf{s}^{(1)} \rangle_{\ell}, \ldots, \langle \mathbf{s}^{(t+1)} \rangle_{\ell}$ to $\langle \mathbf{s}^{(1)} \rangle_{p-1}, \ldots, \langle \mathbf{s}^{(t+1)} \rangle_{p-1}$, for any $\ell \in \{d, d+p-1\}$. In the protocol, parties first locally "expand" the input sharings into n sharings using RS linear error-correcting code that can error-correct up to t errors. Namely, let $\mathbf{S}(X)$ be the vectors of degree-t polynomials where $\mathbf{S}(X) = \mathbf{s}^{(1)} + \mathbf{s}^{(1)} \cdot X + \ldots + \mathbf{s}^{(t)} \cdot X^t$; i.e. the coefficients of X^j in the polynomials in $\mathbf{S}(X)$ are the elements of $\mathbf{s}^{(j+1)}$. Then consider the vectors of points $\mathbf{S}(1), \ldots, \mathbf{S}(n)$ on $\mathbf{S}(X)$. It follows that the parties can locally compute $\langle \mathbf{S}(i) \rangle_{\ell}$ from $\langle \mathbf{s}^{(1)} \rangle_{\ell}, \ldots, \langle \mathbf{s}^{(t+1)} \rangle_{\ell}$. Each P_i is then designated to robutsly reconstruct $\mathbf{S}(i)$ from $\langle \mathbf{S}(i) \rangle_{\ell}$, followed by generating $\langle \mathbf{S}(i) \rangle_{p-1}$. Note that up to t corrupt parties may generate PSS of *incorrect vectors*. As observed in [23], the shares of every P_i for these n degree-reduced PSSs correspond to an n-length RS codeword with at

most t errors. So if $n = 3t+1$, each party can locally error-correct and obtain the shares for $\langle \mathbf{s}^{(1)} \rangle_{p-1}, \ldots, \langle \mathbf{s}^{(t+1)} \rangle_{p-1}$. From the above discussion, it follows that Π_{DegRed} will work over APSS, provided APSS supports the following features.

- Linearity: this is *true* due to linearity of MACs;
- Robust reconstruction of $\langle \cdot \rangle_\ell$-shared vectors: this is *true* since MACs help to identify incorrect shares;
- Provision to verifiably generate ${}^A\langle \cdot \rangle_\ell$ of inputs by designated parties; this is *possible* via our VSS protocol.

It follows that step 2 of Π_{Beaver} *will work* for APSS as well, since all the above-mentioned pre-requisites for Π_{DegRed} will be satisfied. Unfortunately, step 3 of Π_{Beaver} *does not* work for APSS, as parties *cannot* locally compute ${}^A\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{d+p-1}$. This is because for PSS the following hold:

$$\begin{aligned}\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{d+p-1} = \langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} - \langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \langle \mathbf{b} \rangle_d \\ - \langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} \langle \mathbf{a} \rangle_d + \langle \mathbf{c} \rangle_d + \langle \mathbf{r} \rangle_{d+p-1}.\end{aligned}$$

On the other hand, for APSS, the following hold, as MACs do not support non-linearity:

$$\begin{aligned}{}^A\langle \boldsymbol{x} \times \boldsymbol{y} + \mathbf{r} \rangle_{d+p-1} \neq {}^A\langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} {}^A\langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} - {}^A\langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1} \langle \mathbf{b} \rangle_d \\ - {}^A\langle \boldsymbol{y} + \mathbf{b} \rangle_{p-1} {}^A\langle \mathbf{a} \rangle_d + {}^A\langle \mathbf{c} \rangle_d + \langle \mathbf{r} \rangle_{d+p-1}.\end{aligned}$$

For instance, parties *cannot* locally compute the MAC-tags and MAC-keys corresponding to ${}^A\langle (\boldsymbol{x} + \mathbf{a}) \times \mathbf{b} \rangle_{d+p-1}$, given ${}^A\langle \boldsymbol{x} + \mathbf{a} \rangle_{p-1}$ and ${}^A\langle \mathbf{b} \rangle_d$. The absence of MAC-tags will further *prohibit* the parties to robustly reconstruct underlying degree-$(d+p-1)$-sharings, required as part of Π_{DegRed} in step 4 of Π_{Beaver}.

The Way out. We note that even though the parties *cannot* robustly reconstruct degree-$(d+p-1)$ PSS as a part of Π_{DegRed} during step 4, they can *detect* the presence of more than $\frac{t}{2}$ errors, since $d+p-1 = t+\frac{t}{2}, n = 3t+1$ and the minimum distance of underlying RS code will be $t + \frac{t}{2} + 1$. If any party complaints about the presence of more than $\frac{t}{2}$ errors while reconstructing its designated degree-$(d+p-1)$ PSS within Π_{DegRed} in step 4 of Π_{Beaver}, we can publicly open all the underlying ${}^A\langle \cdot \rangle_{p-1}$-sharings and all the underlying ${}^A\langle \cdot \rangle_d$-sharings, which are involved in the computation of the underlying degree-$(d+p-1)$ PSSs in step 3. From this, the complaining party can locally identify more than $\frac{t}{2}$ corrupt parties and discard their shares, after which it can robustly reconstruct its designated degree-$(d+p-1)$ PSSs and complete the remaining steps of Π_{DegRed} in step 4. A similar idea is used in [45], which requires a restart after fault identification. We *do not* require any such restart. The details follow.

Multiplication Protocol for APSS. The actual multiplication protocol takes input pairs $\{({}^A\langle \boldsymbol{x}^{(i)} \rangle_d, {}^A\langle \boldsymbol{y}^{(i)} \rangle_d)\}$, auxiliary triples $\{({}^A\langle \mathbf{a}^{(i)} \rangle_d, {}^A\langle \mathbf{b}^{(i)} \rangle_d, {}^A\langle \mathbf{c}^{(i)} \rangle_d)\}$ and auxiliary pairs $\{({}^A\langle \mathbf{r}^{(i)} \rangle_d, {}^A\langle \mathbf{r}^{(i)} \rangle_{d+p-1})\}$. To deal with the problem discussed

earlier, the parties hold *additional* auxiliary triples $\{({}^{A}\langle\hat{\mathbf{a}}^{(i)}\rangle_d, {}^{A}\langle\hat{\mathbf{b}}^{(i)}\rangle_d, {}^{A}\langle\hat{\mathbf{c}}^{(i)}\rangle_d)\}$ and auxiliary pairs $\{({}^{A}\langle\hat{\mathbf{r}}^{(i)}\rangle_d, {}^{A}\langle\hat{\mathbf{r}}^{(i)}\rangle_{d+p-1})\}$. We refer to this additional auxiliary information as "sacrificing randomness" as their privacy will be lost to identify the corrupt parties. The protocol outputs $\{{}^{A}\langle\boldsymbol{z}^{(i)}\rangle_d\}$, where $\boldsymbol{z}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{y}^{(i)}$, iff $\mathbf{c}^{(i)} = \mathbf{a}^{(i)} \times \mathbf{b}^{(i)}$. Here $i \in \{1, \ldots, t+1\}$. The reason for computing $t+1$ products is that it will lead to $\mathcal{O}(1)$ communication overhead via ammortization.

The parties will follow the first two steps of Π_{Beaver} outlined earlier. During the third step, the parties locally compute the following PSS (and not APSS) $\langle\mathbf{s}^{(i)}\rangle_{d+p-1} = \langle\boldsymbol{x}^{(i)} + \mathbf{a}^{(i)}\rangle_{p-1}\langle\boldsymbol{y}^{(i)} + \mathbf{b}^{(i)}\rangle_{p-1} - \langle\boldsymbol{x}^{(i)} + \mathbf{a}^{(i)}\rangle_{p-1}\langle\mathbf{b}^{(i)}\rangle_d - \langle\boldsymbol{y}^{(i)} + \mathbf{b}^{(i)}\rangle_{p-1}\langle\mathbf{a}^{(i)}\rangle_d + \langle\mathbf{c}^{(i)}\rangle_d + \langle\mathbf{r}^{(i)}\rangle_{d+p-1}$, where $\mathbf{s}^{(i)} = \boldsymbol{x}^{(i)} \times \boldsymbol{y}^{(i)} + \mathbf{r}^{(i)}$. The parties then "optimistically" proceed with the steps of Π_{DegRed} to transform $\langle\mathbf{s}^{(i)}\rangle_{d+p-1}$ into $\langle\mathbf{s}^{(i)}\rangle_{p-1}$: let $\mathbf{S}(X)$ be the vectors of degree-t polynomials where $\mathbf{S}(X) = \mathbf{s}^{(1)} + \mathbf{s}^{(2)} \cdot X + \ldots + \mathbf{s}^{(t+1)} \cdot X^t$. For $i \in \{1, \ldots, n\}$, the parties locally compute $\langle\mathbf{S}(i)\rangle_{d+p-1}$ and send their shares of $\langle\mathbf{S}(i)\rangle_{d+p-1}$ to P_i for reconstruction. Party P_i error-correct $\frac{t}{2}$ errors, assuming at most $\frac{t}{2}$ incorrect shares. If P_i fails to reconstruct any $\mathbf{S}(i)$ through error-correction, then it knows that more than $\frac{t}{2}$ corrupt parties have sent incorrect shares and so it complains publicly. On the other hand, if error-correction outputs any vector for P_i then it is $\mathbf{S}(i)$. This is because the minimum distance of the underlying RS code (defined by degree-$(d+p-1)$ polynomials) is $t + \frac{t}{2} + 1$ if $p = \frac{t}{4} + 1$ and $n = 3t + 1$. So even if adversary introduces t errors in the shares of $\langle\mathbf{S}(i)\rangle_{d+p-1}$, the received vector of shares will be at a distance of at least $\frac{t}{2} + 1$ from the RS codeword of any other $\mathbf{S}'(i)$ where $\mathbf{S}'(i) \neq \mathbf{S}(i)$. If no party complains then the parties can complete the remaining steps of Π_{DegRed}, followed by step 5 of Π_{Beaver}. Otherwise, the parties resolve the complaints by deploying the sacrificing randomness as follows.

For each $i \in \{1, \ldots, t+1\}$, the parties first locally compute the PSS $\langle\hat{\mathbf{s}}^{(i)}\rangle_{d+p-1} = \langle\boldsymbol{x}^{(i)} + \mathbf{a}^{(i)}\rangle_{p-1}\langle\boldsymbol{y}^{(i)} + \mathbf{b}^{(i)}\rangle_{p-1} - \langle\boldsymbol{x}^{(i)} + \mathbf{a}^{(i)}\rangle_{p-1}\langle\hat{\mathbf{b}}^{(i)}\rangle_d - \langle\boldsymbol{y}^{(i)} + \mathbf{b}^{(i)}\rangle_{p-1}\langle\hat{\mathbf{a}}^{(i)}\rangle_d + \langle\hat{\mathbf{c}}^{(i)}\rangle_d + \langle\hat{\mathbf{r}}^{(i)}\rangle_{d+p-1}$. Let $\hat{\mathbf{S}}(X)$ be the vectors of degree-t polynomials where $\hat{\mathbf{S}}(X) = \hat{\mathbf{s}}^{(1)} + \hat{\mathbf{s}}^{(2)} \cdot X + \ldots + \hat{\mathbf{s}}^{(t+1)} \cdot X^t$. For $i \in \{1, \ldots, n\}$, the parties locally compute $\langle\hat{\mathbf{S}}(i)\rangle_{d+p-1}$ and send their shares of $\langle\hat{\mathbf{S}}(i)\rangle_{d+p-1}$ to P_i for reconstructing $\hat{\mathbf{S}}(i)$. If more than $\frac{t}{2}$ corrupt parties send incorrect shares then P_i *cannot* reconstruct $\hat{\mathbf{S}}(i)$. The parties next collectively generate a random value, say R. For each $j \in \{1, \ldots, t+1\}$, consider the vector $\mathbf{s}^{(j)} + R \cdot \hat{\mathbf{s}}^{(j)}$, which is same as

$$(R+1) \cdot (\boldsymbol{x}^{(j)} + \mathbf{a}^{(j)}) \times (\boldsymbol{y}^{(j)} + \mathbf{b}^{(j)}) - (\boldsymbol{x}^{(j)} + \mathbf{a}^{(j)}) \times (\mathbf{b}^{(j)} + R \cdot \hat{\mathbf{b}}^{(j)}) - (\boldsymbol{y}^{(j)} + \mathbf{b}^{(j)}) \times (\mathbf{a}^{(j)} + R \cdot \hat{\mathbf{a}}^{(j)}) + (\mathbf{c}^{(j)} + R \cdot \hat{\mathbf{c}}^{(j)}) + (\mathbf{r}^{(j)} + R \cdot \hat{\mathbf{r}}^{(j)}).$$

It turns out that $\mathbf{S}(i) + R \cdot \hat{\mathbf{S}}(i)$ is a linear combination of $\{\mathbf{s}^{(j)} + R \cdot \hat{\mathbf{s}}^{(j)}\}_{j \in \{1,\ldots,t+1\}}$. So if we enable every complaining P_i compute $\{\mathbf{s}^{(j)} + R \cdot \hat{\mathbf{s}}^{(j)}\}_{j \in \{1,\ldots,t+1\}}$, then P_i can recompute $\langle\mathbf{S}(i) + R \cdot \hat{\mathbf{S}}(i)\rangle_{d+p-1}$ and find out the identity of more than $\frac{t}{2}$ corrupt parties who sent incorrect shares for $\langle\mathbf{S}(i)\rangle_{d+p-1}$ (and also possibly for $\langle\hat{\mathbf{S}}(i)\rangle_{d+p-1}$). By ignoring these shares, P_i can robustly reconstruct $\mathbf{S}(i)$ and complete the rest of the steps of Π_{DegRed}. The idea here is that the corrupt parties while sending incorrect shares for $\langle\mathbf{S}(i)\rangle_{d+p-1}$ (and $\langle\hat{\mathbf{S}}(i)\rangle_{d+p-1}$) to P_i

will not know R apriori. If P_i can compute $\langle \mathbf{S}(i) + R \cdot \hat{\mathbf{S}}(i) \rangle_{d+p-1}$, then with a high probability, P_i can identify the incorrect shares.

To enable P_i compute $\{\mathbf{s}^{(j)} + R \cdot \hat{\mathbf{s}}^{(j)}\}_{j \in \{1,\ldots,t+1\}}$, we let P_i robustly reconstruct $\{(\boldsymbol{x}^{(j)} + \mathbf{a}^{(j)}), (\boldsymbol{y}^{(j)} + \mathbf{b}^{(j)}), (\mathbf{b}^{(j)} + R \cdot \hat{\mathbf{b}}^{(j)}), (\mathbf{a}^{(j)} + R \cdot \hat{\mathbf{a}}^{(j)}), (\mathbf{c}^{(j)} + R \cdot \hat{\mathbf{c}}^{(j)}), (\mathbf{r}^{(j)} + R \cdot \hat{\mathbf{r}}^{(j)})\}_{j \in \{1,\ldots,t+1\}}$. This is possible since the parties can locally compute APSS ($^A\langle \cdot \rangle_d$-sharing) of these vectors (as R will be known publicly). Note that the privacy of multiplicands $\boldsymbol{x}^{(j)}$ and $\boldsymbol{y}^{(j)}$ is maintained, as they are masked with $\mathbf{a}^{(j)}$ and $\mathbf{b}^{(j)}$. Moreover, the privacy of $\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)}$ and $\mathbf{r}^{(j)}$ is preserved, as they are masked by $\hat{\mathbf{a}}^{(j)}, \hat{\mathbf{b}}^{(j)}, \hat{\mathbf{c}}^{(j)}$ and $\hat{\mathbf{r}}^{(j)}$ respectively.

A small issue above is that we may need to reconstruct $\Theta(n)$ $^A\langle \cdot \rangle_d$-sharings for $\Theta(n)$ complaining parties. To maintain $\mathcal{O}(1)$ communication overhead, we perform multiplication for "multiple" batches in parallel. If any complaint arises, then we resolve the complaints of *only one* batch, through which the complainants identify more than $\frac{t}{2}$ corrupt parties; see Section 6 for details.

3 Preliminaries

We assume a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ with $n = 3t+1$, connected by pairwise *synchronous* private and authentic channels, where t is the maximum number of parties, which can be under the control of a computationally-unbounded malicious (Byzantine) adversary $\mathcal{A}$. For simplicity, we assume that the parties also have access to a system-wide broadcast channel, which allow any designated party to send any message identically to all the parties; one can emulate broadcast with $n = 3t + 1$ using constant expected round protocols (more on this later). The communication complexity of any protocol is defined to be the total number of bits sent by the *honest* parties, namely the parties which are not under adversary's control, both over point-to-point as well as broadcast channels. Specifically, we denote the communication complexity of a protocol by $\mathsf{P2P}(M) + N \times \mathsf{BC}(L)$, which means that honest parties send $\mathcal{O}(M)$ bits in total over the point-to-point channels, and calls the broadcast channel N times with a message of $\mathcal{O}(L)$ bits. We set the packing parameter p to $\frac{t}{4}+1$ and $d = t+p-1$.

Let κ be the statistical security parameter. We consider functions that can be represented as arithmetic circuits over a finite field $\mathbb{F}$ where $|\mathbb{F}| \geq 2^\kappa$, consisting of input, addition, multiplication and output gates. For any circuit $\mathtt{ckt}$ satisfying the above conditions, we have $|\mathbb{F}| \geq |\mathtt{ckt}| + n$. This is because $|\mathbb{F}| \geq 2^\kappa$ and both $|\mathtt{ckt}|$ and n are polynomial in κ. This also implies that $|\mathbb{F}| \geq 2n$. Without loss of generality we assume that $1, \ldots, n \in \mathbb{F}$ (and so $-1, \ldots, -n \in \mathbb{F}$ as well). Each element of $\mathbb{F}$ can be represented by $\mathcal{O}(\kappa)$ bits

We will be operating on vectors of values over $\mathbb{F}$. Variables like $\mathbf{a}, \mathbf{b}, \mathbf{c}$ will denote one-dimensional vectors. Notations like $\mathbf{A}, \mathbf{B}, \mathbf{C}$ will denote two-dimensional vectors (matrices). Variables like $\mathbb{A}, \mathbb{B}, \mathbb{C}$ will denote three-dimensional vectors. We will be using the following standard property of Reed-Solomon (RS) codes.

Theorem 3.1 ([41]). *Let $C \in \mathbb{F}^N$ be an RS code word, corresponding to a k-degree polynomial (containing $k+1$ coefficients) such that the distance of the*

code is $d = N - k$. Let $\bar{C} \in \mathbb{F}^N$, such that the distance between C and $\bar{C}$ is at most T. Let e be a parameter such that $e \leq T$ and $d > T + e$. If we apply the RS decoding algorithm to $\bar{C}$ to error-correct e errors, then the following hold: (a) If the decoding algorithm outputs a codeword C' then $C' = C$; (b) Else the decoding algorithm detects the presence of more than e errors in $\bar{C}$.

3.1 Information-Theoretic Message Authentication Codes (MAC)

We will use information-theoretically MAC. Here $\mathsf{K} = (\boldsymbol{\mu}, v)$ is selected as the MAC-key, where $\boldsymbol{\mu} \in \mathbb{F}^L$ and $v \in \mathbb{F}$. The MAC-tag on a vector of values $\mathbf{s} \in \mathbb{F}^L$ is defined as $\mathsf{MAC}_{\mathsf{K}}(\mathbf{s}) = \tau \stackrel{\mathsf{def}}{=} \boldsymbol{\mu} \cdot \mathbf{s} + v$, where $\boldsymbol{\mu} \cdot \mathbf{s}$ denotes the vector dot-product of $\boldsymbol{\mu}$ and $\mathbf{s}$. The MACs will be used as follows: a party P_i will hold some $\mathbf{s} \in \mathbb{F}^L$ and a MAC tag τ, while party P_j will hold the key K. Later, when P_i wants to reveal $\mathbf{s}$ to P_j, it sends $\mathbf{s}$, along with τ. Party P_j then verifies the revealed data and the tag, with respect to the key K held by P_j. If each component of K is selected randomly, then the probability that a *corrupt* P_i can find a *different* $(\mathbf{s}', \tau')$, such that $\tau' = \boldsymbol{\mu} \cdot \mathbf{s}' + v$ holds is at most $2^{-\Omega(\kappa)}$.

We call keys $\mathsf{K} = (\boldsymbol{\mu}, v)$ and $\mathsf{K}' = (\boldsymbol{\mu}', v')$ as *consistent* if $\boldsymbol{\mu} = \boldsymbol{\mu}'$ holds. Given consistent keys $\mathsf{K} = (\boldsymbol{\mu}, v)$ and $\mathsf{K}' = (\boldsymbol{\mu}, v')$ and a public constant $c \in \mathbb{F}$, we define the following operations:

$$\mathsf{K} + \mathsf{K}' \stackrel{\mathsf{def}}{=} (\boldsymbol{\mu}, v + v'), \quad \mathsf{K} + c \stackrel{\mathsf{def}}{=} (\boldsymbol{\mu}, v + \boldsymbol{\mu} \cdot \mathbf{c}) \quad \text{and} \quad c \cdot \mathsf{K} \stackrel{\mathsf{def}}{=} (\boldsymbol{\mu}, cv),$$

where $\mathbf{c} \stackrel{\mathsf{def}}{=} (c, \ldots, c \ (L \text{ times}))$. Given consistent keys K, K' and a public constant $c \in \mathbb{F}$, the following linearity property holds for any $\mathbf{s}, \mathbf{s}' \in \mathbb{F}^L$.

- **Addition**: $\mathsf{MAC}_{\mathsf{K}}(\mathbf{s}) + \mathsf{MAC}_{\mathsf{K}'}(\mathbf{s}') = \mathsf{MAC}_{\mathsf{K}+\mathsf{K}'}(\mathbf{s} + \mathbf{s}')$.
- **Addition/Subtraction by a Constant**: Let $\mathbf{c} \stackrel{\mathsf{def}}{=} (c, \ldots, c \ (L \text{ times}))$. Then: $\mathsf{MAC}_{\mathsf{K}-c}(\mathbf{s} + \mathbf{c}) = \mathsf{MAC}_{\mathsf{K}}(\mathbf{s})$ and $\mathsf{MAC}_{\mathsf{K}+c}(\mathbf{s} - \mathbf{c}) = \mathsf{MAC}_{\mathsf{K}}(\mathbf{s})$.
- **Multiplication by a Constant**: $c \cdot \mathsf{MAC}_{\mathsf{K}}(\mathbf{s}) = \mathsf{MAC}_{c \cdot \mathsf{K}}(c \cdot \mathbf{s})$.

3.2 Various Sharing Semantics and their Properties

We discuss the various secret-sharings used in our protocols and their properties.

Definition 3.2. *Let $s \in \mathbb{F}$ and $\mathbf{s} = (s^{(1)}, \ldots, s^{(M)}) \in \mathbb{F}^M$.*

- **ℓ-sharing**: *s is ℓ-shared, if there exists a degree-ℓ polynomial $f(\cdot)$ over $\mathbb{F}$, with $f(0) = s$ and each (honest) P_i has a share $s_i \stackrel{\mathsf{def}}{=} f(i)$ of s. A vector of shares of a ℓ-sharing of s is denoted by $[s]_\ell$.*
- *$\mathbf{s}$ is ℓ-shared, provided each $s^{(m)} \in \mathbf{s}$ is ℓ-shared. $[\mathbf{s}]_\ell \stackrel{\mathsf{def}}{=} \left[s^{(1)}\right]_\ell, \ldots, \left[s^{(M)}\right]_\ell$.*
- **Twisted ℓ-sharing with respect to P_i**: *Denoted by $\lceil s \rfloor_\ell^i$, this is same as $[s]_\ell$ except that $f(i) = s$ and every (honest) party $P_j \in \mathcal{P} \setminus \{P_i\}$ holds its twisted-share $s_j \stackrel{\mathsf{def}}{=} f(j)$. The twisted-share s_i of P_i is defined to be $s_i \stackrel{\mathsf{def}}{=} f(0)$.*
- *$\mathbf{s}$ is twisted ℓ-shared with respect to P_i, denoted by $\lceil \mathbf{s} \rfloor_\ell^i$, provided each $s^{(m)} \in \mathbf{s}$ is twisted ℓ-shared with respect to P_i. That is, $\lceil \mathbf{s} \rfloor_\ell^i \stackrel{\mathsf{def}}{=} \lceil s^{(1)} \rfloor_\ell^i, \ldots, \lceil s^{(M)} \rfloor_\ell^i$.*

– **ℓ-sharing and twisted ℓ-sharing with Different Slots for Secrets**: *$[s||i]_\ell$ for any $i \in \{1, p+1\}$ is same as $[s]_\ell$, except that $f(-i) = s$ holds. $\lceil s||i \rfloor_\ell^i$ is same as $\lceil s \rfloor_\ell^i$, except that s is stored at the evaluation point $-i$.*

It follows that both $[\cdot]_\ell$-sharing as well as $\lceil \cdot \rfloor_\ell$-sharing satisfies the linearity property. That is, given $[a]_{\ell_1}$ and $[b]_{\ell_2}$, where $\ell_1 \geq \ell_2$ and public constants $c_1, c_2 \in \mathbb{F}$, then each P_i can locally compute its share for $[c_1 a + c_2 b]_{\ell_1}$ as $c_1 \cdot [a]_{\ell_1} + c_2 \cdot [b]_{\ell_2} = [c_1 a + c_2 b]_{\ell_1}$. Similarly, given $[\mathbf{a}]_{\ell_1}$ and $[\mathbf{b}]_{\ell_2}$, where $\mathbf{a}, \mathbf{b} \in \mathbb{F}^M$, the condition $c_1 \cdot [\mathbf{a}]_{\ell_1} + c_2 \cdot [\mathbf{b}]_{\ell_2} = [c_1 \cdot \mathbf{a} + c_2 \cdot \mathbf{b}]_{\ell_1}$ holds. On the other hand, the conditions $[a]_{\ell_1} \cdot [b]_{\ell_2} = [ab]_{\ell_1+\ell_2}$ and $[\mathbf{a}]_{\ell_1} \cdot [\mathbf{b}]_{\ell_2} = [\mathbf{a} \cdot \mathbf{b}]_{\ell_1+\ell_2}$ hold.

Definition 3.3 (**Packed Secret Sharing (PSS)** [29]). *A vector $\mathbf{s} = (s^{(1)}, \ldots, s^{(k)}) \in \mathbb{F}^k$ is degree-ℓ packed-secret-shared where $\ell \geq k-1$, if there exists a degree-ℓ polynomial $f(\cdot)$ over $\mathbb{F}$, where $f(-1) = s^{(1)}, \ldots, f(-k) = s^{(k)}$ and each (honest) party $P_i \in \mathcal{P}$ has its share $s_i \stackrel{\text{def}}{=} f(i)$. A vector of shares for a degree-ℓ packed secret-sharing of $\mathbf{s}$ is denoted by $\langle \mathbf{s} \rangle_\ell$.*

Let $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)})$, where $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)} \in \mathbb{F}^k$. $\langle \mathbf{S} \rangle_\ell \stackrel{\text{def}}{=} (\langle \mathbf{s}^{(1)} \rangle_\ell, \ldots, \langle \mathbf{s}^{(m)} \rangle_\ell)$.

Let $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(\ell)})$ where each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}), \ldots, \mathbf{s}^{(u,m)})$ and where each $\mathbf{s}^{(u,v)} \in \mathbb{F}^k$. Then $\langle \mathbb{S} \rangle_\ell \stackrel{\text{def}}{=} (\langle \mathbf{S}^{(1)} \rangle_\ell, \ldots, \langle \mathbf{S}^{(\ell)} \rangle_\ell)$.

Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^k$. It is easy to see that given $\langle \mathbf{a} \rangle_{\ell_1}$ and $\langle \mathbf{b} \rangle_{\ell_2}$ where $\ell_1 \geq \ell_2$ and $\ell_1 + \ell_2 \leq n$ and public constants $c_1, c_2 \in \mathbb{F}$, then $c_1 \cdot \langle \mathbf{a} \rangle_{\ell_1} + c_2 \cdot \langle \mathbf{b} \rangle_{\ell_2} = \langle c_1 \cdot \mathbf{a} + c_2 \cdot \mathbf{b} \rangle_{\ell_1}$. On the other hand, $\langle \mathbf{a} \rangle_{\ell_1} \times \langle \mathbf{b} \rangle_{\ell_2} = \langle \mathbf{a} \times \mathbf{b} \rangle_{\ell_1+\ell_2}$. The above properties carry over for $\langle \mathbf{S} \rangle_\ell$ as well as $\langle \mathbb{S} \rangle_\ell$.

Definition 3.4 (**Authenticated Packed Secret Sharing (APSS)**). *A vector of vectors $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)})$, where $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)} \in \mathbb{F}^k$ is degree-ℓ packed-secret-shared with authentication where $\ell \geq k-1$, if all the following hold.*

– *Each $P_i \in \mathcal{P}$ has its share-vector $\mathbf{s}_i \in \mathbb{F}^m$ corresponding to $\langle \mathbf{S} \rangle_\ell$*
– *For every pair (P_i, P_j), there exists a MAC key $\mathsf{K}_{ji} = (\boldsymbol{\mu}_{ji}, v_{ji}) \in \mathbb{F}^{m+1}$ held by P_j, such that P_i holds the MAC tag $\tau_{ij} \stackrel{\text{def}}{=} \mathsf{MAC}_{\mathsf{K}_{ji}}(\mathbf{s}_i)$.*

We denote by ${}^A\langle \mathbf{S} \rangle_\ell$ the vector of shares, MAC keys and the MAC tags of $\mathbf{S}$ corresponding to all the (honest) parties in $\mathcal{P}$. That is: ${}^A\langle \mathbf{S} \rangle_\ell \stackrel{\text{def}}{=} \left\{ \mathbf{s}_i, (\tau_{i1}, \ldots, \tau_{in}), (\mathsf{K}_{i1}, \ldots, \mathsf{K}_{in}) \right\}_{i=1}^n$. Let $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$ where each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}), \ldots, \mathbf{s}^{(u,m)})$ and where each $\mathbf{s}^{(u,v)} \in \mathbb{F}^k$. Then ${}^A\langle \mathbb{S} \rangle_\ell \stackrel{\text{def}}{=} {}^A\langle \mathbf{S}^{(1)} \rangle_\ell, \ldots, {}^A\langle \mathbf{S}^{(L)} \rangle_\ell$.

${}^A\langle \mathbf{S} \rangle_{\ell_1}$ and ${}^A\langle \mathbf{S}' \rangle_{\ell_2}$, where $|\mathbf{S}| = |\mathbf{S}'|$ are said to be key-consistent if every $P_i \in \mathcal{P}$ holds consistent MAC keys for every $P_j \in \mathcal{P}$ in ${}^A\langle \mathbf{S} \rangle_{\ell_1}$ and ${}^A\langle \mathbf{S}' \rangle_{\ell_2}$.

${}^A\langle \mathbb{S} \rangle_{\ell_1}$ and ${}^A\langle \mathbb{S}' \rangle_{\ell_2}$, where $|\mathbb{S}| = |\mathbb{S}'|$ are said to be key-consistent if every $P_i \in \mathcal{P}$ holds consistent MAC keys for every $P_j \in \mathcal{P}$ in ${}^A\langle \mathbb{S} \rangle_{\ell_1}$ and ${}^A\langle \mathbb{S}' \rangle_{\ell_2}$.

Let $\mathbf{A} = (\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(m)})$ and $\mathbf{B} = (\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(m)})$, where each $\mathbf{a}^{(j)}, \mathbf{b}^{(j)} \in \mathbb{F}^k$. Given public constants $c_1, c_2 \in \mathbb{F}$, it follows from the linearity of MACs that $c_1 \cdot {}^A\langle \mathbf{A} \rangle_{\ell_1} + c_2 \cdot {}^A\langle \mathbf{B} \rangle_{\ell_2} = {}^A\langle \mathbf{C} \rangle_{\ell_1}$ holds, where $\ell_1 \geq \ell_2$. Here $\mathbf{C} = c_1 \cdot \mathbf{A} + c_2 \cdot \mathbf{B} = (\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)})$, where each $\mathbf{c}^{(j)} = c_1 \cdot \mathbf{a}^{(j)} + c_2 \cdot \mathbf{b}^{(j)}$.

On the other hand, if $\mathbf{C} = (\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}) = \mathbf{A} \times \mathbf{B}$, where each $\mathbf{c}^{(j)} = \mathbf{a}^{(j)} \times \mathbf{b}^{(j)}$, then ${}^{A}\langle \mathbf{A} \rangle_{\ell_1} \times {}^{A}\langle \mathbf{B} \rangle_{\ell_2} \neq {}^{A}\langle \mathbf{C} \rangle_{\ell_1+\ell_2}$, as parties *cannot* locally compute their tags and MAC-keys corresponding to ${}^{A}\langle \mathbf{C} \rangle_{\ell_1+\ell_2}$, since MACs *do not* support any multiplicative property. The above discussion extends for ${}^{A}\langle \mathbb{A} \rangle_{\ell}$ and ${}^{A}\langle \mathbb{B} \rangle_{\ell}$.

Definition 3.5 (Double APSS). *Let* $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)})$, *where* $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)} \in \mathbb{F}^p$. *Here* p *is the packing parameter and* $d = t + p - 1$. *Then* $\mathbf{S}$ *is double-packed-secret-shared with authentication, if the parties hold* ${}^{A}\langle \mathbf{S} \rangle_{d}$ *and* ${}^{A}\langle \mathbf{S} \rangle_{d+p-1}$.

$\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$ *is double-packed-secret-shared with authentication, if the parties hold* ${}^{A}\langle \mathbf{S}^{(u)} \rangle_{d}$ *and* ${}^{A}\langle \mathbf{S}^{(u)} \rangle_{d+p-1}$, *for each* $u \in \{1, \ldots, L\}$.

Definition 3.6 (degree-ℓ Degenerate Packed Twisted Sharing [38]). *Let* $\mathbf{s} = (s^{(0)}, \ldots, s^{(\ell)}) \in \mathbb{F}^{\ell+1}$. *A degree-*$\ell$ *degenerate packed secret-sharing of* $\mathbf{s}$ *with respect to* P_i, *denoted by* $[[\mathbf{s}]]^i_\ell$, *consists of shares* $s_1, \ldots, s_n \in \mathbb{F}$, *such that (note that the vector* $[[\mathbf{s}]]^i_\ell$ *is completely determined by* $\mathbf{s}$*):*

- *There exists a degree-*ℓ *polynomial* $f(X)$ *such that* $f(i) = s^{(0)}$ *and* $f(-1) = s^{(1)}, \ldots, f(-(i-1)) = s^{(i-1)}, f(-(i+1)) = s^{(i+1)}, \ldots, f(-\ell)) = s^{(\ell)}$.
- *Every party* $P_j \in \mathcal{P} \setminus \{P_i\}$ *has the share* $s_j = f(j)$.
- *Party* P_i *has the share* $s_i = f(0)$.

3.3 Reconstruction Protocols

We have the following reconstruction protocols for $n = 3t + 1$.

– **Reconstructing $[\cdot]_t$-shared and $\lceil \cdot \rfloor^i_t$-shared values**: This is done by the standard one round protocol where the parties exchange shares and applies RS error-correction to error-correct up to t errors on a t-degree polynomial.

– **Reconstructing $\langle \cdot \rangle_\ell$-shared values**: Let $\mathbf{s} = (s^{(1)}, \ldots, s^{(k)})$. Given $\langle \mathbf{s} \rangle_\ell$ where $\ell \geq k - 1$ and $\ell \leq t$, any $P_R \in \mathcal{P}$ can reconstruct $\mathbf{s}$ by letting each party send its share to P_R, who error-corrects incorrect shares through RS error-correction. The protocol has an overhead of $\mathcal{O}(1)$ bits, since $|\mathbf{s}| = \Theta(n\kappa)$ bits. The procedure can be extended even for $\langle \mathbb{S} \rangle_\ell$, provided each vector in $\mathbb{S}$ consists of k elements where k and ℓ satisfies the above constraints.

– **Reconstructing ${}^{A}\langle \cdot \rangle_\ell$-shared values**: Let $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)})$ where each $\mathbf{s}^{(v)} \in \mathbb{F}^k$. Given ${}^{A}\langle \mathbf{S} \rangle_\ell$ where $n - t > \ell$, any P_j can reconstruct $\mathbf{S}$ and also recompute $\langle \mathbf{S} \rangle_\ell$ corresponding to ${}^{A}\langle \mathbf{S} \rangle_\ell$ (if needed), by letting each $P_i \in \mathcal{P}$ send its share-vector $\mathbf{s}_i$ for ${}^{A}\langle \mathbf{S} \rangle_\ell$, along with the MAC-tag τ_{ij} on $\mathbf{s}_i$. Party P_j on receiving $\mathbf{s}_i$ and τ_{ij} can verify them using its MAC-key K_{ji} for ${}^{A}\langle \mathbf{S} \rangle_\ell$. Once P_j has $\ell + 1$ verified share-vectors for $\mathbf{S}$ (which it will have as there are at least $n - t$ honest P_i), it can use them to reconstruct each $\mathbf{s}^{(v)} \in \mathbf{S}$ (and if needed then $\langle \mathbf{s}^{(v)} \rangle_\ell$). The protocol requires one round and communication of $\mathcal{O}(nm\kappa)$ bits. As $|\mathbf{S}| = \Theta(mk\kappa)$ bits, if $k = \Theta(n)$, then the protocol has a communication overhead of $\mathcal{O}(1)$ bits. The procedure extends easily for ${}^{A}\langle \mathbb{S} \rangle_\ell$, provided $n - t > \ell$.

3.4 Existing Ideal Functionalities

The following ideal functionalities can be instantiated with $n = 3t + 1$.

– **Generating degree-t sharing ($\mathcal{F}_{[\cdot]_t}$, Functionality B.1)**: $\mathcal{F}_{[\cdot]_t}$ takes input N sharing polynomials from a designated dealer and distributes shares to the parties, only if the polynomials are of degree-t. To instantiate $\mathcal{F}_{[\cdot]_t}$, we use the VSS protocol from [1], with $\mathcal{O}(1)$ rounds and communication of $\mathsf{P2P}(N \cdot n\kappa) + n \times \mathsf{BC}(n\kappa)$ bits.
– **Generating random values ($\mathcal{F}_{\mathsf{Rand}}$, Functionality B.2)**: $\mathcal{F}_{\mathsf{Rand}}$ generates a uniformly random element $r \in \mathbb{F}$, only when invoked by at least one honest party so that the adversary will not know r beforehand. If there are N random values to be generated via N calls to $\mathcal{F}_{\mathsf{Rand}}$, then the parties can do the following: each party P_i can call $\mathcal{F}_{[\cdot]_t}$ with $\frac{N}{n-t}$ random elements. This is followed by extracting N random t-sharings using standard randomness extraction techniques based on Vandermonde matrix [6]. Finally, to make public the ℓ^{th} random value, the parties can publicly reconstruct the ℓ^{th} extracted t-sharing. After realizing $\mathcal{F}_{[\cdot]_t}$, this incurs $\mathcal{O}(1)$ rounds and communication of $\mathsf{P2P}(N \cdot n^2\kappa) + n \times \mathsf{BC}(n^2\kappa)$ bits. Note that each party can simultaneously broadcast all the data it needs to broadcast across all the n instances of $\mathcal{F}_{[\cdot]_t}$.
– **Generating degree-t twisted-sharing with 0 as twisted-shares for a designated set of parties ($\mathcal{F}_{\lceil\cdot\rfloor_t^i}$, Functionality B.3)**: $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ allows a designated dealer to verifiably generate twisted t-sharing of its input(s) with respect to a designated P_i, such that the twisted-shares of a designated subset of parties $\mathcal{T}$ of size up to $t+1$ (possibly including P_i) are 0. One can securely instantiate $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ for N inputs in $\mathcal{F}_{\mathsf{Rand}}$-hybrid model. After instantiating $\mathcal{F}_{\mathsf{Rand}}$, this incurs $\mathcal{O}(1)$ rounds and communication of $\mathsf{P2P}(N \cdot n\kappa + n^2\kappa) + n \times \mathsf{BC}(n^2\kappa)$ bits.
– **Generating random degree-t twisted-sharing with 0 as twisted-shares for a designated set of parties and designated slot for secret ($\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$, Functionality B.4)**: $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$ generates random twisted degree-t twisted t-sharings (with respect to a designated P_i) based on the corrupted parties' shares and then distribute the honest parties' shares to them, such that the twisted-shares of a designated subset of parties of size up to $t+1$ are 0. Moreover, the underlying secrets of the generated $\lceil\cdot\rfloor_t^i$-sharings will be embedded at a designated evaluation point $-\mathsf{pos}$ (instead of i), where $\mathsf{pos} \in \{1, \ldots, p+1\}$. To instantiate $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$ for N random values, each party can call $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ with $\frac{N}{n-t}$ random elements, followed by extracting N random sharings using standard randomness extraction techniques based on Vandermonde matrix [6]. After realizing $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$, this incurs $\mathcal{O}(1)$ rounds and communication of $\mathsf{P2P}(N \cdot n\kappa + n^2\kappa) + n \times \mathsf{BC}(n^2\kappa)$ bits. Note that a single random value can be used across all the n instances of $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ and so a single instance of $\mathcal{F}_{\mathsf{Rand}}$ suffices.

3.5 Broadcast and Parallel Broadcast with $n = 3t + 1$

If $n > 3t$, then one can use the perfectly-secure broadcast protocol of [3] over point-to-point channels. To emulate a broadcast of ℓ bit message, the protocol requires an expected $\mathcal{O}(1)$ rounds and a communication of $\mathsf{P2P}(n\ell)$ bits plus an expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. In our protocols, we will encounter scenarios where n parties wish to broadcast a message of size ℓ bits in parallel. For this, we can

again use the parallel broadcast protocol of [3] over point-to-point channels, which requires an expected $\mathcal{O}(1)$ rounds and a communication of $\mathsf{P2P}(n^2\ell)$ bits plus an expected $\mathsf{P2P}(n^3 \log^2 n)$ bits.

4 (Packed) Information Checking Protocol (ICP)

An ICP involves three entities: a *signer* $S \in \mathcal{P}$, an *intermediary* $I \in \mathcal{P}$ and the set $\mathcal{P}$ acting as *verifiers* (S and I also plays the role of verifiers). S has inputs $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)} \in \mathbb{F}^{\ell}$. Through ICP, S gives a "signature" on $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ to I, who later reveals it publicly, claiming that it got the signature from S.[9] An ICP consists of a sequence of following three phases.

- **Generation Phase**: Executed through a protocol Gen, where S sends $\mathbf{s}^{(1)}$, $\ldots, \mathbf{s}^{(M)}$ to I along with some *auxiliary information* and to each verifier, S gives some *verification information*.
- **Authentication Phase**: Executed through a protocol Auth, to verify whether S distributed "consistent" information to I and the verifiers. Upon successful verification I sets a Boolean variable $V_{S,I}$ to 1 and the information held by I is considered as S's *information-checking signature* (IC-signature) on $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ given to I, denoted as $\mathsf{Sig}(S \rightarrow I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$.
- **Public Reveal Phase**: Executed through a protocol Rev, where I publicly reveals $\mathsf{Sig}(S \rightarrow I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$, which gets verified by the parties, upon which the parties either output $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ or reject.

Definition 4.1 (ICP). *A triplet of protocols* $(\mathsf{Gen}, \mathsf{Auth}, \mathsf{Rev})$ *where* S *has input* $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)} \in \mathbb{F}^{\ell}$ *for* Gen *is called an ICP, if the following hold.*

- **Correctness***: If S and I are honest, then I sets $V_{S,I}$ to 1 during* Auth*. Moreover, every honest party outputs* $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ *during* Rev*.*
- **Privacy***: If S and I are honest, then the view of the adversary remains independent of* $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ *during* Gen *and* Auth*.*
- **Unforgeability***: If S is honest and I reveals* $\mathsf{Sig}(S \rightarrow I, \overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)})$ *during* Rev *and if the honest parties output* $\overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)}$*, then except with a negligible probability, the condition* $\overline{\mathbf{s}}^{(m)} = \mathbf{s}^{(m)}$ *holds, for* $m = 1, \ldots, M$*.*
- **Non-repudiation***: If S is corrupt and I is honest and if I sets $V_{S,I}$ to* 1 *holding* $\mathsf{Sig}(S \rightarrow I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$ *during* Auth*, then except with a negligible probability, all honest parties output* $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ *during* Rev*.*

The Construct. We present an ICP for $n = 3t + 1$. During Gen, S hides $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ in the higher order ℓ coefficients of random $(\ell + t - 1)$-degree polynomials $f^{(1)}(X), \ldots, f^{(M)}(X)$ respectively and sends them to I. And each verifier P_i receives a random evaluation-point α_i from S and the value of $f^{(1)}(X), \ldots, f^{(M)}(X)$ at α_i as *verification-tags*. The *privacy* of $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ for an *honest* S is maintained, as adversary learns at most t points on each polynomial. During Rev, I

[9] In the original notion of ICP [43], I reveals signature to a single verifier. Following [42], we extend this notion where signature is publicly revealed and verified.

makes public $f^{(1)}(X), \ldots, f^{(M)}(X)$, followed by each P_i locally verifying them through their verification-tags and publicly announcing the result. If at least $t+1$ verifiers respond positively, then with a high probability, I revealed the correct polynomials, as there is at least one *honest* verifier among those verifiers, whose verification-tags will not be known to the adversary. This ensures *unforgeability*.

Unfortunately, *non-repudiation* is not guaranteed yet, as a *corrupt* S can distribute "inconsistent" polynomials and verification-tags to I and an honest verifier respectively. So during Auth, the parties verify the consistency of the information distributed. For this, S distributes an additional random $(\ell+t-1)$-degree *masking polynomial* $R(X)$ to I during Gen and the value of $R(X)$ at α_i to each P_i. Then during Auth, the parties publicly verify if a random linear combination of $f^{(1)}(X), \ldots, f^{(M)}(X)$ and $R(X)$, is consistent with the same linear combination of the verification-tags of a set of parties $\mathcal{R}$ of size at least $n-t$. As a *corrupt* S will *not* know the random linear combination apriori, any inconsistency in the information distributed during Gen will be caught with a high probability. If a $\mathcal{R}$ exists, it will have at least $n-2t > t$ *honest* verifiers (provided $n = 3t+1$) who held consistent verification-tags and who will later respond positively to the polynomials revealed by an honest I.

Protocol 4.2: (Gen,Auth,Rev) – ICP

Generation Phase: Protocol $\mathsf{Gen}(S, I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$

Round 1: S with input $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)} \in F^{\ell}$, does the following for $m = 1, \ldots, N$.

- Select random $(\ell+t-1)$-degree polynomials $f^{(1)}(X), \ldots, f^{(M)}(X)$ over $\mathbb{F}$, where the higher order ℓ coefficients of $f^{(m)}(X)$ are elements of $\mathbf{s}^{(m)}$.
- Select a random $(\ell+t-1)$-degree *masking-polynomial* $R(x)$ over $\mathbb{F}$.
- Corresponding to each verifier P_i, select a random *evaluation-point* $\alpha_i \in \mathbb{F}$.
- Send $R(X), f^{(1)}(X), \ldots, f^{(M)}(X)$ to I.
- Send $\alpha_i, r_i, v_{1,i}, \ldots, v_{M,i}$ to each P_i, where $r_i = R(\alpha_i)$ and $v_{m,i} = f^{(m)}(\alpha_i)$.

Authentication Phase: Protocol $\mathsf{Auth}(S, I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$

Round 1: I on receiving $(\ell+t-1)$-degree polynomials $R(X), f^{(1)}(X), \ldots, f^{(M)}(X)$ from S during the protocol Gen, initializes a set $\mathcal{R}$ to $\emptyset$ and does the following.

- Select a random $d \in \mathbb{F}$ and compute the *masked-polynomial* $M(X) \stackrel{\mathsf{def}}{=} R(X) + df^{(1)}(X) + \ldots + d^M f^{(M)}(X)$ and broadcast $d, M(X)$.

Round 2: Every party $P_i \in \mathcal{P}$ (including S and I) upon receiving $d, M(X)$ from the broadcast of I, does the following.

- Broadcast $\mathtt{Accept}_i$, provided $M(\alpha_i) = r_i + dv_{1,i} + \ldots + d^M v_{M,i}$ holds.

Local Computation by I: I does the following.

- If P_i broadcasts $\mathtt{Accept}_i$, include P_i in $\mathcal{R}$.
- For $m = 1, \ldots, M$, let $\mathbf{s}^{(m)} \in \mathbb{F}^{\ell}$ be the higher order ℓ coefficients of $f^{(m)}(X)$, where $f^{(1)}, \ldots, f^{(M)}(X)$ has been received by I from S. If $|\mathcal{R}| \geq n-t$, then set $V_{S,I} = 1$ and $\mathsf{Sig}(\mathsf{sid}, S \to I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}) = (f^{(1)}(X), \ldots, f^{(M)}(X))$.

Public Reveal Phase: Protocol $\mathsf{Rev}(S, I, \mathcal{P}, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$

Round 1: If $V_{\mathsf{sid},S,I} = 1$ then I broadcasts $f^{(1)}(X), \ldots, f^{(M)}(X)$.

Round 2: Every party $P_i \in \mathcal{P}$ (including S and I) upon receiving $(\ell + t - 1)$-degree polynomials $f^{(1)}(X), \ldots, f^{(M)}(X)$ from the broadcast of I, broadcasts $\mathtt{Accept}_i$, provided P_i has earlier broadcast $\mathtt{Accept}_i$ during Auth and $f^{(m)}(\alpha_i) = v_{m,i}$ holds, for $m = 1, \ldots, M$.

Local Computation: Every $P_j \in \mathcal{P}$ (including S and I) does the following.

- If there are at least $n - 2t$ parties P_i such that $\mathtt{Accept}_i$ is received from $\mathcal{F}_{\mathsf{BC}}$ where P_i is the sender during Round 2 of Rev, then output $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$, where $\mathbf{s}^{(m)}$ constitute the higher order ℓ coefficients of $f^{(m)}(X)$.

Notation 4.3 (Terminologies for ICP)**.** *For any triplet P_i, P_j, P_k, acting as S, I and verifier respectively, we say that:*

- *"P_i gives $\mathsf{Sig}(P_i \to P_j, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$ to P_j": means P_i invokes Gen with input $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$.*
- *"P_j receives $\mathsf{Sig}(P_i \to P_j, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$ from P_i": means P_j sets V_{P_i,P_j} to 1 during Auth and holds $(\ell + t - 1)$-degree polynomials $f^{(1)}(X), \ldots, f^{(M)}(X)$, where the higher order ℓ coefficients of $f^{(m)}(X)$ are elements of $\mathbf{s}^{(m)}$.*
- *"P_j publicly reveals $\mathsf{Sig}(P_i \to P_j, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$": means P_j invokes Rev.*
- *"P_k accepts $\mathsf{Sig}(P_i \to P_j, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$": means P_k outputs $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ during Rev.*

Our construct has the following complexities: Gen requires one round and a communication of $\mathsf{P2P}((\ell M + nM)\kappa)$ bits. Protocol Auth requires two rounds and a communication of $\mathsf{BC}((\ell+n)\kappa) + n \times \mathsf{BC}(1)$ bits. Protocol Rev requires two rounds and a communication of $\mathsf{BC}((\ell M + nM)\kappa) + n \times \mathsf{BC}(1)$ bits; see Appendix C for the details.

Linearity of ICP. Consider any S and I let S has inputs $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(N)}$, where each $\mathbf{S}^{(k)} = (\mathbf{s}^{(k,1)}, \ldots, \mathbf{s}^{(k,M)})$ and where each $\mathbf{s}^{(k,1)}, \ldots, \mathbf{s}^{(k,M)} \in \mathbb{F}^{\ell}$. Let $\mathsf{Gen}^{(k)}$ and $\mathsf{Auth}^{(k)}$ be the instances of Gen and Auth respectively, used to generate $\mathsf{Sig}(S \to I, \mathbf{S}^{(k)})$, such that the following conditions are satisfied.

- S uses the same random α_i for P_i during $\mathsf{Gen}^{(1)}, \ldots, \mathsf{Gen}^{(N)}$.
- Let $\mathcal{R}_k$ be the instance of $\mathcal{R}$, computed during the instance $\mathsf{Auth}^{(k)}$. Then $|\mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N| \geq n - t$ holds.

Let $\mathbf{S} \stackrel{\mathsf{def}}{=} c_1\mathbf{S}^{(1)} + \ldots + c_N\mathbf{S}^{(N)}$, where $c_1, \ldots, c_N \in \mathbb{F}$ are public constants. If the above conditions are satisfied, then I can locally compute $\mathsf{Sig}(S \to I, \mathbf{S})$ from $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$. Similarly, every verifier in $\mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N$ can locally compute its verification-tags corresponding to $\mathsf{Sig}(S \to I, \mathbf{S})$, from its verification-tags corresponding to $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$; see Appendix C for the details.

Notation 4.4 (Terminologies for Linearity of ICP)**.** *We say that:*

- *"Parties follow the linearity property while generating IC-signatures", to mean that the underlying instances of* Gen *and* Auth *are invoked as above.*
- *"I locally computes* $\mathsf{Sig}(S \to I, \mathbf{S})$ *from* $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$*" to mean that I locally computes* $\mathsf{Sig}(S \to I, \mathbf{S})$ *as above and every party in* $\mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N$ *locally computes its verification-tags as above.*

5 VSS with $n = 3t + 1$ for APSS with $\mathcal{O}(1)$ Overhead

In protocol $\Pi_{A\langle\cdot\rangle}$ there is a dealer $D \in \mathcal{P}$ with input $\mathbb{S}$, where $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$, with each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)})$ and where each $\mathbf{s}^{(u,v)} = (s^{(u,v,1)}, \ldots, s^{(u,v,p)}) \in \mathbb{F}^p$. Hence $\mathbb{S} \in \mathbb{F}^{LMp^2}$, where $L, M \geq 1$. If D is *honest*, then at the end of the protocol parties hold ${}^A\langle\mathbb{S}\rangle_d$ and adversary's view remains independent of $\mathbb{S}$. Here $p = \frac{t}{4} + 1$ and $d = t + p - 1$. If D is *corrupt*, at the end of the protocol there is a well-defined $\mathbb{S} \in \mathbb{F}^{LMp^2}$ (held by D), such that the parties hold ${}^A\langle\mathbb{S}\rangle_d$.

Any ${}^A\langle\cdot\rangle_d$-sharing generated via our VSS will be key-consistent. For this, it would be ensured that the parties *apriori* hold a set-up for using consistent MAC keys (the set-up will be generated within the MPC). Namely, for every pair (P_i, P_j), there exists $\boldsymbol{\mu}_{ji} = (u_{ji}^{(1)}, \ldots, u_{ji}^{(Mp)}) \in \mathbb{F}^{Mp}$ held by P_j, such that the parties hold their respective shares for $[\boldsymbol{\mu}_{ji}]_t^i$, where shares of P_i are 0. If P_j is *honest* and P_i is *corrupt*, then $\boldsymbol{\mu}_{ji}$ will be random for P_i. Interestingly, if both P_i and P_j are *honest* then adversary will completely know $\boldsymbol{\mu}_{ji}$, since it will have t twisted-shares of the corrupt parties and twisted-share of P_i which is 0. However, this is fine since an honest P_i will never forge an incorrect MAC to P_j.

The high-level overview has been already discussed in detail in Section 2.2, assuming $L = 1$. Now D will have L batches of bivariate polynomials for $\mathbb{S}$. Throughout D has to reveal linear combination of its bivariate polynomials *four* times. For privacy of $\mathbb{S}$, dealer D additionally picks 4 batches of random "masking data" $\mathbf{S}^{(L+1)}, \ldots, \mathbf{S}^{(L+4)}$, each consisting of Mp^2 random elements. During tag generation, for every pair (P_i, P_j) and $u \in \{1, \ldots, L+4\}$, we want to compute $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}$ and reconstruct the tag $\tau_{ij}^{(u)}$ for P_i, where $\tau_{ij}^{(u)} = \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$. To randomize $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}$, we mask it with the following sharing:

$$\lceil m \rfloor_{t+d}^i = [[\mathbf{e}_1]]_d^i \cdot \lceil r_{ji}^{(u,1)} || 1 \rfloor_t + \ldots + [[\mathbf{e}_d]]_d^i \cdot \lceil r_{ji}^{(u,d)} || d \rfloor_t.$$

Here $\lceil r_{ji}^{(u,1)} || 1 \rfloor_t^i, \ldots, \lceil r_{ji}^{(u,d)} || d \rfloor_t^i$ are t-sharings of random values $r_{ji}^{(u,1)}, \ldots, r_{ji}^{(u,d)}$, where $r_{ji}^{(u,1)}, \ldots, r_{ji}^{(u,d)}$ are embedded at $-1, \ldots, -d$ respectively in the underlying sharing polynomials. Moreover, $\mathbf{e}_1, \ldots, \mathbf{e}_d \in \mathbb{F}^{d+1}$ are *publicly known* vectors, where $\mathbf{e}_1 = (0, 1, 0, 0, \ldots, 0), \mathbf{e}_2 = (0, 0, 1, 0, \ldots, 0), \ldots \mathbf{e}_d = (0, \ldots, 0, 1)$. Furthermore, $[[\mathbf{e}_1]]_d^i, \ldots, [[\mathbf{e}_d]]_d^i$ denotes publicly-known degree-d degenerate packed twisted-sharing (see Definition 3.6) of $\mathbf{e}_1, \ldots, \mathbf{e}_d$ respectively with respect to P_i. In essence, for each $[[\mathbf{e}_k]]_d^i$ the underlying sharing polynomial evaluates to 0 at i, evaluates to 1 at $-k$ and evaluates to 0 at every $j \in \{-1, \ldots, -d\} \setminus \{-k\}$. It is

easy to see that the underlying sharing polynomial $f_m(X)$ for $\lceil m \rfloor^i_{t+d}$ evaluates to 0 at $X = i$. Hence $m = 0$. Moreover, $f_m(X)$ evaluates to $r^{(u,1)}_{ji}, \ldots, r^{(u,d+1)}_{ji}$ at $X = -1, \ldots, -d$ respectively. Since $r^{(u,1)}_{ji}, \ldots, r^{(u,d)}_{ji}$ are random and adversary will know at most $t+1$ points on $f_m(X)$ (the shares of corrupt parties and the i^{th} point which is 0), this leaves d degree-of-freedom in $f_m(X)$ as desired.

We note that P_l's tag-share $\tau^{(u)}_{ij,l}$ for $\lceil \tau^{(u)}_{ij} \rfloor^i_{t+d}$ (after masking) should be:

$$\mathbf{s}^{(u)}_l \cdot \boldsymbol{\mu}_{ji,l} + v^{(u)}_{ji,l} + \mathbf{e}_{1,l} \cdot r^{(u,1)}_{ji,l} + \ldots + \mathbf{e}_{d,l} \cdot r^{(u,d)}_{ji,l},$$

where $\boldsymbol{\mu}_{ji,l}$ and $v^{(u)}_{ji,l}$ are P_l's shares in $\lceil \boldsymbol{\mu}_{ji} \rfloor^i_t$ and $\lceil v^{(u)}_{ji} \rfloor^i_t$ respectively. And $\mathbf{e}_{k,l}$ and $r^{(u,k)}_{ji,l}$ denote P_l's shares corresponding to $[[\mathbf{e}_k]]^i_d$ and $\lceil r^{(u,k)}_{ji} || k \rfloor^i_t$ respectively. To verify P_l's tag-share $\tau^{(u)}_{ij,l}$, a naive (privacy-free) solution could be to let D make public $\mathbf{s}^{(u)}_l$ in a verifiable way. Then consider the t-sharing

$$\lceil \mathbf{s}^{(u)}_l \rfloor^i_0 \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor^i_t + \lceil v^{(u)}_{ji} \rfloor^i_t + \mathbf{e}_{1,l} \cdot [r^{(u,1)}_{ji} || 1]^i_t + \ldots + \mathbf{e}_{d,l} \cdot [r^{(u,d)}_{ji} || d]^i_t,$$

where P_l's share is $\tau^{(u)}_{ij,l}$. So P_i can verify P_l's tag-share by reconstructing the above t-sharing. For privacy, instead of applying the above idea on the values $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L+4)}$, we take their random linear combination. If D is ever identified to be cheating at any step, then the parties terminate with a default ${}^A\langle 0^{LMp^2} \rangle_d$.

Protocol 5.1: $\Pi_{A\langle\cdot\rangle}$ **– VSS in** $(\mathcal{F}_{\mathsf{Rand}}, \mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor^i_t})$**-hybrid model**

Initialization: Parties hold public sets $\mathsf{Dp}, \mathsf{Cr} \subset \mathcal{P}$, initialized to $\emptyset$ and carried over if the parties restart the protocol. They also maintain public Boolean variables flag_1 and flag_2, initialized to 0, to track whether the sharing-phase and the pairwise tag-generation phase failed earlier (for the same D).

Input: D has input $\mathbb{S} \in \mathbb{F}^{LMp^2}$, where $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$ and each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)})$. Moreover, each $\mathbf{s}^{(u,v)} = (s^{(u,v,1)}, \ldots, s^{(u,v,p)})$.

Setup: For every pair (P_i, P_j), there exists a $\boldsymbol{\mu}_{ji} = (u^{(1)}_{ji}, \ldots, u^{(Mp)}_{ji})$ held by P_j, such that the parties hold $\lceil \boldsymbol{\mu}_{ji} \rfloor^i_t$, where the twisted-shares of P_i are 0.

(Sharing Phase)

1. D randomly selects $\mathbf{S}^{(L+1)}, \ldots, \mathbf{S}^{(L+4)} \in \mathbb{F}^{Mp^2}$, where $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)})$ for $u = L+1, \ldots, L+4$ and $\mathbf{s}^{(u,v)} = (s^{(u,v,1)}, \ldots, s^{(u,v,p)})$, for $v = 1, \ldots, Mp$.
2. For $u = 1, \ldots, L+4$ and $v = 1, \ldots, M$, dealer D picks a bivariate polynomial $F^{(u,v)}(X, Y)$ over $\mathbb{F}$ of degree d in each variable where $d \stackrel{\mathsf{def}}{=} t + p - 1$. Each $F^{(u,v)}(X, Y)$ is otherwise random, except the following:
   - $F^{(u,v)}(X, -1), \ldots, F^{(u,v)}(X, -p)$ embeds $\mathbf{s}^{(u,(v-1)\cdot p+1)}, \ldots, \mathbf{s}^{(u,v\cdot p)}$ respectively. Namely, elements of $\mathbf{s}^{(u,(v-1)\cdot p+1)}$ are embedded at $X = -1, \ldots, -p$ in $F^{(u,v)}(X, -1)$ and likewise for other polynomials.
   - $F^{(u,v)}(X, i) = F^{(u,v)}(i, Y) = 0$, for every $P_i \in \mathsf{Dp} \cup \mathsf{Cr}$.

3. D sends $\{f_i^{(u,v)}(X), g_i^{(u,v)}(Y)\}_{u=1,\ldots,L+4,v=1,\ldots,M}$ to every $P_i \notin \mathsf{Dp} \cup \mathsf{Cr}$, where $f_i^{(u,v)}(X) \stackrel{\text{def}}{=} F^{(u,v)}(X,i)$ and $g_i^{(u,v)}(Y) \stackrel{\text{def}}{=} F^{(u,v)}(i,Y)$.
4. On receiving d-degree polynomials $\{f_i^{(u,v)}(X), g_i^{(u,v)}(Y)\}_{u=1,\ldots,L+4,v=1,\ldots,M}$ from D, every $P_i \notin \mathsf{Dp} \cup \mathsf{Cr}$ does the following.
   - Broadcast OK_i.
   - For $u = 1, \ldots, L+4$, give $\mathsf{Sig}(P_i \to D, \mathbf{g}_i^{(u,1)}, \ldots, \mathbf{g}_i^{(u,M)})$ to D, *where the parties follow the linearity property while generating IC-signatures* (see Notation 4.3 and 4.4). Here for $v = 1, \ldots, M$, $\mathbf{g}_i^{(u,v)} \stackrel{\text{def}}{=} (g_i^{(u,v)}(1), \ldots, g_i^{(u,v)}(n))$. % Here P_i is invoking $L+4$ instances of ICP, where in each instance, it is signing M vectors for D, each consisting of n values.
5. For $u = 1, \ldots, L+4$, upon receiving $\mathsf{Sig}(P_i \to D, \mathbf{g}_i^{(u,1)}, \ldots, \mathbf{g}_i^{(u,M)})$ from $P_i \notin \mathsf{Dp} \cup \mathsf{Cr}$ and upon receiving OK_i from the broadcast of P_i, dealer D includes P_i to Core (initialized to $\emptyset$), provided $\mathbf{g}_i^{(u,v)} = (g_i^{(u,v)}(1), \ldots, g_i^{(u,v)}(n))$ holds, for $v = 1, \ldots, M$.
   - If $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t + p$, then D broadcasts Core.
   - Else D sets $\mathsf{Dp} = (\mathcal{P} \setminus \mathsf{Core})$ and broadcasts Dp.
6. The parties in $\mathcal{P}$ do the following.
   - If D broadcasts Core is where $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t + p$ then do:
     - If OK_i is received from the broadcast of each $P_i \in \mathsf{Core}$, then proceed to the next phase.
     - Else discard D and terminate.
   - Else if D broadcasts Dp then do:
     - If $\mathsf{flag}_1 = 0$ and $t - p + 1 \leq |\mathsf{Dp}| \leq t$, then set $\mathsf{flag}_1 = 1$, update Dp and restart the protocol.
     - Else discard D and terminate.
   - Else discard D and terminate the protocol.

(Verification Phase)

1. Every $P_i \in \mathcal{P}$ sends Rand to $\mathcal{F}_{\mathsf{Rand}}$ and receives r_1 from $\mathcal{F}_{\mathsf{Rand}}$.
2. The parties invoke $\mathsf{Ver}(r_1)$ (see below the steps of Ver). If D is not discarded after executing the steps of $\mathsf{Ver}(r_1)$, then proceed to the next phase. % The code of Ver will be invoked 4 times within this protocol. So to avoid repetition, we will invoke Ver every time with different randomness.

Verification Process $\mathsf{Ver}(r_\alpha) : \alpha \in \{1, \ldots, 4\}$

1. For $v = 1, \ldots, M$, dealer D does the following.
   - $\mathsf{F}^{(\alpha,v)}(X,Y) \stackrel{\text{def}}{=} F^{(1,v)}(X,Y) + r_\alpha \cdot F^{(2,v)}(X,Y) + \ldots + r_\alpha^{L+3} \cdot F^{(L+4,v)}(X,Y)$.
   - For every $P_i \in \mathsf{Core}$, $\mathsf{G}_i^{(\alpha,v)}(y) \stackrel{\text{def}}{=} \mathsf{F}^{(\alpha,v)}(i,Y)$. Let $\mathsf{G}_i^{(\alpha,v)} \stackrel{\text{def}}{=} (\mathsf{G}_i^{(\alpha,v)}(1), \ldots, \mathsf{G}_i^{(\alpha,v)}(n))$.
   - For every $P_i \in \mathsf{Core}$, D locally computes the signature $\mathsf{Sig}(P_i \to D, \mathsf{G}_i^{(1,1)}, \ldots, \mathsf{G}_i^{(1,M)})$ from $\mathsf{Sig}(P_i \to D, \mathbf{g}_i^{(1,1)}, \ldots, \mathbf{g}_i^{(1,M)}), \mathsf{Sig}(P_i \to D, \mathbf{g}_i^{(2,1)}, \ldots, \mathbf{g}_i^{(2,M)}), \ldots, \mathsf{Sig}(P_i \to D, \mathbf{g}_i^{(L+4,1)}, \ldots, \mathbf{g}_i^{(L+4,M)})$.

D broadcasts $(\mathsf{F}^{(\alpha,1)}(X,Y),\ldots,\mathsf{F}^{(\alpha,M)}(X,Y))$. For every $P_i \in \mathsf{Core}$, D publicly reveals $\mathsf{Sig}(P_i \rightarrow D, \mathsf{G}_i^{(1,1)},\ldots,\mathsf{G}_i^{(1,M)})$.

2. Every $P_k \in \mathcal{P}$ discards D if any of the following holds.
   - D broadcasts $(\mathsf{F}^{(\alpha,1)}(X,Y),\ldots,\mathsf{F}^{(\alpha,M)}(X,Y)))$ and there exists some $v \in \{1,\ldots,M\}$, such that:
     - $\mathsf{F}^{(\alpha,v)}(X,Y)$ is not a (d,d)-degree bivariate polynomial.
     - $\mathsf{F}^{(\alpha,v)}(X,i) \neq \mathbf{0}$ or $\mathsf{F}^{(\alpha,v)}(i,Y) \neq \mathbf{0}$ for some $P_i \in \mathsf{Dp} \cup \mathsf{Cr}$.
   - $\exists P_i \in \mathsf{Core}$, such that P_k does not accept $\mathsf{Sig}(P_i \rightarrow D, \mathsf{G}_i^{(1,1)},\ldots,\mathsf{G}_i^{(1,M)})$.
   - $\exists P_i \in \mathsf{Core}, v \in \{1,\ldots,M\}$, where values in $\mathsf{G}_i^{(\alpha,v)}$ do not lie on $\mathsf{F}^{(\alpha,v)}(i,Y)$.

(Non-Core Row and Column Reconstruction Phase)

1. Each $P_i \in \mathsf{Core}$ sends $\{(f_i^{(u,v)}(j), g_i^{(u,v)}(j))\}_{u=1,\ldots,L+4,v=1,\ldots,M}$ to each $P_j \in \mathcal{P} \setminus (\mathsf{Core} \cup \mathsf{Dp} \cup \mathsf{Cr})$.
2. Parties send Rand to $\mathcal{F}_{\mathsf{Rand}}$, receive r_2 and invoke $\mathsf{Ver}(r_2)$. If D is not discarded during $\mathsf{Ver}(r_2)$, every $P_j \in \mathcal{P} \setminus (\mathsf{Core} \cup \mathsf{Dp} \cup \mathsf{Cr})$ does the following:
   - On receiving $\{(f_i^{(u,v)}(j), g_i^{(u,v)}(j))\}_{u=1,\ldots,L+4,v=1,\ldots,M}$ from $P_i \in \mathsf{Core}$, include P_i to $\mathcal{S}_j$ (initialized to $\emptyset$), if $\forall v \in \{1,\ldots,M\}$, following hold:

$$\mathsf{F}^{(2,v)}(i,j) = \sum_{u=1}^{L+4} (r_2)^{u-1} g_i^{(u,v)}(j), \quad \mathsf{F}^{(2,v)}(j,i) = \sum_{u=1}^{L+4} (r_2)^{u-1} f_i^{(u,v)}(j).$$

   - $\forall u \in \{1,\ldots,L+4\}, v \in \{1,\ldots,M\}$, interpolate $\{(i, f_i^{(u,v)}(j))\}_{P_i \in \mathcal{S}_j} \cup \{(i,0)\}_{P_i \in \mathsf{Dp}}$ to recompute $g_i^{(u,v)}(Y)$ and interpolate $\{(i, g_i^{(u,v)}(j))\}_{P_i \in \mathcal{S}_j} \cup \{(i,0)\}_{P_i \in \mathsf{Dp}}$ to recompute the polynomial $f_i^{(u,v)}(X)$.

(Pairwise Tag Generation)

1. For each ordered pair (P_i, P_j) where $P_i \notin \mathsf{Cr}$, parties prepare the following by calling $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$ appropriately.
   (a) Random twisted t-sharings $\lceil v_{ji}^{(u)} \rfloor_t^i, \lceil r_{ji}^{(u,1)} || 1 \rfloor_t^i, \lceil r_{ji}^{(u,2)} || 2 \rfloor_t^i, \ldots, \lceil r_{ji}^{(u,d)} || d \rfloor_t^i$ with respect to P_i, where the twisted-shares of the parties in Cr are 0.
   (b) Corresponding to each $P_l \notin \mathsf{Cr}$, random twisted t-sharings $\lceil \mathtt{mask}_{ji}^{(l)} || 1 \rfloor_t^i$ and $\lceil \mathtt{mask}_{ji}^{(l)} || 1 \rfloor_t^i$ with respect to P_i, where the shares of P_l are 0.
2. **(Tag Share Computation)**: For every ordered pair (P_i, P_j) where $P_i \notin \mathsf{Cr}$, let $\tau_{ij}^{(u)} \stackrel{\mathsf{def}}{=} \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$ and for $u = 1,\ldots,L+4$, let

$$\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^i = \lceil \mathbf{s}_i^{(u)} \rfloor_d^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \lceil v_{ji}^{(u)} \rfloor_t^i + \sum_{k=1}^{d} [[\mathbf{e}_k]]_d^i \cdot \lceil r_{ji}^{(u,k)} || k \rfloor_t^i,$$

where, $\mathbf{s}_i^{(u)} \stackrel{\mathsf{def}}{=} (g_i^{(u,1)}(-1),\ldots,g_i^{(u,1)}(-p),\ldots,g_i^{(u,M)}(-1),\ldots,g_i^{(u,M)}(-p))$. Here $\mathbf{e}_1,\ldots,\mathbf{e}_d \in \mathbb{F}^{d+1}$, where $\mathbf{e}_1 = (0,1,0,\ldots,0), \mathbf{e}_2 = (0,0,1,0,\ldots,0),\ldots,$ $\mathbf{e}_d = (0,\ldots,0,1)$. Moreover, $[[\mathbf{e}_1]]_d^i,\ldots,[[\mathbf{e}_d]]_d^i$ denote degree-d degenerate packed twisted sharing of $\mathbf{e}_1,\ldots,\mathbf{e}_d$ (Definition 3.6). Every $P_l \notin \mathsf{Cr}$ locally computes its tag-share $\tau_{ij,l}^{(u)}$ for $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^i$ and sends it to P_i.

3. **(Tag Share Verification)**
   (a) The parties call Rand, receive r_3 and invoke $\mathsf{Ver}(r_3)$.% 3rd call of Ver. If D is not discarded during $\mathsf{Ver}(r_3)$, then for every ordered triplet (P_i, P_j, P_l) where $P_i, P_l \notin \mathsf{Cr}$, the parties do the following.
   - Let $\mathbf{s}_l \stackrel{\mathsf{def}}{=} (\mathsf{F}^{(3,1)}(l,-1), \ldots, \mathsf{F}^{(3,1)}(l,-p), \ldots, \mathsf{F}^{(3,M)}(l,-1), \mathsf{F}^{(3,M)}(l,-1), \ldots, \mathsf{F}^{(3,M)}(l,-p))$, where $\mathbf{s}_l \in \mathbb{F}^{M \cdot p}$. Parties compute

$$\lceil \beta_{ij,l} \rfloor_t^i = \lceil \mathbf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \sum_{\nu=1}^{L+3} (r_3)^{\nu-1} \left(\lceil v_{ji}^{(\nu)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(\nu,k)} || k \rfloor_t^i \right) + \lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$$

     Here $[\mathbf{s}_l]_0^i$ denotes the default degree-0 twisted sharing of $\mathbf{s}_l$, where all twisted-shares are $\mathbf{s}_l$. And $e_{1,l}, \ldots, e_{d,l}$ denotes P_l's shares of $[[\mathbf{e}_1]]_d^i, \ldots, [[\mathbf{e}_d]]_d^i$. The parties reconstruct $\lceil \beta_{ij,l} \rfloor_t^i$ towards P_i.
   - If P_i receives $(\tau_{ij,l}^{(1)}, \ldots, \tau_{ij,l}^{(L+4)})$ from P_l, then it includes P_l to $\mathcal{R}_{ij}$, provided P_l's share in the reconstructed sharing $\lceil \beta_{ij,l} \rfloor_t^i$ is same as $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$.

   (b) Every $P_i \notin \mathsf{Cr}$ does the following.
   - If for every P_j, $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \geq 2t+p$, then interpolate $\{(l, \tau_{ij,l}^{(u)})\}_{P_l \in \mathcal{R}_{ij}} \cup \{(l,0)\}_{P_l \in \mathsf{Cr}} \cup (0, \tau_{ij,i}^{(u)})$ to compute $\tau_{ij}^{(u)}$ and broadcast Tag_i.
   - Else include P_l to Dp_i, provided there is some P_j such that $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \leq 2t+p$ and $P_l \notin \mathcal{R}_{ij} \cup \mathsf{Cr}$. Broadcast $(\mathsf{NOK}, \mathsf{Dp}_i)$.

   (c) If $\mathsf{flag}_2 = 1$ OR if Tag_k is received from the broadcast of every $P_k \notin \mathsf{Cr}$, then the parties do the following.
   - For every ordered pair (P_i, P_j) where $P_i \notin \mathsf{Cr}$, parties reconstruct $\lceil v_{ji}^{(u)} \rfloor_t^i$ towards P_j, for every $u \in \{1, \ldots, L\}$.
   - For $u = 1, \ldots, L$, each $P_i \notin \mathsf{Cr}$ outputs $\mathbf{s}_i^{(u)}$ as its shares, $(\tau_{i1}^{(u)}, \ldots, \tau_{in}^{(u)})$ as MAC-tags and $(\mathsf{K}_{i1}^{(u)}, \ldots, \mathsf{K}_{in}^{(u)})$ as MAC-keys corresponding to ${}^A\langle \mathbf{S}^{(u)} \rangle_d$. Moreover, based on its information corresponding to ${}^A\langle \mathbf{S}^{(1)} \rangle_d$, $\ldots$, ${}^A\langle \mathbf{S}^{(L)} \rangle_d$, party P_i outputs its respective information corresponding to ${}^A\langle \mathbb{S} \rangle_d$. Here $\mathsf{K}_{ij}^{(u)} = (\boldsymbol{\mu}_{ij}, v_{ij}^{(u)})$ for every $P_j \notin \mathsf{Cr}$, while $\mathsf{K}_{ij}^{(u)} = (\boldsymbol{\mu}_{ij}, 0)$ for every $P_j \in \mathsf{Cr}$.

   (d) Else the parties proceed to the next phase.

4. **(Tag Share Complaint Resolution)**
   (a) If P_i broadcasts $(\mathsf{NOK}, \mathsf{Dp}_i)$ such that $|\mathsf{Dp}_i| \leq t$, then every $P_l \in \mathsf{Dp}_i$ broadcasts $\{(\tau_{ij,l}^{(1)}, \ldots, \tau_{ij,l}^{(L+4)})\}_{j=1,\ldots,n}$.

   (b) The parties send Rand to $\mathcal{F}_{\mathsf{Rand}}$, receive r_4 and invoke $\mathsf{Ver}(r_4)$. % 4th call of Ver. If D is not discarded during $\mathsf{Ver}(r_4)$, then for every pair (P_i, P_l) where $(\mathsf{NOK}, \mathsf{Dp}_i)$ is received from the broadcast of P_i and where $P_l \in \mathsf{Dp}_i$, the parties do the following upon receiving $\{\tau_{ij,l}^{(1)}, \ldots, \tau_{ij,l}^{(L+4)}\}_{j=1,\ldots,n}$ from the broadcast of P_l.
   - Let $\mathsf{s}_l \stackrel{\mathsf{def}}{=} (\mathsf{F}^{(4,1)}(l,-1), \ldots, \mathsf{F}^{(4,1)}(l,-p), \ldots, \mathsf{F}^{(4,M)}(l,0), \mathsf{F}^{(4,M)}(l,-1), \ldots, \mathsf{F}^{(4,M)}(l,-p))$, where $\mathsf{s}_l \in \mathsf{s}_l$. For $j = 1, \ldots, n$, parties compute

$$\lceil \gamma_{ij,l} \rfloor_t^i = \lceil \mathsf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \sum_{\nu=1}^{L+3} (r_4)^{\nu-1} \left(\lceil v_{ji}^{(\nu)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(\nu,k)} || k \rfloor_t^i \right) + \lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$$

Here $\lceil \mathsf{s}_l \rfloor_0^i$ is degree-0 twisted-sharing of s_l where all twisted-shares are s_l. Parties publicly reconstruct $\lceil \gamma_{i1,l} \rfloor_t^i, \ldots, \lceil \gamma_{in,l} \rfloor_t^i$.

- If $\exists j \in \{1, \ldots, n\}$, where P_l's share in the reconstructed $[\gamma_{ij,l}]_t^i$ is different from $\tau_{ij,l}^{(1)} + r_4 \cdot \tau_{ij,l}^{(2)} + \ldots + r_4^{L+3} \cdot \tau_{ij,l}^{(L+4)}$, include P_l to Cr.
- Else P_i removes P_l from Dp_i and includes it to $\mathcal{R}_{ij}$, for $j = 1, \ldots, n$.

(c) If $|\mathsf{Cr}| \geq t - p + 1$, then the parties set $\mathsf{flag}_2 = 1$ and restart the protocol.

(d) Else the parties do the following.

- For every order pair of parties (P_i, P_j) where $P_i \notin \mathsf{Cr}$, parties reconstruct $\lceil v_{ji}^{(u)} \rfloor_t^i$ towards P_j, for every $u \in \{1, \ldots, L\}$.
- Each $P_i \notin \mathsf{Cr}$ does the following. If $\exists P_j \in \mathcal{P}$ for which the MAC-tags are not yet computed then interpolate $\{(l, \tau_{ij,l}^{(u)})\}_{P_l \in \mathcal{R}_{ij}} \cup \{(l, 0)\}_{P_l \in \mathsf{Cr}} \cup (0, \tau_{ij,i}^{(u)})$ to compute $\tau_{ij}^{(u)}$. Here if P_l is added to $\mathcal{R}_{ij}$ in this phase, then consider $\tau_{ij,l}^{(u)}$ broadcasted by P_l in this phase.

  For $u = 1, \ldots, L$, output $\mathbf{s}_i^{(u)}$ as shares, $(\tau_{i1}^{(u)}, \ldots, \tau_{in}^{(u)})$ as MAC-tags and $(\mathsf{K}_{i1}^{(u)}, \ldots, \mathsf{K}_{in}^{(u)})$ as MAC-keys corresponding to ${}^A\langle \mathbf{S}^{(u)} \rangle_d$. Based on this, P_i outputs its respective information corresponding to ${}^A\langle \mathbb{S} \rangle_d$. Here $\mathsf{K}_{ij}^{(u)} = (\boldsymbol{\mu}_{ij}, v_{ij}^{(u)})$ for every $P_j \notin \mathsf{Cr}$, while $\mathsf{K}_{ij}^{(u)} = (\boldsymbol{\mu}_{ij}, 0)$ for every $P_j \in \mathsf{Cr}$.

In Appendix D, we prove the following.

Theorem 5.2. *Let $n = 3t+1$ and let D has input $\mathbb{S} \in \mathbb{F}^{LMp^2}$ for protocol $\Pi_{A\langle\cdot\rangle}$, where $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$ and each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)})$. Protocol $\Pi_{A\langle\cdot\rangle}$ generates key-consistent sharing ${}^A\langle \mathbb{S} \rangle_d = ({}^A\langle \mathbf{S}^{(1)} \rangle_d, \ldots, {}^A\langle \mathbf{S}^{(L)} \rangle_d)$, such that the view of the adversary remains independent of $\mathbb{S}$ if D is honest. Moreover, even if D is corrupt, there exist some $\mathbb{S} \in \mathbb{F}^{LMp^2}$ known to D, such that the protocol generates ${}^A\langle \mathbb{S} \rangle_d$. Here $p = \frac{t}{4} + 1$ and $d = t + p - 1$.*

The protocol requires a constant number of rounds and a communication of $\mathsf{P2P}((n^2LM + n^4L + n^5)\kappa) + \mathsf{BC}(n^2L\kappa) + \mathsf{BC}(n^2M\kappa) + n \times \mathsf{BC}(n^2L\kappa)$ bits, apart from at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$.

Protocol $\Pi_{A\langle\cdot\rangle}$ Over Point-to-Point Channels and $\mathcal{O}(1)$ Overhead. If $n = 3t + 1$, we can use the protocols of [3] to instantiate the broadcast by individual designated parties and parallel broadcasts, where all n parties need to simultaneously broadcast some messages in protocol $\Pi_{A\langle\cdot\rangle}$. After securely realizing $\mathcal{F}_{\mathsf{Rand}}$ over point-to-point channels, the resultant protocol will then require an $\mathcal{O}(1)$ expected number of rounds over point-to-point channels. The communication complexity will be $\mathsf{P2P}((n^2LM + n^4L + n^3M + n^5)\kappa)$ bits plus an additional

expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. If we set $L = \Theta(n)$ and $M = \Theta(n^2)$, then $\mathbb{S}$ contains $\Theta(n^5)$ field elements and hence $\Theta(n^5\kappa)$ bits and the protocol will require a communication of $\mathsf{P2P}(n^5\kappa)$ bits and an additional expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. Hence, the protocol will generate ${}^{A}\langle \mathbb{S} \rangle_d$ with $\mathcal{O}(1)$ overhead.

From the above analysis it follows that we *cannot* set $L = 1$ and achieve $\mathcal{O}(1)$ overhead, since in that case the number of secrets shared will be only $n^2 M$, but we have a term with $n^3 M$ in the overall communication. This explains the reason for considering three-dimensional APSS for achieving $\mathcal{O}(1)$ overhead.

Protocol $\Pi_{A\langle\cdot\rangle}$ for Degree $p-1$ and $d+p-1$. As mentioned earlier, in our MPC protocol, we will come across situations where parties need to verifiably generate degree-ℓ APSS of their inputs where $\ell \in \{p-1, d, d+p-1\}$. We have already discussed how to do this via protocol $\Pi_{A\langle\cdot\rangle}$ for the case $\ell = d$. In Appendix D, we discuss how to modify protocol $\Pi_{A\langle\cdot\rangle}$ to handle the cases $\ell = p-1$ and $\ell = d+p-1$, *without* affecting the communication complexity. Note that with $n = 3t+1$, we can robustly reconstruct a $(p-1)$-degree polynomial using RS error-correction (since $p = \frac{t}{4}+1$) and no MAC tags are needed. Hence for the case of $\ell = p-1$, it is sufficient to just generate PSS via $\Pi_{A\langle\cdot\rangle}$, instead of APSS and one can skip the tag-generation phase. However, we still generate the authentication tags even for $\ell = p-1$ to maintain a generic protocol.

6 Multiplication Over APSS with a Constant Overhead

We present our multiplication protocol Π_{Beaver} (Protocol 6.1) for securely multiplying vectors of ${}^{A}\langle\cdot\rangle_d$-shared values. We have already explained the high-level overview of the protocol for one-dimensional vectors in Section 2.3; here we extend that idea for three-dimensional vectors to get $\mathcal{O}(1)$ communication overhead via amortization. The protocol assumes the existence of a *robust* degree-reduction protocol Π_{RoDegRed} (Protocol F.9) which takes input ${}^{A}\langle \mathbb{S}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_d$ and outputs ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$, where each $\mathbb{S}^{(i)} \in \mathbb{F}^{n^3 p^2}$.

Protocol 6.1: Multiplication Protocol for APSS – Π_{Beaver}

Initialization: Each $P_i \in \mathcal{P}$ initializes $\mathsf{C} = \emptyset$ and $\mathsf{Dp}_i = \emptyset$

Input: For $\gamma \in \{1, \ldots, t+1\}$, parties hold.

- Multiplicands $({}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d)$.
- Triples $({}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d)$ and random pairs $({}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1})$.
- Random triples $({}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_d)$ and random pairs $({}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1})$.

The protocol: For $\gamma = 1, \ldots, t+1$, parties do the following.

1. Locally compute ${}^{A}\langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d + {}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d + {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d$.
2. Invoke the degree reduction protocol Π_{RoDegRed} (Protocol F.9) with inputs $\{{}^{A}\langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_d\}$ to obtain $\{{}^{A}\langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_{p-1}\}$ and $\{{}^{A}\langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_{p-1}\}$ respectively.

3. Locally compute the PSS $\langle \mathbb{Z}^{(\gamma)} \rangle_{d+p-1} = \langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_{p-1} \times \langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_{p-1} - \langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_{p-1} \times \langle \mathbb{B}^{(\gamma)} \rangle_{d} - \langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_{p-1} \times \langle \mathbb{A}^{(\gamma)} \rangle_{d} + \langle \mathbb{C}^{(\gamma)} \rangle_{d}$ and $\langle \mathbb{S}^{(\gamma)} \rangle_{d+p-1} = \langle \mathbb{Z}^{(\gamma)} \rangle_{d+p-1} + \langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1}$

Note that for each $\mathbb{L}^{(\gamma)}$ above where $\mathbb{L}^{(\gamma)} \in \{\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)}, \mathbb{R}^{(\gamma)}, \hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)}, \hat{\mathbb{R}}^{(\gamma)}, \mathbb{Z}^{(\gamma)}, \mathbb{S}^{(\gamma)}\}$, we have $\mathbb{L}^{(\gamma)} \in \mathbb{F}^{n^3 p^2}$ and ${}^{A}\langle \mathbb{L}^{(\gamma)} \rangle_{\ell} = ({}^{A}\langle \mathbf{L}^{(\gamma,1)} \rangle_{\ell}, \ldots, {}^{A}\langle \mathbf{L}^{(\gamma,n)} \rangle_{\ell})$ for each $\ell \in \{d, d+p-1\}$, where each $\mathbf{L}^{(\gamma,j)} \in \mathbb{F}^{n^2 p^2}$.

(Non-robust Degree Reduction - Attempt)

4. Let $\mathbb{S}(X) = \mathbb{S}^{(1)} + \mathbb{S}^{(2)} \cdot X + \ldots + \mathbb{S}^{(t+1)} \cdot X^t$ be vectors of degree-t polynomials. For $i = 1, \ldots, n$, locally compute PSS $\langle \mathbb{S}(i) \rangle_{d+p-1} = \langle \mathbb{S}^{(1)} \rangle_{d+p-1} + i \cdot \langle \mathbb{S}^{(2)} \rangle_{d+p-1} + \ldots + i^t \cdot \langle \mathbb{S}^{(t+1)} \rangle_{d+p-1}$.
5. For $i = 1, \ldots, n$, let $\mathbb{S}(i) = (\mathbf{S}^{(1)}(i), \ldots, \mathbf{S}^{(n)}(i))$, where each $\mathbf{S}^{(j)}(i) \in \mathbb{F}^{n^2 p^2}$. Reconstruct $\mathbb{S}(i)$ towards P_i by sending respective shares of PSS $\langle \mathbf{S}^{(1)}(i) \rangle_{d+p-1}, \ldots, \langle \mathbf{S}^{(n)}(i) \rangle_{d+p-1}, \ldots$ to P_i. Party P_i applies RS error-correction algorithm on the shares received to error-correct $\frac{t}{2}$ errors. If the error-correction does not output degree-$(d+p-1)$ polynomials, then P_i broadcasts $\mathsf{complain}(i, B_i)$ where $B_i = \{\beta |$ reconstruction failed for $\mathbf{S}^{(\beta)}(i)\}$.
6. Add party P_i to C if P_i broadcasts $\mathsf{complain}(i, B_i)$. Let $B = \cup_i B_i$ for each $P_i \in \mathsf{C}$. Moreover, for $\beta \in \{1, \ldots, n\}$, let $\mathsf{C}^{(\beta)} = \{P_i \in \mathsf{C} \mid \beta \in B_i\}$.
7. If $|\mathsf{C}^{(\beta)}| = 0$ for each $\beta \in \{1, \ldots, n\}$, then proceed to Step 11a. Otherwise, for every $P_i \in \mathsf{C}$, let β_i be the smallest value of β in B_i.

(Complaint Resolution and Fault Identification)

8. For $\gamma = 1, \ldots, t+1$, the parties do the following
   - Locally compute PSS $\langle \hat{\mathbb{Z}}^{(\gamma)} \rangle_{d+p-1} = \langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_{p-1} \times \langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_{p-1} - \langle \mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)} \rangle_{p-1} \times \langle \hat{\mathbb{B}}^{(\gamma)} \rangle_{d} - \langle \mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)} \rangle_{p-1} \times \langle \hat{\mathbb{A}}^{(\gamma)} \rangle_{d} + \langle \hat{\mathbb{C}}^{(\gamma)} \rangle_{d}$ and $\langle \hat{\mathbb{Z}}^{(\gamma)} + \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1} = \langle \hat{\mathbb{Z}}^{(\gamma)} \rangle_{d+p-1} + \langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1}$ and let $\hat{\mathbb{S}}^{(\gamma)} = \hat{\mathbb{Z}}^{(\gamma)} + \hat{\mathbb{R}}^{(\gamma)}$.
   - Define vectors of degree-t polynomials $\hat{\mathbb{S}}(X) = \hat{\mathbb{S}}^{(1)} + \hat{\mathbb{S}}^{(2)} \cdot X + \ldots + \hat{\mathbb{S}}^{(t+1)} \cdot X^t$. For $i = 1, \ldots, n$, locally compute PSS $\langle \hat{\mathbb{S}}(i) \rangle_{d+p-1} = \langle \hat{\mathbb{S}}^{(1)} \rangle_{d+p-1} + i \cdot \langle \hat{\mathbb{S}}^{(2)} \rangle_{d+p-1} + \ldots + i^t \cdot \langle \hat{\mathbb{S}}^{(t+1)} \rangle_{d+p-1}$. Let $\hat{\mathbb{S}}(i) = (\hat{\mathbf{S}}^{(1)}(i), \ldots, \hat{\mathbf{S}}^{(n)}(i))$, where each $\hat{\mathbf{S}}^{(j)}(i) \in \mathbb{F}^{n^2 p^2}$. Reconstruct $\hat{\mathbb{S}}(i)$ towards P_i by sending respective shares of PSS $\langle \hat{\mathbb{S}}^{((1))}(i) \rangle_{d+p-1}, \ldots, \langle \hat{\mathbb{S}}^{((n))}(i) \rangle_{d+p-1}$ to P_i.
9. The parties send Rand to $\mathcal{F}_{\mathsf{Rand}}$ and receives R from it. Then for each $P_i \in \mathsf{C}$, parties robustly reconstruct ${}^{A}\langle \mathbf{X}^{(\beta_i,\gamma)} + \mathbf{A}^{(\beta_i,\gamma)} \rangle_{d}$, ${}^{A}\langle \mathbf{Y}^{(\beta_i,\gamma)} + \mathbf{B}^{(\beta_i,\gamma)} \rangle_{d}$, ${}^{A}\langle \mathbf{A}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{A}}^{(\beta_i,\gamma)} \rangle_{d}$, ${}^{A}\langle \mathbf{B}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{B}}^{(\beta_i,\gamma)} \rangle_{d}$, ${}^{A}\langle \mathbf{C}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{C}}^{(\beta_i,\gamma)} \rangle_{d}$ and ${}^{A}\langle \mathbf{R}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{R}}^{(\beta_i,\gamma)} \rangle_{d}$, only towards P_i.
10. Every $P_i \in \mathsf{C}$ does the following.
    (a) Recompute PSS $\langle \mathbf{X}^{(\beta_i,\gamma)} + \mathbf{A}^{(\beta_i,\gamma)} \rangle_{d}$, $\langle \mathbf{Y}^{(\beta_i,\gamma)} + \mathbf{B}^{(\beta_i,\gamma)} \rangle_{d}$, $\langle \mathbf{A}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{A}}^{(\beta_i,\gamma)} \rangle_{d}$, $\langle \mathbf{B}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{B}}^{(\beta_i,\gamma)} \rangle_{d}$, $\langle \mathbf{C}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{C}}^{(\beta_i,\gamma)} \rangle_{d}$ and $\langle \mathbf{R}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{R}}^{(\beta_i,\gamma)} \rangle_{d}$.

(b) Compute PSS $\langle \mathbf{S}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{S}}^{(\beta_i,\gamma)} \rangle_{d+p-1} = \langle \mathbf{X}^{(\beta,\gamma)} + \mathbf{A}^{(\beta,\gamma)} \rangle_{p-1} \times \langle \mathbf{Y}^{(\beta,\gamma)} + \mathbf{B}^{(\beta,\gamma)} \rangle_{p-1} + R \cdot \langle \mathbf{X}^{(\beta,\gamma)} + \mathbf{A}^{(\beta,\gamma)} \rangle_{p-1} \times \langle \mathbf{Y}^{(\beta,\gamma)} + \mathbf{B}^{(\beta,\gamma)} \rangle_{p-1} - \langle \mathbf{X}^{(\beta,\gamma)} + \mathbf{A}^{(\beta,\gamma)} \rangle_{p-1} \times \langle \mathbf{B}^{(\beta,\gamma)} + R \cdot \hat{\mathbf{B}}^{(\beta,\gamma)} \rangle_{d} - \langle \mathbf{Y}^{(\beta,\gamma)} + \mathbf{B}^{(\beta,\gamma)} \rangle_{p-1} \times \langle \mathbf{A}^{(\beta,\gamma)} + R \cdot \hat{\mathbf{A}}^{(\beta,\gamma)} \rangle_{d} + \langle \hat{\mathbf{C}}^{(\beta,\gamma)} + R \cdot \hat{\mathbf{C}}^{(\beta,\gamma)} \rangle_{d} + \langle \mathbf{R}^{(\beta,\gamma)} + R \cdot \hat{\mathbf{R}}^{(\beta,\gamma)} \rangle_{d}$.

(c) Compute $\langle \mathbf{S}^{(\beta_i)}(i) + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1} = \langle (\mathbf{S}^{(\beta_i,1)} + R \cdot \hat{\mathbf{S}}^{(\beta_i,1)}) \rangle_{d+p-1} + i \cdot \langle (\mathbf{S}^{(\beta_i,2)} + R \cdot \hat{\mathbf{S}}^{(\beta_i,2)}) \rangle_{d+p-1} + \ldots + i^t \cdot \langle (\mathbf{S}^{(\beta_i,t+1)} + R \cdot \hat{\mathbf{S}}^{(\beta_i,t+1)}) \rangle_{d+p-1}$.

(d) For every P_j, use the shares sent by P_j during Step 5 and Step 8 to recompute the share of P_j for $\langle \mathbf{S}^{(\beta_i)}(i) + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1}$ and compare it with the shares of P_j in $\langle \mathbf{S}^{(\beta_i)}(i) + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1}$ computed in the previous step. If the shares are different then set $\mathsf{Dp}_i = \mathsf{Dp}_i \cup \{P_j\}$.

(e) Recover $\mathbf{S}^{(1)}(i), \ldots, \mathbf{S}^{(n)}(i)$ and hence $\mathbb{S}(i)$ from the shares of $\langle \mathbf{S}^{(1)}(i) \rangle_{d+p-1}$, $\ldots, \langle \mathbf{S}^{(n)}(i) \rangle_{d+p-1}$ received from the parties outside Dp_i during Step 5 by error correcting $t - |\mathsf{Dp}_i|$ errors.

(Output Computation)

11. The parties do the following.
    (a) Each P_i generates ${}^{A}\langle \mathbb{S}(i) \rangle_{p-1}$ by acting as a dealer and invoking an instance of the protocol $\Pi_{{}^{A}\langle \cdot \rangle}$ with input $\mathbb{S}(i)$.
    (b) Every party $P_i \in \mathcal{P}$ applies the RS-decoding procedure on their shares (points on the polynomials) corresponding to ${}^{A}\langle \mathbb{S}(1) \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}(n) \rangle_{p-1}$ to error-correct t errors. This local error-correction procedure allows P_i to identify at most t corrupt positions, say, w.l.o.g., they are $\{1, \ldots, t\}$. P_i now linearly combine (where the linear combination is as per Lagrange's interpolation) its shares of ${}^{A}\langle \mathbb{S}(t+1) \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}(n) \rangle_{p-1}$ to get the shares (including the MAC-tags) corresponding to ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots,$ ${}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$. It further broadcasts the corrupt set, which is $\{1, \ldots, t\}$ in this case for simplicity, to all so that every verifier P_j can adjust the verification information corresponding to the MAC-tags of P_i. The broadcast is for a n-length bit vector, where kth bit being 0 denotes P_i found P_k to be corrupt and P_k's resharing is excluded for the linear combination.
    (c) For $\gamma = 1, \ldots, t+1$, parties output ${}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_{d} = {}^{A}\langle \mathbb{S}^{(\gamma)} \rangle_{p-1} - {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d}$.

The proof of the following theorem appears in Appendix F.5.

Theorem 6.2. *Let $n = 3t + 1$. For $\gamma \in \{1, \ldots, t+1\}$, given inputs $({}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_{d}, {}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_{d}), ({}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_{d}, {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_{d}, {}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_{d}), ({}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d}, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1})$ and $({}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_{d}, {}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_{d}, {}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_{d}), ({}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d}, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1})$, where each $\mathbb{X}^{(\gamma)}$ and $\mathbb{Y}^{(\gamma)}$ consists of n^3p^2 elements from $\mathbb{F}$, protocol Π_{Beaver} generates $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_{d}\}$, where $\mathbb{Z}^{(\gamma)} = \mathbb{X}^{(\gamma)} \times \mathbb{Y}^{(\gamma)}$ component-wise iff $\mathbb{C}^{(\gamma)} = \mathbb{A}^{(\gamma)} \times \mathbb{B}^{(\gamma)}$. Moreover, the view of the adversary remains independent of $\mathbb{X}^{(\gamma)}$ and $\mathbb{Y}^{(\gamma)}$, if the remaining inputs are random for the adversary.*

After securely realizing broadcasts over point-to-point channels, the protocol requires a constant expected number of rounds and incurs a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus an expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 31 number of calls to $\mathcal{F}_{\mathsf{Rand}}$.

7 Our MPC Protocol

Figure 3 illustrates the blueprint of our MPC protocol, depicting the various building blocks (with $\mathcal{O}(1)$ communication overhead) and their inter-connection. The pre-processing phase protocol (Π_{Off}) generates APSS of vectors of random

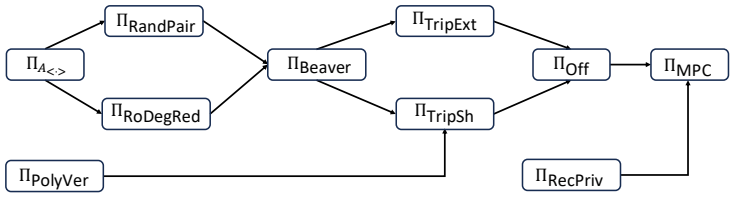


Fig. 3: Blueprint of our MPC protocol

multiplication-triples (and additional random auxiliary data), which are used in the MPC protocol Π_{MPC}. Protocol Π_{Off} is obtained by extending the generic framework of [15] for generating t-sharing of random multiplication-triples from any polynomial-based VSS to APSS for three-dimensional vectors of values. The key components of the framework are a protocol Π_{TripSh} for letting a designated dealer to verifiably generate APSS of vectors of multiplication-triples and a protocol Π_{TripExt} for "extracting" APSS of random multiplication-triples from a set of verifiably generated APSS of random and non-random vectors of multiplication-triples. Our VSS protocol (protocol $\Pi_{A\langle\cdot\rangle}$) and multiplication protocol (Π_{Beaver}) play crucial role in designing the entire framework. Parts of the blueprint has been used in some of the existing works on MPC based on PSS [27,2,45,23]. Since our goal is to achieve GOD with $\mathcal{O}(D)$ rounds, we instantiate all the underlying building blocks *robustly* with constant communication overhead for the *first* time. The complete details of the preprocessing phase and respectively of the MPC protocol are available in Appendix F and G.

Theorem 7.1. *Let $n = 3t+1$ and let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function, consisting of c_M multiplication gates and has multiplicative depth D, where f has a single input from each P_i and a single output for all n parties. Then there exists a statistically-secure MPC for computing f in an SIMD fashion $\Theta(n^7)$ times. The protocol requires $\mathcal{O}(D)$ expected rounds and incurs a communication of $\mathcal{O}(c_M n^7 + n^8)$ field elements, implying an overhead of $\mathcal{O}(1)$ field elements per gate.*[10] [11]

[10] Since f is evaluated n^7 times simultaneously, there are total $n \cdot n^7 = n^8$ input gates, $c_M n^7$ multiplication gates and $n \cdot n^7 = n^8$ output gates (since the output is supposed to be learnt by all the n parties).

[11] Note that $\mathcal{O}(1)$ field elements implies $\mathcal{O}(\kappa)$ bits of overhead, which is optimal for arithmetic circuits over large fields, specifically, when the inputs belong to $|\mathbb{F}| \geq 2^{\kappa}$.

Acknowledgment. We would like to sincerely thank Yuval Ishai for several helpful discussions and suggestions, which improved our understanding of the related work.

References

1. Abraham, I., Asharov, G., Patil, S., Patra, A.: Detect, Pack and Batch: Perfectly-Secure MPC with Linear Communication and Constant Expected Time. In: EUROCRYPT. LNCS, vol. 14005, pp. 251–281. Springer (2023)
2. Agarwal, A., Bienstock, A., Damgård, I., Escudero, D.: Honest Majority GOD MPC with $\mathcal{O}(\text{depth}(C))$ Rounds and Low Online Communication. In: ASIACRYPT. LNCS, vol. 15489, pp. 234–265. Springer (2024)
3. Asharov, G., Chandramouli, A.: Perfect (Parallel) Broadcast in Constant Expected Rounds via Statistical VSS. In: Joye, M., Leander, G. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 14655, pp. 310–339. Springer (2024)
4. Asharov, G., Lindell, Y.: A full proof of the bgw protocol for perfectly secure multiparty computation. Journal of Cryptology (2017)
5. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference (1991)
6. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-Secure MPC with Linear Communication Complexity. In: TCC. vol. 4948, pp. 213–230. Springer (2008)
7. Beerliova-Trubiniova, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3. pp. 305–328. Springer (2006)
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC (1988)
9. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure mpc with a dishonest minority. In: CRYPTO. pp. 663–680. Springer (2012)
10. Blum, E., Boyle, E., Cohen, R., Liu-Zhang, C.: Communication lower bounds for cryptographic broadcast protocols. In: DISC. LIPIcs, vol. 281, pp. 10:1–10:19 (2023)
11. Bracha, G.: Asynchronous byzantine agreement protocols. Inf. Comput. **75**(2), 130–143 (1987)
12. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: STOC (1988)
13. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science (1985)
14. Chor, B., Kushilevitz, E.: A communication-privacy tradeoff for modular addition. Inf. Process. Lett. **45**(4), 205–210 (1993)
15. Choudhury, A., Patra, A.: Optimally resilient asynchronous mpc with linear communication complexity. In: ICDCN. ICDCN '15, ACM (2015)
16. Choudhury, A., Patra, A.: An efficient framework for unconditionally secure multiparty computation. IEEE Transactions on Information Theory (2016)
17. Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.D.: Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In: CRYPTO. Lecture Notes in Computer Science, vol. 5157, pp. 241–261. Springer (2008)

18. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure mpc and the computational overhead of cryptography. In: EUROCRYPT. pp. 445–465. Springer (2010)
19. Damgård, I., Larsen, K.G., Nielsen, J.B.: Communication lower bounds for statistically secure mpc, with or without preprocessing. In: Annual International Cryptology Conference. pp. 61–84. Springer (2019)
20. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Annual International Cryptology Conference (2007)
21. Damgård, I., Nielsen, J.B., Ostrovsky, R., Rosén, A.: Unconditionally secure computation with reduced interaction. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 9666, pp. 420–447. Springer (2016)
22. Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.A.: On the communication required for unconditionally secure multiplication. In: CRYPTO. LNCS, vol. 9815, pp. 459–488. Springer (2016)
23. Damgård, I., Patil, S., Patra, A., Roy, L.: New upper and lower bounds for perfectly secure MPC. IACR Cryptol. ePrint Arch. p. 1206 (2025), https://eprint.iacr.org/2025/1206
24. Damgård, I., Schwartzbach, N.I.: Communication lower bounds for perfect maliciously secure mpc. Cryptology ePrint Archive (2020)
25. Data, D., Prabhakaran, V.M., Prabhakaran, M.M.: Communication and randomness lower bounds for secure computation. IEEE Trans. Inf. Theory **62**(7), 3901–3929 (2016)
26. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. Journal of the ACM (JACM) (1985)
27. Escudero, D., Goyal, V., Polychroniadou, A., Song, Y.: Turbopack: honest majority mpc with constant online communication. In: CCS. pp. 951–964 (2022)
28. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: STOC. pp. 554–563. ACM (1994)
29. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada. pp. 699–710. ACM (1992)
30. Genkin, D., Ishai, Y., Polychroniadou, A.: Efficient Multi-party Computation: From Passive to Active Security via Secure SIMD Circuits. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 721–741. Springer (2015)
31. Goyal, V., Polychroniadou, A., Song, Y.: Unconditional Communication-Efficient MPC via Hall's Marriage Theorem. In: Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II. pp. 275–304. Lecture Notes in Computer Science, Springer (2021)
32. Goyal, V., Polychroniadou, A., Song, Y.: Sharing Transformation and Dishonest Majority MPC with Packed Secret Sharing. In: Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. pp. 3–32. Lecture Notes in Computer Science, Springer (2022)
33. Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional MPC with guaranteed output delivery. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 85–114. Springer (2019)

34. Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional mpc with guaranteed output delivery. In: CRYPTO (2019)
35. Goyal, V., Song, Y., Zhu, C.: Guaranteed output delivery comes free in honest majority MPC. In: CRYPTO. LNCS, vol. 12171, pp. 618–646. Springer (2020)
36. Hirt, M., Maurer, U., Przydatek, B.: Efficient secure multi-party computation. In: International conference on the theory and application of cryptology and information security (2000)
37. Ishai, Y., Kushilevitz, E., Prabhakaran, M., Sahai, A., Yu, C.: Secure Protocol Transformations. In: CRYPTO. Lecture Notes in Computer Science, vol. 9815, pp. 430–458. Springer (2016)
38. Ji, X., Li, J., Song, Y.: Linear-communication asynchronous complete secret sharing with optimal resilience. In: CRYPTO. pp. 418–453. Springer (2024)
39. Kushilevitz, E.: Privacy and communication complexity. In: FOCS. pp. 416–421. IEEE Computer Society (1989)
40. Kushilevitz, E.: Privacy and communication complexity. SIAM J. Discret. Math. **5**(2), 273–284 (1992)
41. MacWilliams, F.J., Sloane, N.J.A.: The theory of error correcting codes, vol. 16. Elsevier (1977)
42. Patra, A., Choudhury, A., Rangan, C.P.: Asynchronous Byzantine Agreement with Optimal Resilience. Distributed Comput. **27**(2), 111–146 (2014)
43. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: ACM Symposium on Theory of Computing (1989)
44. Song, Y., Ye, X.: Perfectly-secure MPC with constant online communication complexity. In: TCC. LNCS, vol. 15367, pp. 329–361. Springer (2024)
45. Song, Y., Ye, X.: Perfectly-secure mpc with constant online communication complexity. Cryptology ePrint Archive (2024)

Supplementary Material

A Security

We primarily provide UC-security based proofs for the preprocessing phase (Section F.8), and the complete MPC protocol (Section G). For other sub-protocols, we *do not* model the security requirements through an ideal functionality. This is attributed to the following technicality. Our protocols use authenticated packed secret sharing , which requires a MAC-key setup among parties that is consistent throughout the MPC protocol. This setup is modeled via the functionality $\mathcal{F}_{\mathsf{MacKeys}}$ (Functionality F.2) which generates the required MAC-key setup. Parties require the setup information during VSS, as well as the preprocessing phase. Modeling them as separate functionalities requires these functionalities to share the common setup information in order to generate the authenticated packed secret sharings that are key-consistent. Such a modeling of functionalities that share common information poses a challenge. Hence, we only provide a single functionality for the entire preprocessing phase. Consequently, for each protocol including VSS, we prove security via property-based definitions and additionally show how to simulate its steps when used as a primitive in a larger protocol.

B Existing Ideal Functionalities with $n = 3t + 1$

B.1 Generation of t-sharing.

Functionality $\mathcal{F}_{[\cdot]_t}$ verifiably generates $[\cdot]_t$-sharing of dealer's inputs.

Functionality B.1: $\mathcal{F}_{[\cdot]_t}$ — The ideal functionality for generating $[\cdot]_t$

$\mathcal{F}_{[\cdot]_t}$ is parametrized by a designated dealer $D \in \mathcal{P}$ and a publicly known value N. The functionality interacts with an adversary $\mathcal{S}$ and the parties $\mathcal{P}$ as follows. Upon receiving the polynomials $q_1(\cdot), \ldots, q_N(\cdot)$ from D, send $q_1(k), \ldots, q_N(k)$ to every $P_k \in \mathcal{P}$, provided $q_1(\cdot), \ldots, q_N(\cdot)$ are t-degree polynomials over $\mathbb{F}$.

B.2 Generating a Random Value.

Functionality $\mathcal{F}_{\mathsf{Rand}}$ generates a uniformly random value for all the parties.

Functionality B.2: $\mathcal{F}_{\mathsf{Rand}}$ — Ideal functionality for a random value

$\mathcal{F}_{\mathsf{Rand}}$ proceeds as follows, interacting with the parties in $\mathcal{P}$ and an adversary $\mathcal{S}$. If (Rand, P_i) has been received from at least $t+1$ parties P_i, then pick a random value $r \in \mathbb{F}$ and send r to every party in $\mathcal{P}$.

B.3 Twisted-Sharing with Zero Shares for a Designated Sset of Parties.

Functionality $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ verifiably generates twisted t-sharing of the inputs of a designated dealer, where the shares of a designated set of parties are 0.

Functionality B.3: $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ — Ideal functionality for generating $\lceil\cdot\rfloor_t^i$ with 0 as the designated shares

The functionality is parametrized by a dealer $D \in \mathcal{P}$, a designated party P_i (which could be same as D) and a publicly known value N. The functionality interacts with an adversary $\mathcal{S}$ and the parties $\mathcal{P}$ as follows. Upon receiving a subset $\mathcal{T} \subset \mathcal{P}$ and polynomials $q_1(\cdot), \ldots, q_N(\cdot)$ from D, send $\{q_1(j), \ldots, q_N(j)\}$ to every $P_j \in \mathcal{P} \setminus \{P_i\}$, provided all the below hold.

- $|\{P_i\} \cup \mathcal{T}| \leq t + 1$.
- $q_1(\cdot), \ldots, q_N(\cdot)$ are all t-degree polynomials over $\mathbb{F}$.
- $q_1(k) = \ldots = q_N(k) = 0$ holds for every $P_k \in \mathcal{T} \setminus \{P_i\}$.
- If $P_i \in \mathcal{T}$ then $q_1(i) = \ldots = q_N(i) = 0$ holds.

Given a secure realization of $\mathcal{F}_{[\cdot]_t}$, a secure realization of $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ is straightforward. The idea is to let D first secret-share its inputs using $\mathcal{F}_{[\cdot]_t}$ and then publicly checking a random linear combination of these sharings to check if D has selected its polynomials satisfying the given constraints.

In more detail, on having input t-degree polynomials $q_1(\cdot), \ldots, q_N(\cdot)$ satisfying the constraints of $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$, the dealer D picks an additional t-degree random "masking polynomial" $q_{N+1}(\cdot)$ satisfying the same constraints and sends these $N+1$ polynomials to $\mathcal{F}_{[\cdot]_\mathsf{t}}$, for distributing the shares of these polynomials to the respective parties.[12] The parties then generate a uniformly random value r by calling $\mathcal{F}_{\mathsf{Rand}}$ and publicly reconstruct the t-degree masked polynomial $M(\cdot) \stackrel{\mathsf{def}}{=} q_{N+1}(\cdot) + rq_1(\cdot) + \ldots + r^N q_N(\cdot)$ and verify if $M(k) = 0$, for every $P_k \in \mathcal{T} \setminus \{P_i\}$ and $M(0) = 0$, if $P_i \in \mathcal{T}$.

B.4 Random Twisted t-Sharings with Designated Zero Shares

Functionality $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$ (Fig B.4) generate random twisted degree-t sharings (with respect to a designated P_i) based on the corrupted parties' shares and then distribute the honest parties' shares to them, such that the twisted-shares of a designated subset of parties of size up to $t+1$ are 0. Moreover, the underlying secrets of the generated sharings will be embedded at point $-\mathsf{position}$, where $\mathsf{position} \in \{1, \ldots, p+1\}$.

Functionality B.4: $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t^i}$ – Generating N random twisted t-sharings where secrets are stored at a designated evaluation point

The functionality parametrized by a public value $N \geq 1$ proceeds as follows, interacting with the parties in $\mathcal{P}$ and an adversary $\mathcal{S}$. Upon receiving $(\mathsf{RandShare}, \mathcal{T}, P_i, \mathsf{position})$ from the honest parties where $|\mathcal{T}| \leq t$ and $\mathsf{position} \in \{1, \ldots, p+1\}$, the functionality randomly selects $r_1, \ldots, r_N \in \mathbb{F}$ and does the following:

- Receive a set of shares for the corrupt parties from $\mathcal{S}$ and then sample random degree-t twisted sharings $\lceil r_1||\mathsf{position}\rfloor_t^i, \ldots, \lceil r_N||\mathsf{position}\rfloor_t^i$, based on the shares of the corrupt parties and the secrets $r_1, \ldots, r_N$ respectively, such that the shares of the parties in $\mathcal{T}$ are 0.
- To every honest P_j, send P_j's shares of $\lceil r_1||\mathsf{position}\rfloor_t^i, \ldots, \lceil r_N||\mathsf{position}\rfloor_t^i$.

C Information-Checking Protocol with $n = 3t + 1$

Here we prove the properties of our ICP; $\mathcal{H}$ denotes the set of honest parties. For $n = 3t+1$, the condition $|\mathcal{H}| \geq 2t+1$ holds.

Lemma C.1 (Correctness). *Let $S, I \in \mathcal{H}$ and let S has input $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)} \in \mathbb{F}^\ell$ for the protocol* Gen. *Then I sets $V_{S,I}$ to 1 during the protocol* Auth. *Moreover, every $P_j \in \mathcal{H}$ outputs $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ during* Rev.

[12] As per the semantics of $\mathcal{F}_{[\cdot]_\mathsf{t}}$, the share of P_i is the i^{th} points on the polynomials received by $\mathcal{F}_{[\cdot]_\mathsf{t}}$. However, since in $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ we are generating twisted-sharings with respect to P_i, the share of P_i will be the 0^{th} point on D's polynomials. Consequently, upon receiving $q_1(\cdot), \ldots, q_{N+1}(\cdot)$, instead of the i^{th} point on these polynomials, the functionality $\mathcal{F}_{[\cdot]_\mathsf{t}}$ distributes the 0^{th} point on these polynomials to P_i as its share.

Proof. During Gen, the conditions $r_i = R(\alpha_i)$ and $v_{m,i} = f^{(m)}(\alpha_i)$ will hold for every $P_i \in \mathcal{P}$. Consequently, $M(\alpha_i) = r_i + dv_{1,i} + \ldots + d^M v_{M,i}$ holds for every $P_i \in \mathcal{P}$. Hence every $P_i \in \mathcal{H}$ will broadcast $\texttt{Accept}_i$ during Auth and hence $\mathcal{H} \subseteq \mathcal{R}$. As $|\mathcal{H}| \geq n-t$, this implies that I will set $V_{S,I}$ to 1. During Rev, intermediary I will broadcast $f^{(1)}(X), \ldots, f^{(M)}(X)$ and each $P_i \in \mathcal{H}$ will broadcast $\texttt{Accept}_i$. Since $|\mathcal{H}| \geq n-t$, it follows that each P_j will output the higher order ℓ coefficients of $f^{(1)}, \ldots, f^{(M)}(X)$, namely $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ respectively. □

Lemma C.2 (Privacy). *Let $S, I \in \mathcal{H}$ and let S has input $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$. Then adversary's view remains independent of $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ during* Gen *and* Auth.

Proof. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of the adversary. During Gen, corresponding to each $P_i \in \{P_1, \ldots, P_t\}$, the adversary learns α_i and the value of the polynomials $R(X), f^{((1))}, \ldots, f^{(M)}(X)$ at α_i. As each of these polynomials is a random $(\ell + t - 1)$-degree polynomial and adversary learns at most t distinct points on each of these polynomials, the view of the adversary during Gen is independent of the higher order ℓ coefficients of $R(X), f^{(1)}(X), \ldots, f^{(M)}(X)$, selected by D. Since the polynomials $R(X)$ and $f^{(m)}(X)$ are randomly chosen by D, it follows that the view of the adversary during protocol Gen remains independent of $\mathbf{r}$ and $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$.

During Auth, adversary learns d and $M(X) = R(X) + df^{(1)}(X) + \ldots + d^M f^{(M)}(X)$. Since the adversary's view during Gen is independent of the higher order ℓ coefficients of $R(X)$ and $f^{(1)}(X), \ldots, f^{(M)}(X)$, it follows that even after learning $M(X)$, the adversary's view remains independent of the higher order ℓ coefficients of $f^{(1)}(X), \ldots, f^{(M)}(X)$. Namely, for every candidate $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)} \in \mathbb{F}^\ell$, there is a corresponding unique $\mathbf{r} \in \mathbb{F}^\ell$, such that all the following hold:

- For $m = 1, \ldots, M$, $f^{(m)}(\alpha_i) = v_{m,i}$, for every $P_i \in \{P_1, \ldots, P_t\}$.
- $R(\alpha_i) = r_i$, for every $P_i \in \{P_1, \ldots, P_t\}$.
- The higher order ℓ coefficients of $f^{(m)}(X)$ are the elements of $\mathbf{s}^m$.
- The higher order ℓ coefficients of $R(X)$ are elements of $\mathbf{r}$.
- The condition $M(X) = R(X) + df^{(1)}(X) + \ldots + d^M f^{(M)}(X)$ holds.

Since $R(X)$ is selected randomly by D, it holds that the view of the adversary at the end of Auth remains independent of $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$. □

Lemma C.3 (Non-repudiation). *Let S be corrupt and let $I \in \mathcal{H}$. If I sets $V_{S,I}$ to 1 and holds $\mathsf{Sig}(S \to I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$ at the end of* Auth*, then except with probability $2^{-\Omega(\kappa)}$, every party $P_j \in \mathcal{H}$ outputs $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ during* Rev.

Proof. Let S be corrupt and let $I \in \mathcal{H}$ such that I has set $V_{S,I}$ to 1 during protocol Auth. This further implies that there is a set $\mathcal{R}$ of size at least $n - t$, such that every $P_i \in \mathcal{R}$ has broadcasted $\texttt{Accept}_i$ during Auth, in response to the polynomial $M(X)$ broadcasted by I. Let $\mathcal{H}_\mathcal{R}$ be the set of honest parties in $\mathcal{R}$, where $|\mathcal{H}_\mathcal{R}| \geq n - 2t$. Moreover, let $R(X)$ and $f^{(1)}(X), \ldots, f^{(M)}(X)$ be the $(\ell + t - 1)$-degree polynomials received by I from S during Gen. We claim that for every $P_i \in \mathcal{H}_\mathcal{R}$, the condition $r_i = R(\alpha_i)$ and the condition $f^{(m)}(\alpha_i) = v_{m,i}$ holds, for $m = 1, \ldots, M$, except with a negligible probability. Here α_i and

$r_i, v_{1,i}, \ldots, v_{M,i}$ are received by P_i from S during Gen. If the claim is true, then it follows that every $P_i \in \mathcal{H}_{\mathcal{R}}$ will again broadcast $\texttt{Accept}_i$ during Rev. This is because I will set $\mathsf{Sig}(S \to I, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$ to $f^{(1)}(X), \ldots, f^{(M)}(X)$ during Auth and will broadcast the polynomials $f^{(1)}(X), \ldots, f^{(M)}(X)$ during Rev.

We prove the claim through a contradiction. Let there exist some $P_i \in \mathcal{H}_{\mathcal{R}}$, such that D distributes points to P_i which are inconsistent with respect to the polynomials distributed to I. Namely, α_i, r_i and $v_{1,i}, \ldots, v_{M,i}$ received by P_i satisfy the following conditions.

- $r_i = R(\alpha_i) + \delta_i$
- $v_{m,i} = f^{(m)}(\alpha_i) + \Delta_{m,i}$, for $m = 1, \ldots, M$.

Here $\delta_i, \Delta_{1,i}, \ldots, \Delta_{M,i} \in \mathbb{F}$. Moreover, at least two of these values are not zero.[13] We now compute the probability with which $M(\alpha_i) = r_i + dv_{1,i} + \ldots + d^M v_{M,i}$ holds, under the above conditions. Substituting the values of $M(X)$ and $r_i, v_{1,i}, \ldots, v_{M,i}$, if $M(\alpha_i) = r_i + dv_{1,i} + \ldots + d^M v_{M,i}$, then the following holds:

$$R(\alpha_i) + df^{(1)}(\alpha_i) + \ldots + d^M f^{(M)}(\alpha_i) = (R(\alpha_i) + \delta_i) + d(f^{(1)}(\alpha_i) + \Delta_{1,i}) + \\ \ldots + d^M(f^{(M)}(\alpha_i) + \Delta_{M,i}).$$

By rearranging the terms we get that the following holds:

$$\delta_i + d\Delta_{1,i} + \ldots + d^M \Delta_{M,i} = 0.$$

This further implies that d should be the root of the polynomial $D(X) \stackrel{\text{def}}{=} \delta_i + \Delta_{1,i}X + \ldots + \Delta_{M,i}X^M$. Notice that the polynomial $D(X)$ is a non-zero polynomial, with at least two of its coefficients being non-zero. We also note that S will not know the value of d beforehand while distributing the polynomials $R(X), f^{(1)}(X), \ldots, f^{(M)}(X)$ to I and $\alpha_i, r_i, v_{1,i}, \ldots, v_{M,i}$ to P_i during Gen, since d is selected during Auth. Moreover, I selects d uniformly at random from $\mathbb{F}$. So the probability that a randomly chosen d turns out to be the root of the polynomial $D(X)$ is at most $\frac{M}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$. Hence, if S distributed inconsistent information to I and P_i during Gen, then except with probability $2^{-\Omega(\kappa)}$, P_i will not broadcast $\texttt{Accept}_i$ during Auth, which is a contradiction. □

Lemma C.4 (Unforgeability). *Let S be honest and have input $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)}$ for Gen. If during Rev, I reveals $\mathsf{Sig}(S \to I, \overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)})$ and if the parties in $\mathcal{H}$ output $\overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)}$, then except with probability $2^{-\Omega(\kappa)}$, each $\overline{\mathbf{s}}^{(m)} = \mathbf{s}^{(m)}$.*

Proof. Let $S \in \mathcal{H}$. If $I \in \mathcal{H}$, then the lemma follows trivially Lemma C.1. So consider the case where I is corrupt.

Let I reveal $\mathsf{Sig}(S \to I, \overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)})$ during Rev by broadcasting $(\ell + t - 1)$-degree polynomials $\overline{f}^{(1)}(X), \ldots, \overline{f}^{(M)}(X)$, where for $m = 1, \ldots, M$, the higher order ℓ coefficients of $\overline{f}^{(m)}(X)$ are the elements of $\overline{\mathbf{s}}^{(m)}$. Moreover, there exist

[13] If exactly one of the values $\delta_i, \Delta_{1,i}, \ldots, \Delta_{M,i}$ is non-zero, then $M(\alpha_i) \neq r_i + dv_{1,i} + \ldots + d^M v_{M,i}$ will hold and hence P_i will never broadcast $\texttt{Accept}_i$ during Auth.

some $m \in \{1, \ldots, M\}$, such that $\mathbf{s}^{(m)} \neq \overline{\mathbf{s}}^{(m)}$, further implying that $f^{(m)}(X) \neq \overline{f}^{(m)}(X)$. We claim that in this case, except with probability at most $\frac{n}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$, no party $P_i \in \mathcal{H}$ will broadcast $\texttt{Accept}_i$ during Rev. This will further imply that no party in $\mathcal{H}$ will output $\overline{\mathbf{s}}^{(1)}, \ldots, \overline{\mathbf{s}}^{(M)}$ during Rev.

Consider an arbitrary $P_i \in \mathcal{H}$. We argue that throughout Gen and Auth, the adversary does not learn any additional information about α_i held by P_i. This is trivially true for Gen. Similarly, during Auth, intermediary I broadcasts $M(X)$ and every $P_i \in \mathcal{H}$ may broadcast $\texttt{Accept}_i$, again revealing no additional information about α_i. Now consider an $m \in \{1, \ldots, M\}$, such that I broadcasts $\overline{f}^{(m)}(X)$ during Rev where $\overline{f}^{(m)}(X) \neq f^{(m)}(X)$. In this case, P_i will broadcast $\texttt{Accept}_i$ during Rev only if $\overline{f}^{(m)}(\alpha_i) = v_{m,i}$ holds. However, since $S \in \mathcal{H}$, it follows that $v_{m,i} = f^{(m)}(\alpha_i)$ holds. This further implies that $\overline{f}^{(m)}(\alpha_i) = f_m(\alpha_i)$ holds. However, α_i is randomly chosen by S and not known to the adversary. So to ensure that $\overline{f}^{(m)}(\alpha_i) = f_m(\alpha_i)$ holds, the adversary has to correctly guess α_i, which it can do with probability $\frac{1}{|\mathbb{F}|}$. Since we considered an arbitrary $P_i \in \mathcal{H}$ and since $|\mathcal{H}| \leq n$, it follows from the union bound that the probability that any party in $\mathcal{H}$ broadcasts $\texttt{Accept}_i$ during Rev is at most $\frac{n}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$. □

Lemma C.5 (Complexity). *The following hold for Protocol 4.2.*

- Gen *requires one round and a communication of* $\mathsf{P2P}((\ell M + nM)\kappa)$ *bits.*
- Auth *requires two rounds and a communication of* $\mathsf{BC}((\ell + n)\kappa) + n \times \mathsf{BC}(1)$ *bits.*
- Rev *requires two rounds and a communication of* $\mathsf{BC}((\ell M + nM)\kappa) + n \times \mathsf{BC}(1)$ *bits.*

Proof. The round complexity of the protocols follow easily.

During Gen, the sender has to distribute $M+1$ polynomials to I over $\mathbb{F}$, each of degree $(\ell + t - 1)$. In addition, each verifier P_i receives a random evaluation-point over $\mathbb{F}$ and the value of the polynomials at this evaluation point. During Auth, the intermediary I has to broadcast a linear combination of the $M + 1$ polynomials received from S, where the linearly combined polynomial is of degree $(\ell + t - 1)$. In response, each party has to broadcast a single bit in the form of an $\texttt{Accept}$ message. Finally, in the protocol Rev, the intermediary I has to broadcast M number of $(\ell+t-1)$-degree polynomials that it received from S and in response, each party may broadcast a bit in the form of an $\texttt{Accept}$ message. The communication complexity of the various protocols now follow from the fact that each element from $\mathbb{F}$ can be represented by κ bits. □

C.1 Linearity

Consider a *fixed* S and I. Let $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(N)}$ be vectors, where for $k = 1, \ldots, N$, we have $\mathbf{S}^{(k)} = (\mathbf{s}^{(k,1)}, \ldots, \mathbf{s}^{(k,M)})$, with each $\mathbf{s}^{(k,1)}, \ldots, \mathbf{s}^{(k,M)} \in \mathbb{F}^{\ell}$. Hence each $\mathbf{S}^{(k)}$ consists of ℓM elements. Let I receive $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$ from S (see Notation 4.3). Moreover, for $k = 1, \ldots, N$, let $\mathsf{Gen}^{(k)}$ and $\mathsf{Auth}^{(k)}$

be the corresponding instances of Gen and Auth respectively, which are used to generate $\mathsf{Sig}(S \to I, \mathbf{S}^{(k)})$, such that all the following conditions are satisfied.

- Corresponding to every $P_i \in \mathcal{P}$, sender S has used the same evaluation-point α_i during the instances $\mathsf{Gen}^{(1)}, \ldots, \mathsf{Gen}^{(N)}$.
- Let $\mathcal{R}_1, \ldots, \mathcal{R}_N$ be the instances of $\mathcal{R}$, computed by I during the instances $\mathsf{Auth}^{(1)}, \ldots, \mathsf{Auth}^{(N)}$ respectively, where each $|\mathcal{R}_k| \geq n - t$, for $k = 1, \ldots, N$. Then $|\mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N| \geq n - t$ holds.

Let $\mathbf{S} \stackrel{\mathsf{def}}{=} c_1\mathbf{S}^{(1)} + \ldots + c_N\mathbf{S}^{(N)}$, where $c_1, \ldots, c_N \in \mathbb{F}$ and known publicly. If the above conditions are satisfied, then I can locally compute $\mathsf{Sig}(S \to I, \mathbf{S})$ from $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$.

In more detail, let $\mathsf{Sig}(S \to I, \mathbf{S}^{(k)}) = (f^{(k,1)}(X), \ldots, f^{(k,M)}(X))$, where each of these polynomials $f^{(k,1)(X)}, \ldots, f^{(k,M)}(X)$ is a $(\ell + t - 1)$-degree polynomial, with the higher order ℓ coefficients being elements of $\mathbf{S}^{(k)}$, namely $\mathbf{s}^{(k,1)}, \ldots, \mathbf{s}^{(k,M)}$ respectively. Let $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(M)})$, where for $m = 1, \ldots, M$, we have $\mathbf{s}^{(m)} = c_1\mathbf{s}^{(1,m)} + \ldots + c_N\mathbf{s}^{(N,m)}$. Then consider the $(\ell+t-1)$-degree polynomial $f^{(m)}(X) \stackrel{\mathsf{def}}{=} c_1 f^{(1,m)}(X) + \ldots + c_N f^{(N,m)}(X)$. It follows that the higher order ℓ coefficients of $f^{(m)}(X)$ are the elements of $\mathbf{s}^{(m)}$. Hence I can locally compute $f^{(1)}(X), \ldots, f^{(M)}(X)$ and set $\mathsf{Sig}(S \to I, \mathbf{S}) = (f^{(1)}(X), \ldots, f^{(M)}(X))$. For a pictorial illustration, see part (a) of Fig 4.

Similarly, every party in $\mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N$ can locally compute its verification-tags corresponding to $\mathsf{Sig}(S \to I, \mathbf{S})$, from its verification-tags corresponding to $\mathsf{Sig}(S \to I, \mathbf{S}^{(1)}), \ldots, \mathsf{Sig}(S \to I, \mathbf{S}^{(N)})$. In more detail, let $\mathcal{R}_{\mathsf{com}} \stackrel{\mathsf{def}}{=} \mathcal{R}_1 \cap \ldots \cap \mathcal{R}_N$. Then every $P_i \in \mathcal{R}_{\mathsf{com}}$ has a common α_i across all the N instances of Gen and Auth. Moreover, during $\mathsf{Gen}^{(k)}$, party P_i would receive the verification-tags, say $v_{1,i}^{(k)}, \ldots, v_{M,i}^{(k)}$ from S. Then P_i can locally compute $v_{m,i} \stackrel{\mathsf{def}}{=} c_1 v_{m,i}^{(1)} + \ldots + c_N v_{m,i}^{(N)}$ and set $v_{1,i}, \ldots, v_{M,i}$ as the verification-tags corresponding to $\mathsf{Sig}(S \to I, \mathbf{S})$. For a pictorial illustration, see part (b) of Fig 4.

Later, if I is asked to reveal $\mathsf{Sig}(S \to I, \mathbf{S})$, then it can invoke Rev to reveal $f^{(1)}, \ldots, f^{(M)}(X)$, defined as above. And each $P_i \in \mathcal{R}_{\mathsf{com}}$ can verify these polynomials with respect to α_i and the verification-tags $v_{1,i}, \ldots, v_{M,i}$.

If S and I are honest, then the correctness property of the instances $(\mathsf{Gen}^{(1)}, \mathsf{Auth}^{(1)}), \ldots, (\mathsf{Gen}^{(N)}, \mathsf{Auth}^{(N)})$ will guarantee that every honest party will accept $\mathbf{S}$. This is because there will be at least $n - 2t$ honest parties in $\mathcal{R}_{\mathsf{com}}$. And for every honest $P_i \in \mathcal{R}_{\mathsf{com}}$, the condition $f^{(m)}(\alpha_i) = v_{m,i}$ will hold, for $m = 1, \ldots, M$. Consequently, at least $n - 2t$ parties will broadcast `Accept` in Rev.

On the other hand, even if S is corrupt and I is honest, all honest parties will accept $\mathbf{S}$. This is because the non-repudiation property of the instances $(\mathsf{Gen}^{(1)}, \mathsf{Auth}^{(1)}), \ldots, (\mathsf{Gen}^{(N)}, \mathsf{Auth}^{(N)})$ would ensure that the verification-tags received by every honest $P_i \in \mathcal{R}_{\mathsf{com}}$ are consistent with the polynomials held by I. Namely, if I holds the polynomials $f^{(k,1)}(X), \ldots, f^{(k,M)}(X)$ and P_i holds the verification-tags $v_{1,i}^{(k)}, \ldots, v_{M,i}^{(k)}$ from $\mathsf{Gen}^{(k)}$, then $\mathsf{Auth}^{(k)}$ would guarantee that except with probability $2^{-\Omega(\kappa)}$, the condition $v_{m,i}^{(k)} = f^{(k,m)}(\alpha_i)$ holds, for

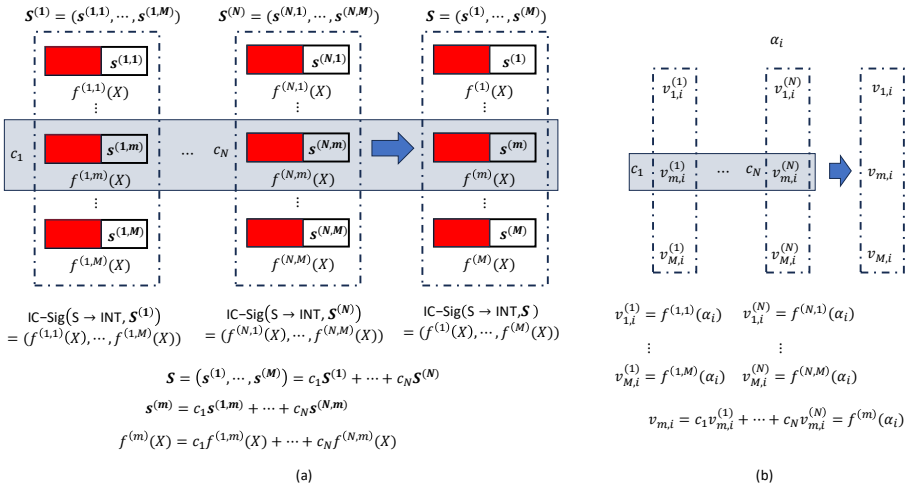


Fig. 4: Pictorial illustration of the linearity of ICP. The information held by I is depicted in part (a), while part (b) depicts the verification-tags held by each P_i.

$m = 1, \ldots, M$. This will further imply that the condition $v_{m,i} = f^{(m)}(\alpha_i)$ will also hold and so every honest $P_i \in \mathcal{R}_{\mathsf{com}}$ will broadcast $\mathtt{Accept}_i$ during Rev.

D Properties of Our VSS

Here we discuss the various properties of protocol $\Pi_{A\langle\cdot,\cdot\rangle}$.

D.1 Proofs of Theorem 5.2 for $\Pi_{A\langle\cdot,\cdot\rangle}$

As mentioned in Section A, we *do not* model the security requirements of $\Pi_{A\langle\cdot,\cdot\rangle}$ through an ideal functionality. Instead, we prove the standard VSS properties of privacy, correctness and commitment. Looking ahead, we will show how to simulate the steps of $\Pi_{A\langle\cdot,\cdot\rangle}$ in the protocols where $\Pi_{A\langle\cdot,\cdot\rangle}$ is used as a primitive. While proving the properties of $\Pi_{A\langle\cdot,\cdot\rangle}$ it will be assumed that the parties will have a prior setup of consistent MAC-keys. Namely, for every pair of parties (P_i, P_j), there will be a $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{Mp}$ held by P_j, such that the parties hold $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$, where the twisted-shares of P_i are 0. Moreover, if P_j is *honest* and P_i is *corrupt* then $\boldsymbol{\mu}_{ji}$ remains random for the adversary.

Claim D.1. *An honest D is not discarded, except with a negligible probability.*

Proof. During the sharing phase, an honest D will not be discarded, since it will broadcast a valid Core set, either during the first run of the protocol or during the second run of the protocol. This is because if D is unable to find a Core of size $2t + p$ during the first run of the protocol, then it will include at

least $t-p+2$ *corrupt* parties in Dp. Subsequently, when the parties restart the protocol, at least $2t+1$ honest parties will be included in Core and consequently $|\mathsf{Core} \cup \mathsf{Dp}| \geq (2t+1)+t-p+2 \geq 2t+p$ will be satisfied, as $p = \frac{t}{4}+1$.

For the rest of the phases, D gets discarded if it fails to reveal the IC-signature of any party from Core. An honest D will always be able to reveal IC-signature of every honest party from Core, which follows from the correctness of ICP (Lemma C.1). On the other hand hand, the non-repudiation property of ICP (Lemma C.3) guarantees that except with a negligible probability, an *honest* D will be able to reveal the IC-signature of every potential corrupt party in Core. As there are polynomially many instances of ICP, it follows that D will not be discarded due to this condition, except with a negligible probability. □

Claim D.2. *If D is honest and $\mathsf{Cr} \neq \emptyset$, then every party in Cr is corrupt.*

Proof. Consider an arbitrary $P_l \in \mathsf{Cr}$. We want to show that P_l is *corrupt*. On contrary, let P_l be honest. Consider an arbitrary pair of parties (P_i, P_j), such that P_i broadcasts $(\mathsf{NOK}, \mathsf{Dp}_i)$ in the protocol where $P_l \in \mathsf{Dp}_i$. If P_l is honest, it correctly computes and sends the tag-share $\tau_{ij,l}^{(u)}$ to P_i, for every $u \in \{1, \ldots, L+4\}$, where the following holds: $\tau_{ij,l}^{(u)} = \mathbf{s}_l^{(u)} \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}^{(u)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(u,k)}$, where $r_{ji,l}^{(u,k)}$ denotes P_l's share of $\lceil r_{ji}^{(u,k)} || k \rfloor_t^i$. We will show that if this is the case then P_l will not be included to Cr, which is a contradiction. Since D is *honest*, it implies that for each $v \in \{1, \ldots, M\}$, the following holds:

$$\mathsf{F}^{(3,v)} = F^{(1,v)}(X,Y) + r_3 \cdot F^{(2,v)}(X,Y) + \ldots + r_3^{L+3} \cdot F^{(L+4,v)}(X,Y).$$

This further implies that $\mathbf{s}_l = \mathbf{s}_l^{(1)} + r_3 \cdot \mathbf{s}_l^{(2)} + \ldots + r_3^{L+3} \cdot \mathbf{s}_l^{(L+4)}$ holds. Now consider the linear combination of the tags: $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$, which will be the same as

$$\mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l} + \Bigg((v_{ji,l}^{(1)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(1,k)}) + r_3 \cdot (v_{ji,l}^{(2)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(2,k)}) + \\ \ldots + r_3^{L+3} \cdot (v_{ji,l}^{(L+4)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(L+4,k)}) \Bigg)$$

Now consider the twisted t-sharing:

$$\lceil \beta_{ij,l} \rfloor_t^i = \lceil \mathbf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \left(\lceil v_{ji}^{(1)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(1,k)} || k \rfloor_t^i \right) \\ + r_3 \cdot \left(\lceil v_{ji}^{(2)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(2,k)} || k \rfloor_t^i \right) + \ldots \\ + r_3^{L+3} \cdot \left(\lceil v_{ji}^{(L+4)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(L+4,k)} || k \rfloor_t^i \right) + \lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$$

It is easy to see that P_l's share in $\lceil \mathbf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ will be $\mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l}$. We also know that P_l's share in $\lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$ will be 0, as $\lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$ is generated by D, who sets the share of P_l to 0. It then turns out that the share of P_l in the rest of twisted t-sharing above will be the same as $(v_{ji,l}^{(1)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(1,k)}) + r_3 \cdot (v_{ji,l}^{(2)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(2,k)}) + \ldots + r_3^{L+3} \cdot (v_{ji,l}^{(L+4)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(L+4,k)})$. Consequently, the share of P_l in $\lceil \beta_{ij,l} \rfloor_t^i$ will be the same as $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$. From the protocol steps, P_i will robustly reconstruct the entire sharing $\lceil \beta_{ij,l} \rfloor_t^i$ and find that the share of P_l is the same as $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$. Now there are two possible cases, depending upon whether P_i is *honest* or *corrupt*.

An honest P_i will *not* include P_l to Dp_i, which is a contradiction. On the other hand, if P_i is *corrupt*, then it may include P_l to Dp_i. Then consider the tag-share complaint resolution phase. By using similar arguments as above, it can be shown that P_l's share in $\lceil \gamma_{ij,l} \rfloor_t^i$ will be the same as $\tau_{ij,l}^{(1)} + r_4 \cdot \tau_{ij,l}^{(2)} + \ldots + r_4^{L+3} \cdot \tau_{ij,l}^{(L+4)}$ and P_l will not be included to Cr, which is a contradiction. □

Lemma D.3 (Privacy). *If D is honest then adversary's view remains independent of D's input $\mathbb{S}$.*

Proof. We show that throughout adversary may learn at most t row and column polynomials and hence $t(t+p) + tp$ distinct points on each bivariate polynomial $F^{(u,v)}(X,Y)$. This leaves p^2 degree-of-freedom in $F^{(u,v)}(X,Y)$ and consequently, adversary will not learn the elements of $\mathbb{S}$ which are embedded in $F^{(u,v)}(X,Y)$. That is, for every candidate $\mathbf{s}^{(u,(v-1)\cdot p+1)}, \ldots, \mathbf{s}^{(u,v\cdot p)} \in \mathbb{F}^p$, there exists a unique degree-(d,d) bivariate polynomial $F^{(u,v)}(X,Y)$, embedding the elements of $\mathbf{s}^{(u,(v-1)\cdot p+1)}, \ldots, \mathbf{s}^{(u,v\cdot p)}$ at $F^{(u,v)}(X,-1), \ldots, F^{(u,v)}(X,-p)$ respectively, which will be consistent with the row and column polynomials learned by the adversary for $F^{(u,v)}(X,Y)$. Since $F^{(u,v)}(X,Y)$ is chosen randomly by D, it follows that adversary's view remains independent of $\mathbf{s}^{(u,(v-1)\cdot p+1)}, \ldots, \mathbf{s}^{(u,v\cdot p)}$.

We next compute the amount of information for each bivariate polynomial $F^{(u,v)}(X,Y)$ learned by the adversary during various steps of the protocol.

- **Sharing phase**: The adversary receives at most t row and column polynomials for $F^{(u,v)}(X,Y)$. Moreover, the row and column polynomials corresponding to the parties in $\mathsf{Dp} \cup \mathsf{Cr}$ will be $\mathbf{0}$. From Claim D.2 and from the proof of Claim D.1, if D is honest then the parties in $\mathsf{Dp} \cup \mathsf{Cr}$ are corrupt and consequently $|\mathsf{Dp} \cup \mathsf{Cr}| \leq t$. Next, consider an arbitrary *honest* P_i. From the privacy property of ICP (Lemma C.2), adversary does not learn any additional information about $\mathbf{g}_i^{(u,v)}$ during the instances of ICP invoked by P_i. So overall in this phase, the adversary learns at most t row and column polynomials on $F^{(u,v)}(X,Y)$. This is true even if parties restart the protocol, since D picks independent bivariate polynomials during the rerun.
- **Publicly revealing linear combinations of bivariate polynomials**: In the worst case, D makes public the linear combinations $\mathsf{F}^{(1,v)}(X,Y), \mathsf{F}^{(2,v)}(X,Y), \mathsf{F}^{(3,v)}(X,Y)$ and $\mathsf{F}^{(4,v)}(X,Y)$ of the bivariate polynomials $F^{(1,v)}(X,Y), \ldots, F^{(L+4,v)}(X,Y)$ during the remaining phases of the protocol. Let $U_1 =$

$\{1,\ldots,L\}$ and $U_2 = \{L+1,\ldots,L+4\}$. As shown above, during the sharing phase, for *every* $F^{(u,v)}(X,Y)$ where $u \in U_1 \cup U_2$ adversary learned at most $t(t+p)+tp$ distinct points and its view remains independent of the vectors $\mathbf{s}^{(u,(v-1)\cdot p+1)},\ldots,\mathbf{s}^{(u,v\cdot p)} \in \mathbb{F}^p$ embedded in this bivariate polynomial. It then follows that even after learning $\mathsf{F}^{(1,v)}(X,Y), \mathsf{F}^{(2,v)}(X,Y), \mathsf{F}^{(3,v)}(X,Y)$ and $\mathsf{F}^{(4,v)}(X,Y)$, adversary does not learn any additional information about the bivariates $F^{(u,v)}(X,Y)$ where $u \in U_1$. That is from adversary's point of view, for every candidate vectors $\mathbf{s}^{(u,(v-1)\cdot p+1)},\ldots,\mathbf{s}^{(u,v\cdot p)} \in \mathbb{F}^p$ embedded in $F^{(u,v)}(X,Y)$ where $u \in U_1$, there exist unique masking vectors $\mathbf{s}^{(u,(v-1)\cdot p+1)},\ldots,\mathbf{s}^{(u,v\cdot p)} \in \mathbb{F}^p$ embedded in $F^{(u,v)}(X,Y)$ where $u \in U_2$ and unique bivariate polynomials $F^{(u,v)}(X,Y)$ where $u \in U_1 \cup U_2$, which will be consistent with $\mathsf{F}^{(1,v)}(X,Y), \mathsf{F}^{(2,v)}(X,Y), \mathsf{F}^{(3,v)}(X,Y), \mathsf{F}^{(4,v)}(X,Y)$ and with the row and column polynomials learned by the adversary throughout. Since for every $F^{(u,v)}(X,Y)$ where $u \in U_2$, the underlying embedded vectors $\mathbf{s}^{(u,(v-1)\cdot p+1)},\ldots,\mathbf{s}^{(u,v\cdot p)} \in \mathbb{F}^p$ are random, as they are part of the random masking-vectors $\mathbf{S}^{(u)}$, it follows that adversary's view still consists of t rows and columns on $F^{(u,v)}(X,Y)$ for every $u \in U_1$.
Note that while making public the linear combinations $\mathsf{F}^{(1,v)}(X,Y), \mathsf{F}^{(2,v)}(X,Y), \mathsf{F}^{(3,v)}(X,Y)$ and $\mathsf{F}^{(4,v)}(X,Y)$, the dealer also reveals the IC-signature on the points of the column polynomials of $\mathsf{F}^{(1,v)}(X,Y), \mathsf{F}^{(2,v)}(X,Y), \mathsf{F}^{(3,v)}(X,Y)$ and $\mathsf{F}^{(4,v)}(X,Y)$, which are already in adversary's view.

- **Tag Generation Phase**: In the setup stage of the this phase, parties generate random t-sharings. For each of these t-sharings, adversary may learn at most t shares and hence the underlying shared values remain random for the adversary. While computing the tags, each party learns a degree-$(d+t)$ sharing of each its supposed MAC-tag. The degree-$(d+t)$ sharing $\lceil \mathbf{s}_i^{(u)} \rfloor_d^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \lceil v_{ji}^{(u)} \rfloor_t^i$ is masked by the random degree-$(d+t)$ polynomial corresponding to $\sum_{k=1}^{d} [[\mathbf{e}_k]]_d^i \cdot \lceil r_{ji}^{(u,k)} || k \rfloor_t^i$ thus ensuring the view of the adversary remains independent of the inputs. While verifying the tag-shares and resolving complaints, the randomness of mask and $\mathtt{mask}$ prevents the adversary from learning any additional information about the bivariates.

Thus, in each phase of $\Pi_{A\langle\cdot\rangle}$, adversarys view is independent of $\mathbb{S}$. □

Lemma D.4. *The parties need to run* $\Pi_{A\langle\cdot\rangle}$ *at most* 3 *times to get an output.*

Proof. The parties need to restart, either due to the failure of tag-generation phase or due to the failure of sharing phase. Suppose the parties restart due to the failure of tag-generation phase. In this case, the parties include at least $t-p+1$ *corrupt* parties in Cr. Note that $|\mathcal{H}| \geq 2t+1$, where $\mathcal{H}$ is the set of honest parties. After restart, during the tag-generation phase, every $P_i \in \mathcal{H}$ is bound to get an $\mathcal{R}_{ij}$ set for every $P_j \in \mathcal{P}$ such that $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \geq 2t+p$ holds. This is because the set $\mathcal{H}$ always constitutes such a candidate $\mathcal{R}_{ij}$ set, as $p = \frac{t}{4}+1$. Consequently, if P_i complaints about not having an $\mathcal{R}_{ij}$ set satisfying the above condition, the parties will ignore the complaint and do not proceed with the complaint resolution, since P_i is guaranteed to be corrupt in this case.

Let us now consider the case when the parties restart due to the failure of sharing phase. In this case, the parties outside Core will be included in the set Dp, where $|\mathsf{Dp}| \geq t-p+1$ and the parties restart the protocol. Now after the restart, an *honest* D is guaranteed to obtain a candidate Core where $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t+p$, since the set $\mathcal{H}$ always constitute a candidate Core set. Consequently, for an *honest* D, the sharing phase after restart will not fail and if at all D fails to produce a Core set where $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t+p$, it will imply that D is *corrupt* and the parties will terminate the protocol with a default output.

From the above discussion it follows that the worst case occurs when the sharing-phase fails once and when the tag-generation phase fails once, resulting in $|\mathsf{Dp}| \geq t-p+1$ and $|\mathsf{Cr}| \geq t-p+1$. This is followed by the third run, where either all the phases succeed or otherwise D is publicly discarded. □

Lemma D.5 (Correctness). *In protocol* $\Pi_{A\langle\cdot\rangle}$ *if* D *is honest and participates with input* $\mathbb{S} \in \mathbb{F}^{LMp^2}$*, then except with probability* $2^{-\Omega(\kappa)}$*, all (honest) parties output their respective information corresponding to* ${}^{A}\langle \mathbb{S} \rangle_d$*.*

Proof. From Claim D.1, an honest D is never discarded, except with a negligible probability. An honest D sends consistent row and column polynomials to all the parties. Moreover, it receives signatures on shares from all honest parties during the sharing phase. Hence, each honest party is always included in Core. It is easy to see that throughout the protocol, each honest P_i holds $\mathbf{s}_i^{(u)}$ as its share that is received from D and does not update it. Moreover, the share $\mathbf{s}_i^{(u)} = 0^{Mp}$ for each $P_i \in \mathsf{Dp} \cup \mathsf{Cr}$ is also consistent with D's bivariate polynomials used during the sharing phase and does not change during the protocol execution. It is therefore established that all honest parties hold shares that are consistent with the dealer's bivariate polynomials that embed the secrets. It remains to show that parties output these shares together with consistent authentication information. We consider the following two cases for the tag-generation phase.

Case 1 - Parties compute their output during Step 3c of tag share verification: In this case, either every $P_i \notin \mathsf{Cr}$ broadcasts Tag_i or $\mathsf{flag}_2 = 1$. In the former case, each P_i has $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \geq 2t+p$ and is able to compute $\tau_{ij}^{(u)}$ corresponding to each P_j. In the latter case, since $\mathsf{flag}_2 = 1$, it must hold that the protocol identified a set of at least $t-p+1 > p$ corrupt parties Cr (Claim D.2) in the previous run and restarted with their shares set to all-zero vectors. Moreover, since the honest parties always send correct tag shares, $\mathcal{R}_{ij}$ of each honest P_i will include all the honest parties. Hence, in this case as well, it must hold that $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \geq 2t+1+(t-p+1) \geq 2t+p$ (as $p = \frac{t}{4}+1$) ensuring that each $P_i \notin \mathsf{Cr}$ can compute its tag corresponding to each P_j.

Case 2 - Parties compute their output during Step 4d of tag share complaint resolution: This implies that the parties have not identified a set of corrupt parties $|\mathsf{Cr}| \geq t-p+1$. Otherwise, they restart with $\mathsf{flag}_2 = 1$ and parties would compute their output via Step 3c of tag share verification as discussed in the prior case, which is a contradiction. This implies that (at least)

$3t+1-|\mathsf{Cr}| \geq 2t+p$ parties P_l are outside Cr and indeed reveal their tag shares $(\tau_{ij,l}^{(1)}, \ldots, \tau_{ij,l}^{(L+4)})$ to each $P_i \notin \mathsf{Cr}$, either during tag share computation (Step 2) or during tag share complaint resolution (Step 4a). Hence, each $P_i \notin \mathsf{Cr}$ has $|\mathcal{R}_{ij} \cup \mathsf{Cr}| \geq 2t+p$ and is able to reconstruct $\tau_{ij}^{(u)}$ corresponding to each P_j.

The correctness of the tags computed can be easily established as follows. Consider a pair of parties (P_i, P_j). During this phase, each $P_l \notin \mathsf{Cr}$ sends its share of $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^{i}$ to each $P_i \notin \mathsf{Cr}$. If P_l's tag shares $\tau_{ij,l}^{(u)}$ sent to P_i are accepted by an (honest) P_i during reconstruction of $\tau_{ij}^{(u)}$, that is P_i includes P_l in $\mathcal{R}_{ij}$, then the following must hold for each $u \in \{1, \ldots, L+4\}$: $\tau_{ij,l}^{(u)} = \mathbf{s}_l^{(u)} \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}^{(u)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(u,k)}$, where $r_{ji,l}^{(u,k)}$ denotes P_l's share of $\lceil r_{ji}^{(u,k)} || k \rfloor_t^i$.

Given that the dealer D is not discarded, it has broadcasted $\mathsf{F}^{(3,1)}(X,Y), \ldots, \mathsf{F}^{(3,M)}(X,Y)$ and for each $v \in \{1, \ldots, M\}$, the following holds:

$$\mathsf{F}^{(3,v)} = F^{(1,v)}(X,Y) + r_3 \cdot F^{(2,v)}(X,Y) + \ldots + r_3^{L+3} \cdot F^{(L+4,v)}(X,Y).$$

This further implies that $\mathbf{s}_l = \mathbf{s}_l^{(1)} + r_3 \cdot \mathbf{s}_l^{(2)} + \ldots + r_3^{L+3} \cdot \mathbf{s}_l^{(L+4)}$ holds. Now consider the following linear combination of the tags:

$$\tau_{ij,l} = \tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}.$$

If $\tau_{ij,l}^{(u)} \neq \mathbf{s}_l^{(u)} \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}^{(u)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(u,k)}$ for some u, then except with negligible probability, the following holds true:

$$\tau_{ij,l} \neq \mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l} + \Bigg((v_{ji,l}^{(1)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(1,k)}) + r_3 \cdot (v_{ji,l}^{(2)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(2,k)}) + \ldots + r_3^{L+3} \cdot (v_{ji,l}^{(L+4)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(L+4,k)}) \Bigg).$$

Now consider the twisted t-sharing $\lceil \beta_{ij,l} \rfloor_t^i$ which is equals to

$$\begin{aligned} &= \lceil \mathbf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \left(\lceil v_{ji}^{(1)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(1,k)} || k \rfloor_t^i \right) + \\ &r_3 \left(\lceil v_{ji}^{(2)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(2,k)} || k \rfloor_t^i \right) + \ldots \\ &+ r_3^{L+3} \cdot \left(\lceil v_{ji}^{(L+4)} \rfloor_t^i + \sum_{k=1}^{d} e_{k,l} \cdot \lceil r_{ji}^{(L+4,k)} || k \rfloor_t^i \right) + \lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i \end{aligned}$$

It is easy to see that P_l's share in $\lceil \mathbf{s}_l \rfloor_0^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ will be $\mathbf{s}_l \cdot \boldsymbol{\mu}_{ji,l}$. We also know that P_l's share in $\lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$ will be 0, as $\lceil \mathsf{mask}_{ji}^{(l)} \rfloor_t^i$ is generated using $\mathcal{F}_{\lceil \cdot \rfloor_t^i}^{\mathsf{R}}$ that sets the share of P_l to 0. It then turns out that the share of P_l in the rest of

twisted t-sharing above will be the same as $(v_{ji,l}^{(1)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(1,k)}) + r_3 \cdot (v_{ji,l}^{(2)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(2,k)}) + \ldots + r_3^{L+3} \cdot (v_{ji,l}^{(L+4)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(L+4,k)})$. Consequently, the share of P_l in $\lceil \beta_{ij,l} \rfloor_t^i$ will not be the same as $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$. From the protocol steps, P_i will robustly reconstruct the entire sharing $\lceil \beta_{ij,l} \rfloor_t^i$ and find that the share of P_l is not the same as $\tau_{ij,l}^{(1)} + r_3 \cdot \tau_{ij,l}^{(2)} + \ldots + r_3^{L+3} \cdot \tau_{ij,l}^{(L+4)}$. Hence, an (honest) P_i will include P_l in Dp_i and not accept its tag shares.

Then consider the tag-share complaint resolution phase. By using similar arguments as above, it can be shown that P_l's share in $\lceil \gamma_{ij,l} \rfloor_t^i$ will not be the same as $\tau_{ij,l}^{(1)} + r_4 \cdot \tau_{ij,l}^{(2)} + \ldots + r_4^{L+3} \cdot \tau_{ij,l}^{(L+4)}$. Hence, except with a negligible probability, P_l will not get included in $\mathcal{R}_{ij}$ and its shares will not be used for reconstructing the sharing $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^i$ and compute $\tau_{ij}^{(u)} = \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$. However, this is a contradiction, and hence we have that $\tau_{ij,l}^{(u)} = \mathbf{s}_l^{(u)} \cdot \boldsymbol{\mu}_{ji,l} + v_{ji,l}^{(u)} + \sum_{k=1}^{d} e_{k,l} \cdot r_{ji,l}^{(u,k)}$ for each $u \in \{1, \ldots, L+4\}$. Finally, P_i reconstructs its tag $\tau_{ij}^{(u)}$ from the shares of parties for which the above condition holds. Hence, we are guaranteed that it reconstructs the sharing corresponding to $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^i = \lceil \mathbf{s}_i^{(u)} \rfloor_d^i \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_t^i + \lceil v_{ji}^{(u)} \rfloor_t^i + \sum_{k=1}^{d} [[\mathbf{e}_k]]_d^i \cdot \lceil r_{ji}^{(u,k)} || k \rfloor_t^i$. It is easy to see that the polynomial corresponding to $\sum_{k=1}^{d} [[\mathbf{e}_k]]_d^i \cdot \lceil r_{ji}^{(u,k)} || k \rfloor_t^i$ evaluates to 0 at i due to the degenerate packed twisted sharing of $\mathbf{e}_1, \ldots, \mathbf{e}_d$. This ensures that $\tau_{ij}^{(u)}$ which is embedded at i in the underlying polynomial for $\lceil \tau_{ij}^{(u)} \rfloor_{t+d}^i$ is indeed $\mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$.

In conclusion, when the dealer is not discarded, except with negligible probability, for each party $P_i \notin \mathsf{Cr}$ we have that:

- P_i outputs $\mathbf{s}_i^{(u)}$ as its shares of the secrets. Moreover, from Claim D.1, for each honest P_i, it holds that $P_i \notin \mathsf{Cr}$.
- Also, P_i outputs $(\mathsf{K}_{i1}^{(u)}, \ldots, \mathsf{K}_{in}^{(u)})$ as the MAC-keys, where each $\mathsf{K}_{ij}^{(u)} = (\boldsymbol{\mu}_{ij}, v_{ij}^{(u)})$. Of these, $\boldsymbol{\mu}_{ij}$ is known to P_i at the onset of the protocol. Whereas, $v_{ij}^{(u)}$ for each $P_j \notin \mathsf{Cr}$ is robustly reconstructed from $\lceil v_{ij}^{(u)} \rfloor_t^j$ used during tag computation, and $v_{ij}^{(u)} = 0$ for each $P_j \in \mathsf{Cr}$.
- P_i outputs $(\tau_{i1}^{(u)}, \ldots, \tau_{in}^{(u)})$ as MAC-tags, where $\tau_{ij}^{(u)} = \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$ holds.

This implies that except with a negligible probability, all (honest) parties output their respective information corresponding to ${}^{A}\langle \mathbb{S} \rangle_d$. □

Lemma D.6 (Commitment). *If D is corrupt, then except with probability $2^{-\Omega(\kappa)}$, there exists some $\mathbb{S} \in \mathbb{F}^{LMp^2}$ known to D, such that all (honest) parties output their respective information corresponding to ${}^{A}\langle \mathbb{S} \rangle_d$ at the end of $\Pi_{A\langle\cdot\rangle}$.*

Proof. If D is discarded prematurely, then the lemma holds trivially. So we consider the case when D is not discarded and analyze the information held by the (honest) parties in each phase of the protocol.

- **Sharing Phase:** It holds that either in the first or the second run of the protocol, D broadcasts a Core such that $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t + p$ and OK_i

is broadcasted by each $P_i \in \mathsf{Core}$. This implies that there are at least $|\mathsf{Core} \cup \mathsf{Dp}| - t \geq t + p \geq d + 1$ honest parties in $\mathsf{Core} \cup \mathsf{Dp}$, where the honest parties in Core have received degree-d column polynomials $g_i^{(u,v)}(Y)$ from D and gave the signatures on the points on their column-polynomials to D. On the other hand, the column-polynomials of the honest parties in Dp are supposed to be 0. For each $u \in \{1, \ldots, L\}$ and $v \in \{1, \ldots, M\}$, let $F^{(u,v)}(X, Y)$ be the degree-(d, d) bivariate polynomial, defined by the column-polynomials of the first $d + 1$ honest parties in $\mathsf{Core} \cup \mathsf{Dp}$, with the column-polynomials of the honest parties in Dp supposedly being 0. Moreover, let $\mathbf{s}^{(u,(v-1)\cdot p)}, \ldots, \mathbf{s}^{(u,v\cdot p)}$ be the vectors, each consisting of p elements, embedded in $F^{(u,v)}(X, Y)$ at $F^{(u,v)}(X, -1), \ldots, F^{(u,v)}(X, -p)$ respectively. Furthermore, let $\mathbb{S} \stackrel{\mathsf{def}}{=} (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$, where $\mathbf{S}^{(u)} \stackrel{\mathsf{def}}{=} (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)})$.

- **Verification Phase:** It holds that $\mathsf{F}^{(1,1)}(X, Y), \ldots, \mathsf{F}^{(1,M)}(X, Y)$ are broadcasted by D and that D successfully revealed $\mathsf{Sig}(P_i \to D, \mathsf{G}_i^{(1,1)}, \ldots, \mathsf{G}_i^{(1,M)})$ publicly, for every $P_i \in \mathsf{Core}$. By the correctness (Lemma C.1) and unforgeability of ICP (Lemma C.4), except with negligible probability, it holds that $\mathsf{G}_i^{(1,v)}(y) = g_i^{(1,v)}(Y) + r_1 \cdot g_i^{(2,v)}(i, Y) + \ldots + r_1^{L+3} \cdot g_i^{(L+4,v)}(i, Y)$ for every honest $P_i \in \mathsf{Core}$ where $g_i^{(u,v)}(Y)$ is held by P_i as its column-polynomial from the previous phase. Moreover, for every honest $P_i \in \mathsf{Dp}$, it must hold that $\mathsf{F}^{(1,v)}(X, i) = \mathbf{0}$ and $\mathsf{F}^{(1,v)}(i, Y) = \mathbf{0}$. Further, since all the honest parties in Core hold row and column-polynomials consistent with the same linear combination of degree-d bivariate polynomials $\mathsf{F}^{(1,1)}(X, Y), \ldots, \mathsf{F}^{(1,M)}(X, Y)$, except with negligible probability, it implies that they all hold polynomials $f_i^{(u,v)}(Y), g_i^{(u,v)}(Y)$ consistent with the defined bivariate polynomial $F^{(u,v)}(X, Y)$.
- **Non-Core Row and Column Reconstruction Phase:** Each honest $P_i \in \mathsf{Core}$ sends its common points $\{(f_i^{(u,v)}(j), g_i^{(u,v)}(j))\}_{u=1,\ldots,L+4, v=1,\ldots,M}$ to each $P_j \in \mathcal{P} \setminus (\mathsf{Core} \cup \mathsf{Dp} \cup \mathsf{Cr})$. Moreover, due to the same argument as above, except with negligible probability, $\mathsf{F}^{(2,v)}(i, j) = \sum_{u=1}^{L+4} (r_2)^{u-1} g_i^{(u,v)}(j)$ and $\mathsf{F}^{(2,v)}(j, i) = \sum_{u=1}^{L+4} (r_2)^{u-1} f_i^{(u,v)}(j)$ holds for each honest $P_i \in \mathsf{Core}$. Hence, each (honest) $P_j \notin \mathsf{Core} \cup \mathsf{Dp} \cup \mathsf{Cr}$ has $P_i \in \mathcal{S}_j$. Since P_j interpolates $f_j^{(u,v)}(Y)$ and $g_j^{(u,v)}(Y)$ from points obtained from parties in $\mathcal{S}_j$ and parties in Dp and due to the guarantee that $|\mathsf{Core} \cup \mathsf{Dp}| - t \geq t + p$, we have that P_j's updated row and column-polynomials are consistent with the polynomials of the honest parties in $\mathsf{Core} \cup \mathsf{Dp}$. Hence, at the end of this phase, all honest parties hold row and column-polynomials that are consistent and define unique degree-(d, d) bivariate polynomials, with the rows and column-polynomials of the parties in Dp being 0.
- **Pairwise Tag Generation:** It remains to show that parties output shares together with consistent authentication information corresponding to ${}^{A}\langle \mathbb{S} \rangle_d$. However, the proof for this is identical to the proof of Lemma D.5.

□

Lemma D.7 (Complexity). *Protocol* $\Pi_{A\langle\cdot\rangle}$ *requires a constant number of rounds and incurs a communication of* $\mathsf{P2P}((n^2LM+n^4L+n^5)\kappa)+\mathsf{BC}(n^2L\kappa)+\mathsf{BC}(n^2M\kappa)+n\times\mathsf{BC}(n^2L\kappa)$ *bits, apart from at most* 15 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

Proof. We first do the analysis assuming parties do not restart the protocol.

During the sharing phase, D distributes $\mathcal{O}(LM)$ rows and column polynomials to each party, where each polynomial is a d-degree polynomial and hence consists of $\mathcal{O}(n)$ field elements as coefficients. This requires one round and incurs a total communication of $\mathcal{O}(n^2LM)$ field elements. Every party upon receiving its rows and column polynomials may broadcast an OK message, so this step requires a communication of $n\times\mathsf{BC}(1)$ bits. Additionally each P_i may sign its column polynomials for D, for which $\mathcal{O}(L)$ instances of Gen and Auth protocols are executed, where in each instance, P_i as a signer has M vectors for signing, each consisting of n field elements. From Lemma C.5, by substituting $l=n$, this step requires a total communication of $\mathsf{P2P}(n^2LM\kappa)+\mathsf{BC}(n^2L\kappa)+n\times\mathsf{BC}(nL)$ bits.[14] Finally, D may broadcast the identity of a Core set, which requires a communication of $\mathsf{BC}(n)$ bits. Hence, the sharing phase will require a constant number of rounds and incurs a communication of $\mathsf{P2P}(n^2LM\kappa)+\mathsf{BC}(n^2L\kappa)+n\times\mathsf{BC}(nL)$ bits.

During verification phase, the parties call $\mathcal{F}_{\mathsf{Rand}}$ to generate r_1. The dealer D broadcasts M bivariate polynomials. This requires a communication of $\mathsf{BC}(n^2M\kappa)$ bits. Additionally, D may invoke n instances of Rev to publicly reveal IC-signatures on M vectors, each consisting of n field elements. Form Lemma C.5, by substituting $l=n$, this step requires two rounds and incurs a communication of $\mathsf{BC}(n^2M\kappa)+n\times\mathsf{BC}(n)$ bits.[15]

Next consider the non-Core row and column reconstruction phase. Here every party in Core may send the supposedly common points on their row and column polynomials to the parties outside Core. This requires one round and a total communication of $\mathsf{P2P}(n^2LM\kappa)$ bits. The parties then call $\mathcal{F}_{\mathsf{Rand}}$ to generate the random value r_2. This is followed by similar steps as in the verification phase, where D may broadcast M bivariate polynomials and publicly reveals IC-signatures of up to n parties on M vectors, each consisting of n field elements. So overall this phase will require a constant number of rounds and incurs a communication of $\mathsf{P2P}(n^2LM\kappa)+\mathsf{BC}(n^2M\kappa)+n\times\mathsf{BC}(n)$ bits.

The next phase is the pairwise tag generation phase. As part of the setup, for every pair of parties (P_i,P_j), parties have to generate twisted t-sharing of $\mathcal{O}(nL)$ random values with the shares of the parties in Cr being 0 in all these sharings. So $\mathcal{O}(n^3L)$ random t-sharings need to be generated via $\mathcal{F}^{\mathsf{R}}_{\lceil\cdot\rfloor_t}$, which costs $\mathsf{P2P}(n^4L\kappa)+n\times\mathsf{BC}(n^2\kappa)$ bits. Additionally, for every triplet of parties

[14] Since in all nL instances of Gen and Auth, the dealer D plays the role of I, it can simultaneously broadcast all the data across all the nL instances of Auth. Similarly, each party can simultaneously broadcast its Boolean response for all the nL instances of Auth.

[15] Since D is playing the role of I in all the Rev, it can simultaneously broadcast all the data across these Rev instances. Similarly, each party can simultaneously broadcast its Boolean response across all the Rev instances.

(P_i, P_j, P_l), the parties generate twisted t-sharing of two random masks, with shares of P_l being 0. For this, the parties make n calls to $\mathcal{F}^{\mathsf{R}}_{\lceil \cdot \rfloor^i_t}$, where in each call, $\mathcal{O}(n^2)$ random sharings are generated, with shares of a designated party being 0. One can optimize the realization of instances $\mathcal{F}^{\mathsf{R}}_{\lceil \cdot \rfloor^i_t}$ here. Recall that to securely realize one call to $\mathcal{F}^{\mathsf{R}}_{\lceil \cdot \rfloor^i_t}$ for generating N random t-sharings, each party has to act as a dealer and generate $\frac{N}{n-t} + 1$ random t-sharings via $\mathcal{F}_{[\cdot]_{\mathsf{t}}}$, prove the "correctness" of its sharings and then parties apply randomness extraction to extract N random t-sharings. In our case, there are n calls to $\mathcal{F}^{\mathsf{R}}_{\lceil \cdot \rfloor^i_t}$, where in each call, n^2 random t-sharings have to be generated. To securely realize all these calls, every party has to generate $\mathcal{O}(\frac{n^3}{n-t})$ random t-sharings via instances of $\mathcal{F}_{[\cdot]_{\mathsf{t}}}$. The observation here is that all these t-sharings can be generated in a single shot via a *single* call to $\mathcal{F}_{[\cdot]_{\mathsf{t}}}$ for each party and so n instances of $\mathcal{F}_{[\cdot]_{\mathsf{t}}}$ for n different dealer parties suffice. Hence, instantiating the n calls to $\mathcal{F}^{\mathsf{R}}_{\lceil \cdot \rfloor^i_t}$ will cost $\mathsf{P2P}(n^4\kappa) + n \times \mathsf{BC}(n^2\kappa)$ bits.

Next, for every triplet of parties (P_i, P_j, P_l), party P_l has to send $\mathcal{O}(L)$ tag-shares to P_i. This incurs a total communication of $\mathsf{P2P}(n^3 L\kappa)$ bits. To verify the tag-shares, the parties first call $\mathcal{F}_{\mathsf{Rand}}$ to generate a random value r_3. This is followed by D making public M bivariate polynomials and IC-signatures of up to n parties on M vectors, each consisting of n field elements. Similar to previous phases, this requires a constant number of rounds and a communication of $\mathsf{BC}(n^2 M\kappa) + n \times \mathsf{BC}(n)$ bits. Then for every triplet of parties (P_i, P_j, P_l), party P_i reconstructs a twisted t-sharing. This requires one round and incurs a communication of $\mathsf{P2P}(n^4\kappa)$ bits. Every party may then broadcast the list of disputed parties from which it supposedly received incorrect tag-shares. This requires a communication of $n \times \mathsf{BC}(n\kappa)$ bits.

The final step is the tag-share complaint resolution phase. Here in the worst case, for up to $\mathcal{O}(n^2)$ pairs of parties (P_i, P_l), party P_l may broadcast $\mathcal{O}(nL)$ tag-shares. This requires a communication of $n \times \mathsf{BC}(n^2 L\kappa)$ bits. The parties then call $\mathcal{F}_{\mathsf{Rand}}$ to generate a random value r_4. This is followed by D making making public M bivariate polynomials and IC-signatures of up to n parties on M vectors, each consisting of n field elements. This requires a communication of $\mathsf{BC}(n^2 M\kappa) + n \times \mathsf{BC}(n)$ bits. Finally, in the worst case, for up to $\mathcal{O}(n^2)$ pairs of parties (P_i, P_l), there will be n number of twisted t-sharings which will be publicly reconstructed. This will require one round and a communication of $\mathsf{P2P}(n^5\kappa)$ bits. So overall the tag-generation phase will require a constant number of rounds and a communication of $\mathsf{P2P}(n^4 L\kappa + n^5\kappa) + \mathsf{BC}(n^2 M\kappa) + n \times \mathsf{BC}(n^2 L\kappa)$ bits.

Hence, if the protocol does not restart, then it requires a constant number of rounds and incurs a communication of $\mathsf{P2P}((n^2 LM + n^4 L + n^5)\kappa) + \mathsf{BC}(n^2 L\kappa) + \mathsf{BC}(n^2 M\kappa) + n \times \mathsf{BC}(n^2 L\kappa)$ bits Additionally, there will be four calls to $\mathcal{F}_{\mathsf{Rand}}$ to generate the random values r_1, r_2, r_3 and r_4 which are used by D to make public its linear combinations of bivariate polynomials.

Next consider the case when the parties restart the protocol. As shown in Lemma D.4, the parties need to rerun the protocol at most 3 times to get an

output. Since each run requires a constant number of rounds, it follows that protocol $\Pi_{A\langle\cdot\rangle}$ will overall require a constant number of rounds to generate output for the parties. Moreover, a communication of $\mathsf{P2P}((n^2LM + n^4L + n^5)\kappa) + \mathsf{BC}(n^2L\kappa) + \mathsf{BC}(n^2M\kappa) + n \times \mathsf{BC}(n^2L\kappa)$ bits will be involved, apart from 15 calls to $\mathcal{F}_{\mathsf{Rand}}$. □

Lemmas D.3, D.6 and D.7, complete the proof of Theorem 5.2 for $\Pi_{A\langle\cdot\rangle}$.

Protocol $\Pi_{A\langle\cdot\rangle}$ for Multiple Dealers and Larger Inputs. In protocol $\Pi_{A\langle\cdot\rangle}$, dealer D has LMp^2 elements from $\mathbb{F}$ as input. If D has multiple groups of n^5p^2 elements to be ${}^{A}\langle\cdot\rangle_d$-shared, then it can run an instance of $\Pi_{A\langle\cdot\rangle}$ for each group. The calls to $\mathcal{F}_{\mathsf{Rand}}$ can be reused to generate the random challenges for D across all the instances of $\Pi_{A\langle\cdot\rangle}$ during various phases. For instance, during the verification phase of all the instances of $\Pi_{A\langle\cdot\rangle}$, a single call to $\mathcal{F}_{\mathsf{Rand}}$ is sufficient to generate the random challenge r_1, which has to be used by D to compute its linear combinations of bivariate polynomials across all the instances of $\Pi_{A\langle\cdot\rangle}$. This will ensure that the total number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is upper bounded by 15.

In our MPC protocol, we will also come across scenarios where multiple parties have to act as dealers and invoke instances of $\Pi_{A\langle\cdot\rangle}$ at the same "time". For efficiency, the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be "clubbed" across all the instances of $\Pi_{A\langle\cdot\rangle}$ invoked by different dealers. This will be possible because the underlying network is synchronous and all the instances of $\Pi_{A\langle\cdot\rangle}$ (even by different dealers) will be executed at the same time. Hence the maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ will be still 15.

D.2 Extending Protocol $\Pi_{A\langle\cdot\rangle}$ for Degrees $d+p-1$ and $p-1$

In our MPC protocol we will come across situations where a dealer has to generate ${}^{A}\langle\cdot\rangle_\ell$-sharing of its inputs where $\ell \in \{p-1, d, d+p-1\}$. While we already discussed how $\Pi_{A\langle\cdot\rangle}$ can handle $\ell = d$, we now discuss the modifications needed to handle the remaining cases. We stress that the communication complexity (and the number of calls to $\mathcal{F}_{\mathsf{Rand}}$) remains the same as earlier.

Extending $\Pi_{A\langle\cdot\rangle}$ for degree $d+p-1$. Let D have an input $\mathbb{S} \in \mathbb{F}^{LMp^2}$ such that $\mathbb{S} = \{s^{(u,v,w)}\}_{u\in\{1,\ldots,L\},v\in\{1,\ldots,Mp\},w\in\{1,\ldots,p\}}$. Let the elements of $\mathbb{S}$ be structured into L batches $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)}$, where each $\mathbf{S}^{(u)}$ consists of Mp vectors $\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,Mp)}$ and where each $\mathbf{s}^{(u,v)}$ consists of the p elements $(s^{(u,v,1)}, \ldots, s^{(u,v,p)})$. The goal is to generate ${}^{A}\langle\mathbf{S}^{(1)}\rangle_{d+p-1}, \ldots, {}^{A}\langle\mathbf{S}^{(L)}\rangle_{d+p-1}$, resulting in ${}^{A}\langle\mathbb{S}\rangle_{d+p-1}$.

- **Sharing phase**: Here the differences are that D embeds its inputs in bivariate polynomials of degree-$(d+p-1)$, instead of d. Moreover, we require D to find a Core where $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t + 2p - 1 = d + p + t$.
- **Verification phase**: This is similar as before, except that degree of the various polynomials are changed from d to $d+p-1$.
- **Non-Core row and column reconstruction phase**: Similar to the case of $\ell = d$, presence of (at least) $d+p$ honest parties in $\mathsf{Core} \cup \mathsf{Dp}$ ensures that

parties outside Core can compute their rows and column polynomials lying on dealer's committed bivariate polynomials successfully.

- **Pairwise tag-generation phase**: During the computation of the tag $\tau_{ij}^{(u)} \stackrel{\text{def}}{=} \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$, parties will hold $\lceil \tau_{ij}^{(u)} \rfloor_{2d}^{i}$. This is because $\mathbf{s}_i^{(u)}$ will be degree-$(d+p-1)$-twisted-shared and $\boldsymbol{\mu}_{ji}$ will be degree-t-twisted-shared and hence $\lceil \mathbf{s}_i^{(u)} \rfloor_{d+p-1}^{i} \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_{t}^{i} = \lceil \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} \rfloor_{2t+p-1}^{i}$. To randomize the degree-$(2t+p-1)$ sharing of $\tau_{ij}^{(u)}$, parties now need to generate degree-t twisted-sharing of $2t+p-1-t = t+p-1$ random values. Moreover, the publicly known vectors used in tag computation will now be $\mathbf{e}_1, \ldots, \mathbf{e}_{t+p-1}$, each containing $t+p$ elements from $\mathbb{F}$, where $\mathbf{e}_1 = (0,1,0,0,\ldots,0), \mathbf{e}_2 = (0,0,1,0,\ldots,0), \ldots \mathbf{e}_{t+p-1} = (0,\ldots,0,1)$.

Extending $\Pi_{A\langle\cdot\rangle}$ for degree $p-1$ (sharing non-private values). During our MPC protocol, we will come across situations where a designated $D \in \mathcal{P}$ has input $\mathbb{S} \in \mathbb{F}^{LMp^2}$, such that $\mathbb{S}$ needs to be learned publicly. To maintain $\mathcal{O}(1)$ communication overhead, D would like to generate ${}^{A}\langle \mathbb{S} \rangle_{p-1}$, where the elements of $\mathbb{S}$ are structured as earlier. That is, $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L)})$, where $\mathbf{S}^{(u)} = \{\mathbf{s}^{(u,v)}\}_{v \in \{1,\ldots,Mp\}}$. To achieve the above goal, D embeds each vector $\mathbf{s}^{(u,v)}$ in a $p-1$-degree polynomial. Note that these polynomials are no longer randomly chosen, since each $\mathbf{s}^{(u,v)}$ has p elements. The sharing-polynomials are further embedded in degree-$(p-1)$ bivariate polynomials, which are *no* longer randomly chosen, as each $F^{(u,v)}(X,Y)$ embeds p^2 elements from the vectors $\{\mathbf{s}^{(u,(v-1)\cdot p)+1)}, \ldots, \mathbf{s}^{(u,v\cdot p)}\}$. As each $\mathbf{s}^{(u,v)}$ is $(p-1)$-packed-secret-shared, we could avoid the tag-generation phase of $\Pi_{A\langle\cdot\rangle}$, since with $n = 3t+1$, we can robustly reconstruct a $(p-1)$-degree polynomial using RS error-correction (since $p = \frac{t}{4}+1$). However, we generate the authentication tags even for $\ell = p-1$ to maintain a generic protocol. The changes needed are described below.

- **Sharing phase**: D picks degree-$(p-1)$ bivariate polynomials as described above. We now require D to find a Core where $|\text{Core}| \geq t+p$. An *honest* D will certainly find a Core, since we have at least $2t+1$ honest parties. Hence we *do not* have any reruns and the parties *need not* maintain any Dp set.
- **Verification phase**: This phase proceeds similarly as earlier, with parties verifying that the rows and columns of the (honest) parties in Core lying on consistent degree-$(p-1)$ bivariate polynomials.
- **Non-Core row and column reconstruction phase**: The presence of (at least) p honest parties in Core ensures that (honest) parties outside Core can compute their corresponding row and column polynomials successfully.
- **Pairwise tag-generation phase**: During the computation of the MAC-tag $\tau_{ij}^{(u)} \stackrel{\text{def}}{=} \mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji} + v_{ji}^{(u)}$, parties will hold $\lceil \tau_{ij}^{(u)} \rfloor_{d}^{i}$. This is because $\mathbf{s}_i^{(u)}$ will be now $(p-1)$-degree twisted-shared and hence $\lceil \mathbf{s}_i^{(u)} \rfloor_{p-1}^{i} \cdot \lceil \boldsymbol{\mu}_{ji} \rfloor_{t}^{i} = [\mathbf{s}_i^{(u)} \cdot \boldsymbol{\mu}_{ji}]_{d}^{i}$. To randomize $\lceil \tau_{ij}^{(u)} \rfloor_{d}^{i}$, parties now need to generate degree-t twisted-sharing of $p-1$ random values. Moreover, the publicly known vectors used in tag computation will be $\mathbf{e}_1, \ldots, \mathbf{e}_{p-1}$, each containing p elements from $\mathbb{F}$, where

$\mathbf{e}_1 = (0, 1, 0, 0, \ldots, 0), \mathbf{e}_2 = (0, 0, 1, 0, \ldots, 0), \ldots \mathbf{e}_{p-1} = (0, \ldots, 0, 1)$. Finally, due to the presence of $2t + 1$ honest parties, this phase *always* succeed.

D.3 Simulator for $\Pi_{A\langle\cdot\rangle}$

We describe a simulator for $\Pi_{A\langle\cdot\rangle}$ which will be used when $\Pi_{A\langle\cdot\rangle}$ is used as a sub-protocol.

Simulator D.8: Simulator for the Protocol $\Pi_{A\langle\cdot\rangle}$

We assume that the simulator has access to the MAC-key setup information. This will be guaranteed while simulating the outer protocol Π_{Off} for the preprocessing phase, since the simulator itself emulates the role of $\mathcal{F}_{\mathsf{MacKeys}}$. The same is ensured while simulating the outer MPC protocol Π_{MPC}, since the simulator itself emulates $\mathcal{F}_{\mathsf{Off}}$ which establishes the setup information.

– **Case 1: Honest Dealer**

- On behalf of the honest parties, initialize $\mathsf{Dp}, \mathsf{Cr} = \phi$.
- Select $L + 4$ random $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L+4)} \in \mathbb{F}^{Mp^2}$ random two-dimensional vectors on behalf of an honest dealer.
- Choose $(L + 4)M$ degree-d bivariate polynomials $F^{(u,v)}(X, Y)$ over $\mathbb{F}$ for $u = 1, \ldots, L + 4$ and $v = 1, \ldots, M$, which embeds the secrets from $\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(L+4)}$, as in the protocol specification.
- For every $P_k \notin \mathsf{Dp} \cup \mathsf{Cr}$ that is under the control of $\mathcal{A}$, let $f_k^{(u,v)}(X) \stackrel{\mathsf{def}}{=} F^{(u,v)}(X, k)$ and $g_k^{(u,v)}(Y) \stackrel{\mathsf{def}}{=} F^{(u,v)}(k, Y)$. On behalf of the dealer, send the polynomials $\{f_k^{(u,v)}(X), g_k^{(u,v)}(Y)\}_{u=1,\ldots,L+4, v=1,\ldots,M}$ to $\mathcal{A}$.
- On behalf of each honest party P_i not under the control of $\mathcal{A}$, emulate broadcast and send OK_i to $\mathcal{A}$.
- For some P_k controlled by $\mathcal{A}$, if OK_k is received from $\mathcal{A}$, then emulate broadcast and send OK_k to all. Additionally, simulate the Gen algorithm of ICP if $\mathcal{A}$ gives $\mathsf{Sig}(P_k \rightarrow D, \mathbf{g}_k^{(u,1)}, \ldots, \mathbf{g}_k^{(u,M)})$ for $u = 1, \ldots, L + 4$ corresponding to some P_k controlled by it by receiving the corresponding values on behalf of the honest dealer and all the honest parties.
- On behalf of the dealer, include each honest P_i not under the control of $\mathcal{A}$ in the set Core. For each P_k under the control of $\mathcal{A}$ that sent the signature, simulate the Auth algorithm of ICP. For each P_k under the control of $\mathcal{A}$, if $\mathcal{A}$ sent OK_k and the signature of P_k is received in the simulation of ICP, then include P_k in Core.
- If $|\mathsf{Core} \cup \mathsf{Dp}| \geq 2t+p$, on behalf of the dealer assume Core to be broadcast, emulate broadcast and send Core to the $\mathcal{A}$. Otherwise, set $\mathsf{Dp} = \mathcal{P} \setminus \mathsf{Core}$. On behalf of the dealer assume Dp to be broadcast, emulate broadcast and send Dp to the $\mathcal{A}$.
- On behalf of the honest parties, set flag_1 as an honest party would, and if required, restart the simulation steps from the beginning.

- Emulate $\mathcal{F}_{\mathsf{Rand}}$ by receiving Rand from $\mathcal{A}$ and sending r_1 to $\mathcal{A}$.

- Perform the computation as per $\mathsf{Ver}(r_1)$ on behalf of the honest dealer. emulate broadcast by sending $(\mathsf{F}^{(1,1)}(X,Y),\ldots,\mathsf{F}^{(1,M)}(X,Y))$ to $\mathcal{A}$, and simulate the Rev algorithm of ICP for each party in Core.

- On behalf of each honest party $P_i \in \mathsf{Core}$, send the points $\{(f_i^{(u,v)}(k), g_i^{(u,v)}(k))\}_{u=1,\ldots,L+4,v=1,\ldots,M}$ to $\mathcal{A}$ for each $P_k \in \mathcal{P} \setminus (\mathsf{Core} \cup \mathsf{Dp} \cup \mathsf{Cr})$ under the control of $\mathcal{A}$.
- Receive Rand from $\mathcal{A}$ and emulate $\mathcal{F}_{\mathsf{Rand}}$ by sending a random r_2 to all.
- Perform the computation as per $\mathsf{Ver}(r_2)$ on behalf of the honest dealer. emulate broadcast by sending $(\mathsf{F}^{(2,1)}(X,Y),\ldots,\mathsf{F}^{(2,M)}(X,Y))$ to $\mathcal{A}$, and simulate the Rev algorithm of ICP for each party in Core.

- For every ordered pair of parties (P_i,P_j) such that $P_i \notin \mathsf{Cr}$, emulate $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ and communicate the appropriate shares to each party P_k under the control of $\mathcal{A}$.
- On behalf of each honest party P_l not under the control of $\mathcal{A}$, compute the sharing of the MAC-tag for every ordered pair (P_i,P_j) where $P_i \notin \mathsf{Cr}$ as per the protocol steps. For each P_k under the control of $\mathcal{A}$, on behalf of every honest P_l, send the tag-shares $(\tau_{kj,l}^{(1)},\ldots,\tau_{kj,l}^{(L+4)})$ corresponding to every ordered pair (P_k,P_j) where $P_k \notin \mathsf{Cr}$. On behalf of each honest P_i, for every ordered pair (P_i,P_j) receive the shares $(\tau_{ij,k}^{(1)},\ldots,\tau_{ij,k}^{(L+4)})$ from $\mathcal{A}$ corresponding to each $P_k \notin \mathsf{Cr}$.
- Receive Rand from $\mathcal{A}$ and emulate $\mathcal{F}_{\mathsf{Rand}}$ by sending a random r_3 to all.
- Perform the computation as per $\mathsf{Ver}(r_3)$ on behalf of the honest dealer. emulate broadcast by sending $(\mathsf{F}^{(3,1)}(X,Y),\ldots,\mathsf{F}^{(3,M)}(X,Y))$ to $\mathcal{A}$, and simulate the Rev algorithm of ICP for each party in Core.
- For every ordered triplet (P_k,P_j,P_l) such that $P_k,P_l \notin \mathsf{Cr}$, compute the shares of honest parties corresponding to $\lceil\beta_{kj,l}\rfloor_t^i$ and send them to $\mathcal{A}$ for every party P_k under the control of $\mathcal{A}$. On behalf of each honest P_i not under the control of $\mathcal{A}$, for every ordered triplet (P_i,P_j,P_l), receive shares of $\lceil\beta_{ij,l}\rfloor_t^i$ from $\mathcal{A}$ corresponding to each P_k controlled by $\mathcal{A}$.
- On behalf of each honest P_i, define the set $\mathcal{R}_{ij}$ as per the protocol specifications, and emulate broadcast on behalf of each honest P_i and send Tag_i or $(\mathsf{NOK},\mathsf{Dp}_i)$ to $\mathcal{A}$ based on the protocol steps.
- Receive from the $\mathcal{A}$, either Tag_k or $(\mathsf{NOK},\mathsf{Dp}_k)$ corresponding to each $P_k \notin \mathsf{Cr}$ that is under the control of $\mathcal{A}$ and send the same to all.
- For every ordered pair (P_i,P_j) where $P_i \notin \mathsf{Cr}$, receive shares of $\lceil v_{ji}^{(u)}\rfloor_t^i$ for every $u \in \{1,\ldots,L\}$ on behalf of every honest P_j from $\mathcal{A}$. On behalf of the honest parties, for every ordered pair (P_i,P_k) where $P_i \notin \mathsf{Cr}$, send shares of $\lceil v_{ki}^{(u)}\rfloor_t^i$ to $\mathcal{A}$ corresponding to each P_k under the control of $\mathcal{A}$.
- Emulate broadcast for each P_i which broadcasts $(\mathsf{NOK},\mathsf{Dp}_i)$ as per the protocol steps.
- Receive Rand from $\mathcal{A}$ and emulate $\mathcal{F}_{\mathsf{Rand}}$ by sending a random r_4 to all.
- Perform the computation as per $\mathsf{Ver}(r_4)$ on behalf of the honest dealer. emulate broadcast by sending $(\mathsf{F}^{(4,1)}(X,Y),\ldots,\mathsf{F}^{(4,M)}(X,Y))$ to $\mathcal{A}$, and simulate the Rev algorithm of ICP for each party in Core.

- For each P_i such that $(\mathsf{NOK}, \mathsf{Dp}_i)$ was for broadcast, on behalf of honest parties, send shares of $\lceil \gamma_{i1,l} \rfloor_t^i, \ldots, \lceil \gamma_{in,l} \rfloor_t^i$ to all for construction corresponding to each $P_l \in \mathsf{Dp}_i$.
- Populate the set Cr (and restart if necessary), and on behalf of each honest P_i, update $\mathcal{R}_{ij}$ as per protocol steps.
- For every order pair of parties (P_i, P_j) where $P_i \notin \mathsf{Cr}$, receive shares of $\lceil v_{ji}^{(u)} \rfloor_t^i$, for every $u \in \{1, \ldots, L\}$ from $\mathcal{A}$ on behalf of every honest P_j. On behalf of the honest parties, for every ordered pair (P_i, P_k) where $P_i \notin \mathsf{Cr}$, send shares of $\lceil v_{ki}^{(u)} \rfloor_t^i$ to $\mathcal{A}$ corresponding to each P_k under the control of $\mathcal{A}$.

– **Case 2: Corrupt Dealer**

- On behalf of every honest $P_i \notin \mathsf{Dp} \cup \mathsf{Cr}$, receive the polynomials $\{f_i^{(u,v)}(X), g_i^{(u,v)}(Y)\}_{u=1,\ldots,L+4, v=1,\ldots,M}$ from $\mathcal{A}$.
- On behalf of each honest party P_i, verify the received polynomials as per protocol specifications and if the conditions pass, it emulates broadcast sending OK_i to $\mathcal{A}$. It also simulates the steps of Gen algorithm of ICP on behalf of all the parties.
- emulate broadcast and receive either Core or Dp from $\mathcal{A}$ and send it to all.
- On behalf of the honest parties, perform the local verification of Core or Dp and if it fails, terminate. Otherwise continue to simulate the remaining protocol steps as below.

- Receive Rand from $\mathcal{A}$ and emulate $\mathcal{F}_{\mathsf{Rand}}$ by sending a random r_1 to all.
- Perform the computation as per $\mathsf{Ver}(r_4)$. emulate broadcast by receiving $(\mathsf{F}^{(1,1)}(X,Y), \ldots, \mathsf{F}^{(1,M)}(X,Y))$ from the $\mathcal{A}$ and send it to all. Simulate the Rev algorithm of ICP for each party in Core.
- On behalf of the honest parties, discard the dealer and terminate if any of the conditions fail as per the protocol specification.

- On behalf of the honest parties, simulate the rest of the protocol as per the protocol steps as in the case of an honest dealer.

E On the Size of SIMD Circuit

Given a set of elements $\mathbb{S}$, our VSS protocol can generate ${}^A\langle \mathbb{S} \rangle_\ell$ where $\ell \in \{p-1, d, d+p-1\}$ with a constant overhead only if $|\mathbb{S}| = n^3p^2$, where $d = t+p-1$. The elements in $\mathbb{S}$ need to be arranged into two-dimensional vectors $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(n)})$. Each $\mathbf{S}^{(u)}$ further consists of n^2p one-dimensional vectors where $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,n^2p)})$, with each $\mathbf{s}^{(u,v)} = (s^{(u,v,1)}, \ldots, s^{(u,v,p)})$. So $\mathbb{S} = \{s^{(u,v,w)}\}_{u \in \{1,\ldots,n^3\}, v \in \{1,\ldots,n^2p\}, w \in \{1,\ldots,p\}}$ and our VSS will generate ${}^A\langle \mathbb{S} \rangle_\ell$ with parties holding ${}^A\langle \mathbf{S}^{(1)} \rangle_\ell, \ldots, {}^A\langle \mathbf{S}^{(n)} \rangle_\ell$. In summary, *each one-dimensional vector* $\mathbf{s}$ *will contain* p *elements, each two-dimensional vector* $\mathbf{S}$ *will contain*

n^2p^2 elements and each three-dimensional vector $\mathbb{S}$ will contain n such two-dimensional vectors.

Looking ahead, to get $\mathcal{O}(1)$ communication overhead in our MPC protocol, the parties need to evaluate $\texttt{ckt}$ over $n(t+1)n^3p^2 = \Theta(n^7)$ inputs in parallel in an SIMD fashion. Hence each P_i will have $n(t+1)n^3p^2$ elements, corresponding to its input-wire x_i in $\texttt{ckt}$, These inputs will be further grouped into n groups, with each group consisting of $t+1$ sub-groups and where each sub-group will consist of n^3p^2 values, which will be arranged as $\{\mathbb{X}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ where each $|\mathbb{X}_i^{(\beta,\gamma)}| = n^3p^2$ and the parties will hold ${}^A\langle\mathbb{X}_i^{(\beta,\gamma)}\rangle_d$ as per the same structure as discussed in the previous paragraph. Each gate in $\texttt{ckt}$ is evaluated over $n(t+1)n^3p^2$ inputs in parallel, producing $n(t+1)n^3p^2$ outputs. Consequently, with every wire in $\texttt{ckt}$, there will be $n(t+1)n^3p^2$ elements which remains ${}^A\langle\cdot\rangle_d$-shared.

Each gate will be evaluated $n(t+1)$ times in parallel, over n^3p^2 inputs. In more detail, consider an arbitrary gate g in $\texttt{ckt}$ with inputs wires x, y and output wire z. Let $\{\mathbb{X}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ and $\{\mathbb{Y}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ be the inputs, associated with wires x an y respectively and goal is to compute $\{\mathbb{Z}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$, where $\mathbb{Z}^{(\beta,\gamma)}$ is the result of applying the gate function on the elements of $\mathbb{X}^{(\beta,\gamma)}$ and $\mathbb{Y}^{(\beta,\gamma)}$ component-wise. For example, if g is an addition gate then component-wise, $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} + \mathbb{Y}^{(\beta,\gamma)}$, while if g is a multiplication-gate then component-wise, $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} \times \mathbb{Y}^{(\beta,\gamma)}$. Then during the shared-circuit evaluation, the parties will hold ${}^A\langle\mathbb{X}^{(\beta,\gamma)}\rangle_d$ and ${}^A\langle\mathbb{Y}^{(\beta,\gamma)}\rangle_d$ for gate-inputs and the goal will be to let the parties securely compute ${}^A\langle\mathbb{Z}^{(\beta,\gamma)}\rangle_d$.

Looking ahead, evaluating a gate $t+1$ times ensure that we have "sufficiently many" ${}^A\langle\cdot\rangle_\ell$-shared inputs for degree-reduction when g is a multiplication-gate to maintain $\mathcal{O}(1)$ communication overhead. On the other hand, having n such groups of evaluations of gate g will ensure that we need to generate sufficiently many ${}^A\langle\cdot\rangle_\ell$-shared random multiplication triples, required for maintaining $\mathcal{O}(1)$ communication overhead. In summary, each of our vector $\mathbf{s}$ will be of size p, each $\mathbf{S}$ will consist of n^2p many $\mathbf{s}$ vectors. Then, each $\mathbb{S}$ will consist of n^3 such $\mathbf{S}$. We group $t+1$ many $\mathbb{S}$ together for degree-reduction. And in our MPC, we further need n such batches to maintain $\mathcal{O}(1)$ communication overhead. Based on this discussion, we next present the preprocessing phase.

F The Preprocessing Phase

Here we present our preprocessing protocol, which securely realizes the ideal functionality presented $\mathcal{F}_{\mathsf{Off}}$. The functionality generates a *one-time* setup, which can be used across multiple instances of our MPC protocol. Namely, for every pair of parties (P_i, P_j), the functionality sets up a twisted t-sharing $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$ with respect to P_i, where the twisted-share of P_i is set to 0. The sharing is generated as follows: if P_j is *honest*, then the functionality picks $\boldsymbol{\mu}_{ji}$ randomly and also generates $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$ randomly. On the other hand, if P_j is *corrupt*, then the functionality receives $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$ from the ideal-world adversary.

Next, the functionality generates two types of preprocessing data to be used later by our online-phase protocol. The first is ${}^A\langle\cdot\rangle_d$-sharing of random

multiplication-triples. In more detail, for every multiplication gate in `ckt`, the functionality generates $n(t+1)$ key-consistent ${}^{A}\langle\cdot\rangle_{d}$-sharing of n^3p^2 random multiplication triples and $n(t+1)$ key-consistent ${}^{A}\langle\cdot\rangle_{d}$-sharing of n^3p^2 random triples (these need not be multiplication triples). The reason for associating so many multiplication-triples and triples with every multiplication-gate is as follows. Recall from Section E that each multiplication-gate will be evaluated over n^3p^2 pairs of inputs in parallel in SIMD fashion $n(t+1)$ times to achieve $\mathcal{O}(1)$ overhead. As discussed in Section 2.3, the protocol for evaluating multiplication gates will have a *non-robust* step where the evaluation may fail if corrupt parties cheat. However, in that case, the parties can run a "fault-localization" process which will enable the honest parties to locally identify a subset of corrupt parties. The fault-localization process will require a vector of triples to be "sacrificed" in the sense that their privacy will be lost. After the fault-localization process, it would be guaranteed that the multiplication gate will be evaluated robustly.

The second type of preprocessing data is double-packed-secret-shared vectors of random values. In more detail, for every multiplication gate in `ckt`, the functionality generates $2n(t+1)$ key-consistent ${}^{A}\langle\cdot\rangle_{d}$-sharing as well as key-consistent ${}^{A}\langle\cdot\rangle_{d+p-1}$-sharing of n^3p^2 random values. The reason for associating an additional random pair with every multiplication gate is same as that for associating a sacrificing triple with every multiplication gate.

In $\mathcal{F}_{\mathsf{Off}}$, the ideal adversary specifies all the "data" that the corrupt parties would like to hold as part of the various ${}^{A}\langle\cdot\rangle$-sharings generated by the functionality. The functionality then generates the ${}^{A}\langle\cdot\rangle$-sharings while keeping them consistent with the data specified by the ideal-world adversary.

Functionality F.1: $\mathcal{F}_{\mathsf{Off}}$ – Ideal functionality for preprocessing

The functionality interacts with the parties in $\mathcal{P}$ and an adversary $\mathcal{S}$ as follows. Let $\mathcal{C}$ be the set of corrupt parties.

- **Setup Generation**: On input SetUp from all the parties in $\mathcal{P} \setminus \mathcal{C}$, do:
  - On behalf of every $P_j \in \mathcal{P} \setminus \mathcal{C}$, do the following.
    - Select n random vectors $\{\boldsymbol{\mu}_{ji}\}_{i=1}^{n}$ where each $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2p}$ and where the ith value is designated to be used in the MAC-key for party P_i.
    - For every $\boldsymbol{\mu}_{ji}$, generate a random $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ where the twisted-share of P_i is 0. Then send to every P_k its shares corresponding to $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$.
  - On behalf of each $P_i \in \mathcal{C}$, do the following.
    - Receive from $\mathcal{S}$ the n vectors $\{\boldsymbol{\mu}_{ji}\}_{i=1}^{n}$ where each $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2p}$ and where the ith value is designated to be used in the MAC-key for P_i.
    - For every $\boldsymbol{\mu}_{ji}$, receive from $\mathcal{S}$ the sharing $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ such that the twisted-share of P_i is 0. Send to every P_k its shares corresponding to $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$.
- **Random Multiplication-Triple Sharings**: On input MultTriple from all the parties in $\mathcal{P} \setminus \mathcal{C}$, generate $n(t+1)c_M$ number of ${}^{A}\langle\cdot\rangle_{d}$-sharing of random multiplication-triples. To generate one such $({}^{A}\langle\mathbb{A}\rangle_{d}, {}^{A}\langle\mathbb{B}\rangle_{d}, {}^{A}\langle\mathbb{C}\rangle_{d})$, where $\mathbb{A}, \mathbb{B}, \mathbb{C}$ are three-dimensional vectors of n^3p^2 elements, do the following.
  - For $u \in \{1, \ldots, n\}, v \in \{1, \ldots, n^2p\}$, select vectors $\mathbf{a}^{(u,v)}$ and $\mathbf{b}^{(u,v)}$, where each $\mathbf{a}^{(u,v)}$ and $\mathbf{b}^{(u,v)}$ consists of p random elements from $\mathbb{F}$.

- Compute the component-wise product $\mathbf{c}^{(u,v)} = \mathbf{a}^{(u,v)} \times \mathbf{b}^{(u,v)}$.
- Let $\mathbb{A} = (\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(n)})$ where $\mathbf{A}^{(u)} = (\mathbf{a}^{(u,1)}, \ldots, \mathbf{a}^{(u,n^2p)})$. Similarly, define $\mathbb{B}$ and $\mathbb{C}$ from $\{\mathbf{b}^{(u,v)}\}$ and $\{\mathbf{c}^{(u,v)}\}$. Run the steps "Single ${}^{A}\langle\cdot\rangle_d$-Sharing Generation" (below) to generate ${}^{A}\langle\mathbb{A}\rangle_d$, ${}^{A}\langle\mathbb{B}\rangle_d$ and ${}^{A}\langle\mathbb{C}\rangle_d$.

– **Random Triple Sharings**: On input Triple from all the parties in $\mathcal{P} \setminus \mathcal{C}$, generate $n(t+1)c_M$ number of ${}^{A}\langle\cdot\rangle_d$-sharings, each consisting of n^3p^2 random triples. To generate one such $({}^{A}\langle\hat{\mathbb{A}}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}\rangle_d)$, do the following.
  - For $u \in \{1, \ldots, n\}, v \in \{1, \ldots, n^2\}$, randomly select vectors $\hat{\mathbf{a}}^{(u,v)}, \hat{\mathbf{b}}^{(u,v)}$ and $\hat{\mathbf{c}}^{(u,v)}$, each consisting of p random elements from $\mathbb{F}$.
  - Let $\hat{\mathbb{A}} = (\hat{\mathbf{A}}^{(1)}, \ldots, \hat{\mathbf{A}}^{(n)})$ where each $\hat{\mathbf{A}}^{(u)} = (\hat{\mathbf{a}}^{(u,1)}, \ldots, \hat{\mathbf{a}}^{(u,n^2p)})$. Similarly, define $\hat{\mathbb{B}}$ and $\hat{\mathbb{C}}$ from $\{\hat{\mathbf{b}}^{(u,v)}\}$ and $\{\hat{\mathbf{c}}^{(u,v)}\}$. Run the steps "Single ${}^{A}\langle\cdot\rangle_d$-Sharing Generation" (below) to generate ${}^{A}\langle\hat{\mathbb{A}}\rangle_d$, ${}^{A}\langle\hat{\mathbb{B}}\rangle_d$ and ${}^{A}\langle\hat{\mathbb{C}}\rangle_d$.

– **Random Double-Sharings**: On input Double from the parties in $\mathcal{P} \setminus \mathcal{C}$, generate $2n(t+1)c_M$ ${}^{A}\langle\cdot\rangle_d$-sharing and ${}^{A}\langle\cdot\rangle_{d+p-1}$-sharing of n^3p^2 random elements. To generate one such $({}^{A}\langle\mathbb{R}\rangle_d, {}^{A}\langle\mathbb{R}\rangle_{d+p-1})$, do the following.
  - For $u \in \{1, \ldots, n\}, v \in \{1, \ldots, n^2p\}$, randomly select vectors $\mathbf{r}^{(u,v)}$, each consisting of p random elements from $\mathbb{F}$.
  - Let $\mathbb{R} = (\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(n)})$ where each $\mathbf{R}^{(u)} = (\mathbf{r}^{(u,1)}, \ldots, \mathbf{r}^{(u,n^2p)})$. Run the steps "Single ${}^{A}\langle\cdot\rangle_d$-Sharing Generation" and "Single ${}^{A}\langle\cdot\rangle_{d+p-1}$-Sharing Generation" (see below) to generate ${}^{A}\langle\mathbb{R}\rangle_d$ and ${}^{A}\langle\mathbb{R}\rangle_{d+p-1}$ respectively.

Single ${}^{A}\langle\cdot\rangle_d$-Sharing Generation: $\mathcal{F}_{\mathsf{Off}}$ does the following to generate ${}^{A}\langle\mathbb{S}\rangle_d$ for a given set of vectors $\{\mathbf{s}^{(u,v)}\}_{u\in\{1,\ldots,n\},v\in\{1,\ldots,n^2p\}}$, where each $\mathbf{s}^{(u,v)} \in \mathbb{F}^p$.

– Let $\mathbb{S} = (\mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(n)})$ where each $\mathbf{S}^{(u)} = (\mathbf{s}^{(u,1)}, \ldots, \mathbf{s}^{(u,n^2p)})$. Let $\mathbf{s}^{(u,v)} = (s^{(u,v,1)}, \ldots, s^{(u,v,p)})$. For each $\mathbf{s}^{(u,v)}$, on receiving the shares $\{s_i^{(u,v)}\}_{P_i\in\mathcal{C}}$ from $\mathcal{S}$ on behalf of the corrupt parties, compute $\langle\mathbf{s}^{(u,v)}\rangle_d$ as follows.
  - Select a random d-degree polynomial $S^{(u,v)}(\cdot)$, where: (a) $S^{(u,v)}(i) = s_i^{(u,v)}, \forall P_i \in \mathcal{C}$; (b) $S^{(u,v)}(-1) = s^{(u,v,-1)}, \ldots, S^{(u,v)}(-p) = s^{(u,v,p)}$.
  - For every $P_i \notin \mathcal{C}$, compute $s_i^{(u,v)} = S^{(u,v)}(i)$.
– Corresponding to each $\mathbf{S}^{(u)}$, generate ${}^{A}\langle\mathbf{S}^{(u)}\rangle_d$ as follows which completes the generation of ${}^{A}\langle\mathbb{S}\rangle_d$.
  - On receiving $\{v_{ji}^{(u)}\}_{P_j\in\mathcal{C},P_i\notin\mathcal{C}}$ from $\mathcal{S}$, the second components of the MAC-key that a corrupt P_j will have for an honest P_i, set $\mathsf{K}_{ji}^{(u)} = (\boldsymbol{\mu}_{ji}, v_{ji}^{(u)})$, where $\boldsymbol{\mu}_{ji}$ was specified by $\mathcal{S}$ in "Setup Generation" stage.
  - For every $P_j \notin \mathcal{C}$ and every $P_i \in \mathcal{P}$, randomly pick $v_{ji}^{(u)} \in \mathbb{F}$ and set $\mathsf{K}_{ji}^{(u)} = (\boldsymbol{\mu}_{ji}, v_{ji}^{(u)})$, where $\boldsymbol{\mu}_{ji}$ is taken from the "Setup Generation" stage.
  - For every $P_i \in \mathcal{P}$, let $\mathbf{s}_i^{(u)}$ be the vector of shares of P_i corresponding to $\langle\mathbf{S}^{(u)}\rangle_d$. That is, $\mathbf{s}_i^{(u)} = \{s_i^{(u,v)}\}_{v\in\{1,\ldots,n^2p\}}$. For every $P_j \in \mathcal{P}$, compute the MAC-tag $\tau_{ij}^{(u)} = \mathsf{MAC}_{\mathsf{K}_{ji}^{(u)}}(\mathbf{s}_i^{(u)})$.
  - For each $P_i \in \mathcal{P}$, send $\{\mathbf{s}_i^{(u)}, \{\tau_{ij}^{(u)}\}_{j=1}^n, \{\mathsf{K}_{ij}^{(u)}\}_{j=1}^n\}$ to P_i.

Single $^{A}\langle\cdot\rangle_{d+p-1}$-Sharing Generation: The steps here are almost the same as for "**Single $^{A}\langle\cdot\rangle_{d}$-Sharing Generation**:" except that for each vector of values, functionality generate a $(d+p-1)$-packed-secret-sharing.

To securely realize the functionality $\mathcal{F}_{\mathsf{Off}}$, we present various subprotocols.

F.1 Functionality for a One-Time Setup for MAC-Keys

Functionality $\mathcal{F}_{\mathsf{MacKeys}}$ generates a *one-time* setup of twisted t-sharing for the $\boldsymbol{\mu}$ component of the MAC-key across every pair of parties.

Functionality F.2: $\mathcal{F}_{\mathsf{MacKeys}}$ — set up for consistent MAC-keys

The functionality is parametrized by the set $\mathcal{C}$ of corrupt parties. The functionality interacts with the parties in $\mathcal{P}$ and an adversary $\mathcal{S}$ as follows. On receiving the input SetUp from all the parties in $\mathcal{P} \setminus \mathcal{C}$, do the following.

- On behalf of every $P_j \in \mathcal{P} \setminus \mathcal{C}$, do the following.
  - Select n random vectors $\{\boldsymbol{\mu}_{ji}\}_{i=1}^{n}$ where each $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2p}$ and where the ith value is designated to be used in the MAC-key for party P_i.
  - Corresponding to every $\boldsymbol{\mu}_{ji}$, generate a random $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ where the twisted-share of P_i is 0. For this, randomly pick n^2p random t-degree polynomials which evaluate to the elements of $\boldsymbol{\mu}_{ji}$ at i and which evaluate to 0 at 0. Then send to every P_k its shares corresponding to $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$.
- On behalf of each $P_i \in \mathcal{C}$, do the following.
  - Receive from $\mathcal{S}$ the n vectors $\{\boldsymbol{\mu}_{ji}\}_{i=1}^{n}$ where each $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2p}$ and where the ith value is designated to be used in the MAC-key for P_i.
  - For every $\boldsymbol{\mu}_{ji}$, receive from $\mathcal{S}$ the sharing $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ such that the twisted-share of P_i is 0. Then send to every P_k its shares corresponding to $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$. If $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ is not a valid twisted t-sharing with twisted-shares of P_i being not 0, then send 0s as the shares to every P_k.

To securely realize $\mathcal{F}_{\mathsf{MacKeys}}$, the parties can run the protocol Π_{MacKeys}. For every pair of parties (P_i, P_j), party P_j picks a random vector $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2p}$. It then generates twisted t-sharing of these values with respect to P_i, by embedding these values in random t-degree polynomials at i, such that P_i's twisted-shares are set to 0. The generation of twisted sharing is done verifiably via $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$. For the dealers who are identified to be corrupt (by sending incorrect polynomials in their respective $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ instances), the parties pick 0 as the default shares.

Protocol F.3: Π_{MacKeys} — Securely ealizing $\mathcal{F}_{\mathsf{MacKeys}}$ in the $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$-hybrid

- Every party $P_j \in \mathcal{P}$ does the following for every $P_i \in \mathcal{P}$.
  - Randomly pick $\boldsymbol{\mu}_{ji} = (u_{ji}^{(1)}, \ldots, u_{ji}^{(n^2p)}) \in \mathbb{F}^{n^2p}$.
  - For every $k \in \{1, \ldots, n^2p\}$, pick a random t-degree polynomial $q_{ji}^{(k)}(\cdot)$, such that $q_{ji}^{(k)}(i) = u_{ji}^{(k)}$ and $q_{ji}^{(k)}(0) = 0$.

  - Send $\mathcal{T}_{ji} = \{P_i\}$ and $q_{ji}^{(1)}(\cdot), \ldots, q_{ji}^{(n^2p)}(\cdot)$ to $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$.
- Every party $P_k \in \mathcal{P}$ computes its output as follows.
  - Include P_j in a set Cr, if there exists any $i \in \{1, \ldots, n\}$, such that no output is obtained from the $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ instance where P_j is the dealer and P_i is the party with respect to whom P_j has invoked the $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ instance.
  - For every $P_j \in \mathsf{Cr}$ and every P_i, set $\boldsymbol{\mu}_{ji}$ to $(0, \ldots, 0$ (n^2p times)).
  - For every (P_i, P_j), output the shares corresponding to $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$ as follows.
    * If $P_j \in \mathsf{Cr}$, then set $(0, \ldots, 0$ (n^2p times)) as the share for $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$.
    * Otherwise: (a) If $P_i = P_k$, then set $(0, \ldots, 0$ (n^2(p times))) as the share for $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$. (b) Else set $(q_{ji}^{(1)}(k), \ldots, q_{ji}^{(n^2p)}(k))$ as the share for $\lceil\boldsymbol{\mu}_{ji}\rfloor_t^i$, where $q_{ji}^{(1)}(k), \ldots, q_{ji}^{(n^2p)}(k)$ is received from the $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ instance invoked by P_j as the dealer, corresponding to P_i.

Lemma F.4. *Protocol Π_{MacKeys} securely realizes $\mathcal{F}_{\mathsf{MacKeys}}$ in $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$-hybrid model with perfect security. The protocol calls $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ n^2 times, each with n^2p inputs.*

Proof. The number of calls to $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ is straightforward. The security of the protocol simply follows from the fact that all $\boldsymbol{\mu}$ vectors are twisted t-shared through calls to $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$. We formalize this by giving a simulator $\mathcal{S}_{\mathsf{MacKeys}}$ (Simulator F.5).

Simulator F.5: $\mathcal{S}_{\mathsf{MacKeys}}$: Simulator for the Protocol Π_{MacKeys}

$\mathcal{S}_{\mathsf{Off}}$ constructs virtual real-world honest parties and invokes the real-world adversary $\mathcal{A}$ who controls at most t parties. The simulator simulates the view of $\mathcal{A}$, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with $\mathcal{F}_{\mathsf{MacKeys}}$. In order to simulate $\mathcal{Z}$, the simulator forwards every message it receives from $\mathcal{Z}$ to $\mathcal{A}$ and vice-versa. The simulator then simulates the various steps of the protocol as follows.

- On behalf of every party P_j not under the control of $\mathcal{A}$ and every $P_i \in \mathcal{P}$, the simulator does the following.
  - Randomly pick $\boldsymbol{\mu}_{ji} = (u_{ji}^{(1)}, \ldots, u_{ji}^{(n^2p)}) \in \mathbb{F}^{n^2p}$.
  - For every $k \in \{1, \ldots, n^2p\}$, pick a random t-degree polynomial $q_{ji}^{(k)}(\cdot)$, such that $q_{ji}^{(k)}(i) = u_{ji}^{(k)}$ and $q_{ji}^{(k)}(0) = 0$.
  - For every party P_k under the control of $\mathcal{A}$, send the following to $\mathcal{A}$ on behalf of the instance of $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$ called by P_j corresponding to P_i:
    * $(0, \ldots, 0$ (n^2p times)), if $P_i = P_k$;
    * $q_{ji}^{(1)}(k), \ldots, q_{ji}^{(n^2p)}(k)$, otherwise.
- On behalf of every P_j under the control of $\mathcal{A}$, the simulator does:
  - Let $q_{ji}^{(1)}(\cdot), \ldots, q_{ji}^{(n^2p)}(\cdot)$ be the polynomials which P_j wants to send to $\mathcal{F}_{\lceil\cdot\rfloor_t^i}$, corresponding to P_i. For each $k \in \{1, \ldots, n^2p\}$, let $u_{ji}^{(k)} = q_{ji}^{(k)}(i)$ and let $\boldsymbol{\mu}_{ji} = (u_{ji}^{(1)}, \ldots, u_{ji}^{(n^2p)})$.
  - Send $\{\boldsymbol{\mu}_{ji}\}_{i=1}^n$ to $\mathcal{F}_{\mathsf{MacKeys}}$ on behalf of P_j.

- For every $\boldsymbol{\mu}_{ji}$, send the vectors $\{q_{ji}^{(k)}(1), \ldots, q_{ji}^{(k)}(i-1), q_{ji}^{(k)}(0), q_{ji}^{(k)}(i+1), \ldots, q_{ji}^{(k)}(n)\}_{k \in \{1,\ldots,n^2p\}}$ to $\mathcal{F}_{\mathsf{MacKeys}}$ on behalf of P_j

---

We next prove a sequence of claims, which help us to prove the lemma.

Claim F.6. *The view of $\mathcal{A}$ in the simulated execution with $\mathcal{S}_{\mathsf{MacKeys}}$ is identically distributed as the view of $\mathcal{A}$ in the real execution of Π_{MacKeys}.*

Proof. In the real execution of Π_{MacKeys}, for every honest P_j, $\mathcal{A}$ receives up to t twisted-shares from $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$ for a random $\boldsymbol{\mu}_{ji}$, corresponding to every $P_i \in \mathcal{P}$. On the other hand, during the simulated execution, the simulator provides t random shares for a random $\boldsymbol{\mu}_{ji}$ selected by the simulator to $\mathcal{A}$. It is easy to see that these twisted-shares are identically distributed. □

Claim F.7. *Conditioned on the view of $\mathcal{A}$, the output of the honest parties is identically distributed in the real execution of Π_{MacKeys} involving $\mathcal{A}$, as well as in the ideal execution involving $\mathcal{S}_{\mathsf{MacKeys}}$ and $\mathcal{F}_{\mathsf{MacKeys}}$.*

Proof. Let $\mathcal{H}$ be the set of honest parties. During the real execution, the output of the honest parties consist of the following.

- Corresponding to every $P_j \in \mathcal{H}$, the twisted-shares for random $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$, lying on random degree-t sharing polynomials.
- Corresponding to every $P_j \notin \mathcal{H}$, the twisted-shares for $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$, provided P_j selected and sent degree-t sharing polynomials for them to $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$. Otherwise the twisted-shares are 0s, implying that $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$ are all 0s.

Now consider the simulated execution. It then follows that corresponding to every $P_j \in \mathcal{H}$, the honest parties receive twisted-shares for random $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$, lying on random degree-t sharing polynomials, selected by $\mathcal{F}_{\mathsf{MacKeys}}$. So they are identically distributed in both the executions. On the other hand, for every $P_j \notin \mathcal{H}$, the simulator will learn the entire underlying sharing-polynomials that P_j would like to send to instances of $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$ for computing twisted-shares for $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$. Based on these polynomials, the simulator computes the underlying $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$ and the corresponding vectors of shares and forwards them to $\mathcal{F}_{\mathsf{MacKeys}}$. If these vectors of shares define twisted t-sharings of $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$, then $\mathcal{F}_{\mathsf{MacKeys}}$ will distrubute the shares and thus the output of the honest parties will be identically distributed in both the executions. On ther other hand, if the vectors of shares do not define twisted t-sharings of $\boldsymbol{\mu}_{j1}, \ldots, \boldsymbol{\mu}_{jn}$, then $\mathcal{F}_{\mathsf{MacKeys}}$ will distribute 0s as twisted-shares. So even in this case, the outputs of the honest parties are identically distributed in both the executions. □

The lemma now follows from the previous two claims. □

Recall that one call to $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$ with N polynomials can be securely realized with statistical security in a constant number of rounds and communication of $\mathsf{P2P}(N \cdot n\kappa + n^2\kappa) + n \times \mathsf{BC}(n^2\kappa)$ bits. In the protocol, there are n^2 calls to $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$,

each with $N = n^2(\frac{t}{4}) = \mathcal{O}(n^3)$ polynomials. This will incur a communication of $\mathsf{P2P}(n^6\kappa) + n \times \mathsf{BC}(n^4\kappa)$ bits, as each party can simultaneously broadcast all its data across n^2 calls to $\mathcal{F}_{\lceil \cdot \rfloor_t^i}$. Securely realizing the parallel broadcast using the randomized protocol of [3] will require a constant expected number of rounds and a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. Hence we get the following Lemma.

Lemma F.8. *If $t < \frac{n}{3}$, then there exists a protocol for securely realizing $\mathcal{F}_{\mathsf{MacKeys}}$ with statistical security in the plain model over point-to-point channels. The protocol requires a constant expected number of rounds and incurs a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits.*

F.2 Robust Degree Reduction Protocol with a Constant Overhead

Protocol Π_{DegRed} takes input $t+1$ number of ${}^{A}\langle\cdot\rangle_d$-shared inputs $\mathbb{S}_1, \ldots, \mathbb{S}_{t+1}$, each consisting of $n^3p^2 = \Theta(n^5)$ elements from $\mathbb{F}$. The protocol outputs ${}^{A}\langle\cdot\rangle_{p-1}$-sharing of $\mathbb{S}_1, \ldots, \mathbb{S}_{t+1}$. Note that the privacy of $\mathbb{S}_1, \ldots, \mathbb{S}_{t+1}$ will not be guaranteed and adversary will completely know these values, however that is fine for our purpose. The protocol is adapted from [23] and its high level overview has been already outlined in Section 2.3 for one-dimensional vectors. Here we extend it for ${}^{A}\langle\cdot\rangle$.-sharings of three-dimensional vectors of values.

Protocol F.9: Π_{RoDegRed} **– Robust Degree-Reduction Protocol**

Input: Parties hold inputs ${}^{A}\langle\mathbb{S}^{(1)}\rangle_d, \ldots, {}^{A}\langle\mathbb{S}^{(t+1)}\rangle_d$, where each $\mathbb{S}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$.

1. Parties define a vector of n^3p^2 degree-t polynomials $\mathbb{S}(X) = \mathbb{S}^{(1)} + \mathbb{S}^{(2)} \cdot X + \ldots + \mathbb{S}^{(t+1)} \cdot X^t$.
2. For $i \in \{1, \ldots, n\}$, parties locally compute ${}^{A}\langle\mathbb{S}(i)\rangle_d = {}^{A}\langle\mathbb{S}^{(1)}\rangle_d + i \cdot {}^{A}\langle\mathbb{S}^{(2)}\rangle_d + \ldots + i^t \cdot {}^{A}\langle\mathbb{S}^{(t+1)}\rangle_d$ and reconstruct $\mathbb{S}(i)$ towards P_i.
3. For each $i \in \{1, \ldots, n\}$, party P_i generates ${}^{A}\langle\mathbb{S}(i)\rangle_{p-1}$ by acting as a dealer and invoking an instance of protocol $\Pi_{{}^{A}\langle\cdot\rangle}$ (Protocol 5.1).
4. Every party $P_i \in \mathcal{P}$ applies the RS-decoding procedure on their shares (points on the polynomials) corresponding to ${}^{A}\langle\mathbb{S}(1)\rangle_{p-1}, \ldots, {}^{A}\langle\mathbb{S}(n)\rangle_{p-1}$ to error-correct t errors. This local error-correction procedure allows P_i to identify at most t corrupt positions, say, w.l.o.g., they are $\{1, \ldots, t\}$. P_i now linearly combine (where the linear combination is as per Lagrange's interpolation) its shares of ${}^{A}\langle\mathbb{S}(t+1)\rangle_{p-1}, \ldots, {}^{A}\langle\mathbb{S}(n)\rangle_{p-1}$ to get the shares (including the MAC-tags) corresponding to ${}^{A}\langle\mathbb{S}^{(1)}\rangle_{p-1}, \ldots, {}^{A}\langle\mathbb{S}^{(t+1)}\rangle_{p-1}$. It further broadcasts the corrupt set, which is $\{1, \ldots, t\}$ in this case for simplicity, to all so that every verifier P_j can adjust the verification information corresponding to the MAC-tags of P_i. The broadcast is for a n-length bit vector, where kth bit being 0 denotes P_i found P_k to be corrupt and P_k's resharing is excluded for the linear combination.

Lemma F.10 (Correctness). *Given inputs* ${}^{A}\langle \mathbb{S}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_d$*, where each* $\mathbb{S}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$*, protocol* Π_{RoDegRed} *outputs* ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$*, except with a negligible probability.*

Proof. Let $\mathbb{S}(X)$ be the vector of degree-t polynomials where $\mathbb{S}(X) = \mathbb{S}^{(1)} + \mathbb{S}^{(2)} \cdot X + \ldots + \mathbb{S}^{(t+1)} \cdot X^t$. Since the parties hold ${}^{A}\langle \mathbb{S}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_d$, from the linearity property of ${}^{A}\langle \cdot \rangle_d$-sharing it follows that at the end of step 2, parties will hold ${}^{A}\langle \mathbb{S}(1) \rangle_d, \ldots, {}^{A}\langle \mathbb{S}(n) \rangle_d$. Note that $\mathbb{S}(1), \ldots, \mathbb{S}(n)$ supposedly constitute vectors of n distinct points on the vectors of degree-t polynomials $\mathbb{S}(X)$. During step 3, each P_i will reconstruct $\mathbb{S}(i)$, except with a negligible probability, which follows from the reconstruction protocol for reconstructing ${}^{A}\langle \cdot \rangle_d$-sharing. During step 4, each *honest* P_i will generate ${}^{A}\langle \mathbb{S}(i) \rangle_{p-1}$, except with a negligible probability, which follows from the correctness property of $\Pi_{{}^{A}\langle \cdot \rangle}$ (Lemma D.5). Note that a *corrupt* P_i may generate ${}^{A}\langle \cdot \rangle_d$-sharing of an *incorrect* $\mathbb{S}(i)$. However, the commitment property of $\Pi_{{}^{A}\langle \cdot \rangle}$ guarantees that on behalf of P_i, there will be some ${}^{A}\langle \cdot \rangle_d$-sharing generated at the end of step 4. As observed in [23], the shares (points of the underlying polynomials) of each party corresponding to ${}^{A}\langle \cdot \rangle_{p-1}$-sharing of $\mathbb{S}(1), \ldots, \mathbb{S}(n)$ themselves constitute codewords of length-n, of which up to t may be erroneous due to the corrupt parties. Thus, each P_i can locally identify the error positions using RS-error correction procedure. For simplicity let the error positions be $\{1, \ldots, t\}$. It can now linearly combine its shares and MAC-tags of ${}^{A}\langle \mathbb{S}(t+1) \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}(n) \rangle_{p-1}$ to obtain its correct shares corresponding to ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$. The linear combination is as per Lagrange's interpolation. To allow P_j to adjust its verification information for MACs, P_i broadcasts an n-length bit vector indicating whose resharing is used. Different P_i may use different set of resharings. But the correctness of two honest parties share of ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$ holds due to correctness of RS error-correction, Lagrange's interpolation and linearity of MACs. □

Lemma F.11 (Complexity). *Given inputs* ${}^{A}\langle \mathbb{S}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_d$*, where each* $\mathbb{S}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$*, protocol* Π_{RoDegRed} *requires a constant expected number of rounds. The protocol incurs a communication of* $\mathsf{P2P}(n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits and calls* $\mathcal{F}_{\mathsf{Rand}}$ *at most 15 times.*

If Π_{RoDegRed} *is executed* L *times in parallel, then it incurs a constant expected number of rounds, communication of* $\mathsf{P2P}(L \cdot n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits and at most* 15 *calls to* $\mathcal{F}_{\mathsf{Rand}}$*.*

Proof. Step 1 and 2 requires the parties to perform local computation. In step 3, for every $i \in \{1, \ldots, n\}$, $\mathbb{S}(i)$ consisting of n^3p^2 elements are reconstructed towards P_i. This requires one round and incurs a *total* communication of $\mathsf{P2P}(n^4p^2)$ elements. In step 4, for every $i \in \{1, \ldots, n\}$, party P_i invokes an instance of $\Pi_{{}^{A}\langle \cdot \rangle}$ to secret-share n^3p^2 elements. From Lemma D.7, after securely realizing the broadcasts over point-to-point channels, this requires a constant expected number of rounds and a *total* communication of $\mathsf{P2P}(n^4p^2)$ elements, plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. The calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across all the dealers and all the instances of $\Pi_{{}^{A}\langle \cdot \rangle}$ invoked in the protocol and hence the maximum

number of calls to $\mathcal{F}_{\mathsf{Rand}}$ will be 15. Step 5 requires incurs a communication of $n \times \mathsf{BC}(n)$ bits.

If the protocol is executed L times in parallel, then the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across all the dealers and so the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ remains at most 15. Moreover, the broadcast complexity will be $n \times \mathsf{BC}(nL)$. Securely realizing the parallel broadcast via the protocol of [3] will require a constant expected number of rounds, communication of $\mathsf{P2P}(n^2L)$ bits plus an expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. The lemma now follows since $p = \Theta(n)$ and each element from $\mathbb{F}$ is represented by $\mathcal{O}(\kappa)$ bits. □

Communication Overhead of the Protocol Π_{RoDegRed}. If Π_{RoDegRed} is executed L times in parallel, then the output consists of APSS of $L \cdot n^3p^2(t+1) = \Theta(L \cdot n^6)$ elements. Since the communication complexity of the protocol is $PP(L \cdot n^6)$ elements, apart from 15 calls to $\mathcal{F}_{\mathsf{Rand}}$, it follows that for sufficiently large L, the asymptotic communication overhead of the protocol is $\mathcal{O}(1)$.

Although we do not model the requirements of Π_{RoDegRed} through a functionality, we describe the simulation steps for Π_{RoDegRed} which will be used when the protocol is used as a sub-protocol.

Simulator F.12: Simulator for the Protocol Π_{RoDegRed}

- On behalf of each honest P_i, perform the local computations as an honest party would as per the protocol specifications and for each P_k under the control of $\mathcal{A}$, send the honest parties' shares of ${}^{A}\langle \mathbb{S}(k) \rangle_d$ to $\mathcal{A}$.
- Receive from the adversary the shares of corrupt parties corresponding to ${}^{A}\langle \mathbb{S}(i) \rangle_d$ on behalf of each honest party P_i not under the control of $\mathcal{A}$.
- On behalf of each honest P_i, simulate the steps of VSS for an honest dealer as per Simulator D.8 with P_i as the dealer and simulate the steps of VSS for a corrupt dealer as per Simulator D.8 for each dealer P_k under $\mathcal{A}$'s control.
- Perform local operations on behalf of the honest parties as per the protocol.

F.3 Generating Double-Secret-Shared Random Values

Protocol Π_{RandPair} outputs APSS of the form $({}^{A}\langle \widetilde{\mathbb{R}} \rangle_d, {}^{A}\langle \widetilde{\mathbb{R}} \rangle_{d+p-1})$, with $\mathcal{O}(1)$ overhead, where each $\widetilde{\mathbb{R}} \in \mathbb{F}^{n^3p^2}$ and remains random for the adversary.

The idea behind the protocol is standard. Every P_i picks a "sufficiently large" random groups elements $\mathbb{R}_i$ and generates ${}^{A}\langle \cdot \rangle_d$ as well as ${}^{A}\langle \cdot \rangle_{d+p-1}$-sharings of these elements through $\Pi_{{}^{A}\langle \cdot \rangle}$. To verify if a *corrupt* P_i generates ${}^{A}\langle \cdot \rangle_d$-sharings and ${}^{A}\langle \cdot \rangle_{d+p-1}$-sharings of *different* elements, we use the standard "sacrificing" trick and publicly verify a random linear combination of the various ${}^{A}\langle \cdot \rangle_d$-sharings and the various ${}^{A}\langle \cdot \rangle_{d+p-1}$-sharings generated by P_i. Once it is confirmed that every party has secret-shared the same random pairs, the parties locally extract secret-shared random pairs by deploying the standard Vandermonde-matrix based randomness extraction [20,6] technique.

Protocol F.13: Generating Secret-Shared Random Pairs – Π_{RandPair}

1. Each party $P_i \in \mathcal{P}$ randomly selects $\mathbf{R}_1^{(1)}, \ldots, \mathbf{R}_i^{(n+1)} \in \mathbb{F}^{n^2p^2}$ and set $\mathbb{R}_i = (\mathbf{R}_1^{(1)}, \ldots, \mathbf{R}^{(n+1)})$.
2. Each $P_i \in \mathcal{P}$ acts a dealer and invokes instances of $\Pi_{A\langle\cdot\rangle}$ (Protocol 5.1) to generate ${}^A\langle\mathbb{R}_i\rangle_d$ and ${}^A\langle\mathbb{R}_i\rangle_{d+p-1}$. %. Recall that ${}^A\langle\mathbb{R}_i\rangle_\ell = ({}^A\langle\mathbf{R}^1\rangle_\ell, \ldots, {}^A\langle\mathbf{R}^{n+1}\rangle_\ell)$.
3. Every $P_i \in \mathcal{P}$ sends Rand to $\mathcal{F}_{\mathsf{Rand}}$ and receives r from $\mathcal{F}_{\mathsf{Rand}}$.
4. For each $P_i \in \mathcal{P}$, parties locally compute
   - ${}^A\langle\hat{\mathbf{R}}_i\rangle_d = {}^A\langle\mathbf{R}_i^{(1)}\rangle_d + r \cdot {}^A\langle\mathbf{R}_i^{(2)}\rangle_d + \ldots + r^n \cdot {}^A\langle\mathbf{R}_i^{(n+1)}\rangle_d$
   - ${}^A\langle\widetilde{\mathbf{R}}_i\rangle_{d+p-1} = {}^A\langle\mathbf{R}_i^{(1)}\rangle_{d+p-1} + r \cdot {}^A\langle\mathbf{R}_i^{(2)}\rangle_{d+p-1} + \ldots + r^n \cdot {}^A\langle\mathbf{R}_i^{(n+1)}\rangle_{d+p-1}$
5. Parties publicly reconstruct $\hat{\mathbf{R}}_i, \widetilde{\mathbf{R}}_i$ for each $P_i \in \mathcal{P}$.
6. For each P_i where $\hat{\mathbf{R}}_i \neq \widetilde{\mathbf{R}}_i$, parties reset $\mathbb{R}_i = 0^{n^3p^2}$ and takes the default sharing of ${}^A\langle\mathbb{R}_i\rangle_d$ and ${}^A\langle\mathbb{R}_i\rangle_{d+p-1}$. on behalf of P_i.
7. For each P_i where $\hat{\mathbf{R}}_i = \widetilde{\mathbf{R}}_i$, parties reset ${}^A\langle\mathbb{R}_i\rangle_d = ({}^A\langle\mathbf{R}^1\rangle_\ell, \ldots, {}^A\langle\mathbf{R}^n\rangle_\ell)$ and ${}^A\langle\mathbb{R}_i\rangle_{d+p-1} = ({}^A\langle\mathbf{R}^1\rangle_{\ell+p-1}, \ldots, {}^A\langle\mathbf{R}^n\rangle_{\ell+p-1})$. %. After reset, for each P_i, the group $\mathbb{R}_i \in \mathbb{F}^{n^3p^2}$.
8. Let $\mathbb{R}(X)$ be the vectors of n^3p^2 degree-$(n-1)$ polynomials passing through the vectors of points $(1, \mathbb{R}_1), \ldots, (n, \mathbb{R}_n)$. Moreover, for $m \in \{1, \ldots, n-t\}$, let $\widetilde{\mathbb{R}}^{(m)} = \mathbb{R}(n+m)$, where $\widetilde{\mathbb{R}}^{(m)} = c_{m,1} \cdot \mathbb{R}_1 + \ldots + c_{m,n} \cdot \mathbb{R}_n$. Here $c_{m,1}, \ldots, c_{m,n}$ are publicly-known Lagrange's interpolation coefficients from $\mathbb{F}$.
9. Parties locally compute and output the following
   - ${}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_d = c_{m,1} \cdot {}^A\langle\mathbb{R}_1\rangle_d + \ldots + c_{m,n} \cdot {}^A\langle\mathbb{R}_n\rangle_d$.
   - ${}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_{d+p-1} = c_{m,1} \cdot {}^A\langle\mathbb{R}_1\rangle_{d+p-1} + \ldots + c_{m,n} \cdot {}^A\langle\mathbb{R}_n\rangle_{d+p-1}$.

Lemma F.14 (Correctness). *For $m = 1, \ldots, n-t$, protocol Π_{RandPair} outputs $\{({}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_d, {}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_{d+p-1})\}$ where $\widetilde{\mathbb{R}}^{(m)} \in \mathbb{F}^{n^3p^2}$.*

Proof. From the correctness and commitment properties of $\Pi_{A\langle\cdot\rangle}$ (Lemma D.5 and Lemma D.6), at the end of step 2, parties will hold $n+1$ number of ${}^A\langle\cdot\rangle_d$-sharings and $n+1$ number of ${}^A\langle\cdot\rangle_{d+p-1}$-sharings of n^2p^2 elements each, on behalf of each party P_i. Moreover, except with a negligible probability, parties will robustly reconstruct $\hat{\mathbf{R}}_i$ and $\widetilde{\mathbf{R}}_i$ during step 5. We claim that if P_i is *corrupt* and generated ${}^A\langle\mathbf{R}_i^{(l)}\rangle_d$ and ${}^A\langle\mathbf{R}_i'^{(l)}\rangle_{d+p-1}$ for any $l \in \{1, \ldots, n+1\}$, then except with a negligible probability, $\hat{\mathbf{R}}_i \neq \widetilde{\mathbf{R}}_i$ will hold and hence parties will take default ${}^A\langle\cdot\rangle_d$ and ${}^A\langle\cdot\rangle_{d+p-1}$-sharings of 0's on behalf of P_i. This simply follows from the fact that r will be random for P_i when ${}^A\langle\mathbf{R}_i^{(l)}\rangle_d$ and ${}^A\langle\mathbf{R}_i'^{(l)}\rangle_{d+p-1}$ are generated. Hence it is guaranteed that at the end of step 6, for every party P_i, parties hold ${}^A\langle\cdot\rangle_d$-sharings and ${}^A\langle\cdot\rangle_{d+p-1}$-sharings of same values. The lemma now simply follows from the protocol steps. □

Lemma F.15 (Privacy). *In Π_{RandPair}, the view of the adversary will be independent of $\{({}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_d, {}^A\langle\widetilde{\mathbb{R}}^{(m)}\rangle_{d+p-1})\}$, where each $\widetilde{\mathbb{R}}^{(m)} \in \mathbb{F}^{n^3p^2}$.*

Proof. Let $\mathcal{H}$. For simplicity, let $|\mathcal{H}| = n - t$. From the privacy property of $\Pi_{A\langle\cdot\rangle}$ (Lemma D.3), corresponding to every $P_i \in \mathcal{H}$, the adversary's view remain independent of $\mathbf{R}_i^{(1)}, \ldots, \mathbf{R}_i^{(n+1)}$ at the end of step 2. During step 4, for

every $P_i \in \mathcal{H}$, adversary learns a linear combination of $\mathbf{R}_i^{(1)}, \ldots, \mathbf{R}_i^{(n+1)}$. Since $\mathbf{R}_i^{(1)}, \ldots, \mathbf{R}_i^{(n+1)}$ are randomly chosen by P_i, it follows that even after learning this linear combination, the view of the adversary remains independent of $\mathbf{R}_i^{(1)}, \ldots, \mathbf{R}_i^{(n)}$ and hence the redefined $\mathbb{R}_i = (\mathbf{R}_i^{(1)}, \ldots, \mathbf{R}_i^{(n)})$.

Next consider the vectors of degree-$(n-1)$ polynomials $\mathbb{R}(X)$, passing through the vectors of points $(1, \mathbb{R}_1), \ldots, (n, \mathbb{R}_n)$. Among these points, the points $\{\mathbb{R}_i\}_{P_i \in \mathcal{H}}$ are random for the adversary and there are $n - t$ such vectors of points. Hence it follows that the vectors of points $\mathbb{R}(n+1), \ldots, \mathbb{R}(n+n-t)$ remain random for the adversary. Namely, for every candidate $\{\mathbb{R}_i\}_{P_i \in \mathcal{H}}$, there exists vectors of degree-$(n-1)$ polynomials passing through those candidate points and the vectors of points $\{\mathbb{R}_i\}_{P_i \notin \mathcal{H}}$. Consequently, $\widetilde{\mathbb{R}}^{(1)}, \ldots, \widetilde{\mathbb{R}}^{(n-t)}$ remains random for the adversary. □

Lemma F.16 (Complexity). *Protocol* Π_{RandPair} *outputs* $(n-t)n^3p^2$ *double-packed-secret-shared elements, requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits and makes at most* 16 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

If Π_{RandPair} *is executed* L *times in parallel, then it outputs* $L \cdot (n-t)n^3p^2$ *double-packed-secret-shared values, requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(L \cdot n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits and makes at most* 16 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

Proof. From Lemma F.14, Π_{RandPair} outputs $\{({}^{A}\langle\widetilde{\mathbb{R}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{R}}^{(m)}\rangle_{d+p-1})\}$, where $m \in \{1, \ldots, n-t\}$ and where each $\widetilde{\mathbb{R}}^{(m)} \in \mathbb{F}^{n^3p^2}$. Thus $(n-t)n^3p^2$ field elements are finally double-shared.

In the protocol, each party has to invoke 2 instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ as a dealer during step 2. From Lemma D.7, after securely realizing the broadcasts and parallel broadcasts via randomized protocols over point-to-point channels, this requires a constant expected number of rounds and incurs a *total* communication of $\mathsf{P2P}(n^6\kappa)$ bits, plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. The calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across all the dealers and all the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$. Hence the maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ due to instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ will be 15. In addition to this, the parties again call $\mathcal{F}_{\mathsf{Rand}}$ during step 3, so the maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is 16.

If the protocol is executed L times in parallel, then the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across various stages for all the L instances. Consequently, the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ remains at most 16. □

Communication Overhead of Π_{RandPair}. If Π_{RandPair} is executed L times in parallel, then the output of the protocol consists of packed-double-secret-sharing of $L \cdot (n-t)n^3p^2 = \Theta(L \cdot n^6)$ field elements. Since the communication complexity of the protocol is $\mathsf{P2P}(L \cdot n^6)$ elements, apart from 16 calls to $\mathcal{F}_{\mathsf{Rand}}$, it follows that for sufficiently large L, the communication overhead of the protocol is $\mathcal{O}(1)$.

We describe the simulation steps for Π_{RandPair} which will be used when the protocol is used as a primitive.

Simulator F.17: Simulator for the Protocol Π_{RandPair}

- On behalf of every party P_j not under the control of $\mathcal{A}$, randomly select $\mathbf{R}_j^{(1)}, \ldots, \mathbf{R}_j^{(n+1)}$ where each $\mathbf{R}_j^{(l)} \in \mathbb{F}^{n^3p^2}$ and simulate the steps for VSS for an honest dealer as per Simulator D.8.
- On behalf of every P_j under the control of $\mathcal{A}$, simulate the steps for VSS for a corrupt dealer as per Simulator D.8.
- For each P_k under the control of $\mathcal{A}$, receive Rand from $\mathcal{A}$, select a random r and send r to $\mathcal{A}$ for every P_k.
- On behalf of every party P_j not under the control of $\mathcal{A}$, for every $P_i \in \mathcal{P}$, compute P_j's share of ${}^{A}\langle \hat{\mathbf{R}}_i \rangle_d$ and ${}^{A}\langle \widetilde{\mathbf{R}}_i \rangle_{d+p-1}$ and send it to every party P_k under the control of $\mathcal{A}$ for reconstruction. Receive shares of ${}^{A}\langle \hat{\mathbf{R}}_i \rangle_d$ and ${}^{A}\langle \widetilde{\mathbf{R}}_i \rangle_{d+p-1}$ from $\mathcal{A}$ corresponding to each P_i under its control.
- For each P_i under the control of $\mathcal{A}$, if $\hat{\mathbf{R}}_i \neq \widetilde{\mathbf{R}}_i$, reset $\mathbb{R}_i = 0^{n^3p^2}$ and take its default sharing on behalf of each P_j not under $\mathcal{A}$'s control.

F.4 Generating Secret-Shared Random Triples

Protocol $\Pi_{\mathsf{RandTriples}}$ outputs APSS of the form $({}^{A}\langle \widetilde{\mathbb{A}} \rangle_d, {}^{A}\langle \widetilde{\mathbb{B}} \rangle_d, {}^{A}\langle \widetilde{\mathbb{C}} \rangle_d)$ with $\mathcal{O}(1)$ communication overhead, where each $\widetilde{\mathbb{A}}, \widetilde{\mathbb{B}}, \widetilde{\mathbb{C}} \in \mathbb{F}^{n^3p^2}$ and are random for the adversary. We stress that the triples *need not* be multiplication-triples. The idea behind the protocol is similar to Π_{RandPair}. Every party secret-shares sufficiently many vectors of random triples and then the parties extract secret-sharing of vectors of random triples from them based on Vandermonde's matrix. The protocol is relatively simpler because we *do not* need any verification mechanism to check the "equality" of the underlying values shared by each party.

Protocol F.18: Generating Secret-Shared Random Triples – $\Pi_{\mathsf{RandTriples}}$

1. Each party $P_i \in \mathcal{P}$ randomly selects $\mathbb{A}_i, \mathbb{B}_i, \mathbb{C}_i \in \mathbb{F}^{n^3p^2}$, acts a dealer and invokes instances of $\Pi_{A\langle\cdot\rangle}$ (Protocol 5.1) to generate ${}^{A}\langle \mathbb{A}_i \rangle_d, {}^{A}\langle \mathbb{B}_i \rangle_d, {}^{A}\langle \mathbb{C}_i \rangle_d$.
2. Let $\mathbb{A}(X), \mathbb{B}(X), \mathbb{C}(X)$ be the vectors of n^3p^2 degree-$(n-1)$ polynomials passing through the vectors of points $\{(1, \mathbb{A}_1), \ldots, (n, \mathbb{A}_n)\}$, $\{(1, \mathbb{B}_1), \ldots, (n, \mathbb{B}_n)\}$ and $\{(1, \mathbb{C}_1), \ldots, (n, \mathbb{C}_n)\}$ respectively. For $m \in \{1, \ldots, n-t\}$, let $c_{m,1}, \ldots, c_{m,n}$ be the publicly-known Lagrange's interpolation coefficients, where
   - $\widetilde{\mathbb{A}}^{(m)} = \mathbb{A}(n+m)$, and $\widetilde{\mathbb{A}}^{(m)} = c_{m,1} \cdot \mathbb{A}_1 + \ldots + c_{m,n} \cdot \mathbb{A}_n$.
   - $\widetilde{\mathbb{B}}^{(m)} = \mathbb{B}(n+m)$, and $\widetilde{\mathbb{B}}^{(m)} = c_{m,1} \cdot \mathbb{B}_1 + \ldots + c_{m,n} \cdot \mathbb{B}_n$.
   - $\widetilde{\mathbb{C}}^{(m)} = \mathbb{C}(n+m)$, and $\widetilde{\mathbb{C}}^{(m)} = c_{m,1} \cdot \mathbb{C}_1 + \ldots + c_{m,n} \cdot \mathbb{C}_n$.

   For each $m \in \{1, \ldots, n-t\}$, parties locally compute and output the following
   - ${}^{A}\langle \widetilde{\mathbb{A}}^{(m)} \rangle_d = c_{m,1} \cdot {}^{A}\langle \mathbb{A}_1 \rangle_d + \ldots + c_{m,n} \cdot {}^{A}\langle \mathbb{A}_n \rangle_d$.
   - ${}^{A}\langle \widetilde{\mathbb{B}}^{(m)} \rangle_d = c_{m,1} \cdot {}^{A}\langle \mathbb{B}_1 \rangle_d + \ldots + c_{m,n} \cdot {}^{A}\langle \mathbb{B}_n \rangle_d$.
   - ${}^{A}\langle \widetilde{\mathbb{C}}^{(m)} \rangle_d = c_{m,1} \cdot {}^{A}\langle \mathbb{C}_1 \rangle_d + \ldots + c_{m,n} \cdot {}^{A}\langle \mathbb{C}_n \rangle_d$.

The proof of Lemma F.19 and F.20 are similar to the proof of Lemma F.14 and F.15 respectively and skipped.

Lemma F.19 (Correctness). $\Pi_{\mathsf{RandTriples}}$ *outputs* $\{({}^{A}\langle\widetilde{\mathbb{A}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{B}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{C}}^{(m)}\rangle_d)\}$, *where for* $m \in \{1,\ldots,n-t\}$, $\widetilde{\mathbb{A}}^{(m)}, \widetilde{\mathbb{B}}^{(m)}, \widetilde{\mathbb{C}}^{(m)} \in \mathbb{F}^{n^3p^2}$.

Lemma F.20 (Privacy). *In* $\Pi_{\mathsf{RandTriples}}$*, the view of the adversary will be independent of* $\{({}^{A}\langle\widetilde{\mathbb{A}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{B}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{C}}^{(m)}\rangle_d)\}_{m\in\{1,\ldots,n-t\}}$.

Lemma F.21 (Complexity). *Protocol* $\Pi_{\mathsf{RandTriples}}$ *generates* $(n-t)n^3p^2$ *packed-secret-shared triples from* $\mathbb{F}$*, requires a constant expected number of rounds incurs a communication of* $\mathsf{P2P}(n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3\log^2 n)$ *bits and makes at most* 15 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

If $\Pi_{\mathsf{RandTriples}}$ *is executed* L *times in parallel, then it outputs* $L\cdot(n-t)n^3p^2$ *packed-secret-shared triples from* $\mathbb{F}$*, requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(L\cdot n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3\log^2 n)$ *bits and makes at most* 15 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

Proof. From Lemma F.19, $\Pi_{\mathsf{RandTriples}}$ outputs $\{({}^{A}\langle\widetilde{\mathbb{A}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{B}}^{(m)}\rangle_d, {}^{A}\langle\widetilde{\mathbb{C}}^{(m)}\rangle_d)\}$, where $m \in \{1,\ldots,n-t\}$ and where each $\widetilde{\mathbb{A}}^{(m)}, \widetilde{\mathbb{B}}^{(m)}, \widetilde{\mathbb{C}}^{(m)} \in \mathbb{F}^{n^3p^2}$. Thus $(n-t)n^3p^2$ triplets of field elements are finally secret-shared. Each party has to invoke an instance of $\Pi_{A\langle\cdot\rangle}$ as a dealer during step 1. From Lemma D.7, after securely realizing the broadcasts and parallel broadcasts via randomized protocols over point-to-point channels, this requires a constant expected number of rounds and incurs a *total* communication of $\mathsf{P2P}(n^6\kappa)$ bits, plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits. The calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across all the dealers and all the instances of $\Pi_{A\langle\cdot\rangle}$. Hence the maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ due to instances of $\Pi_{A\langle\cdot\rangle}$ will be 15.

If the protocol is executed L times in parallel, then the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across various stages for all the L instances. Consequently, the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ remains at most 15. □

Communication Overhead of the Protocol $\Pi_{\mathsf{RandTriples}}$. If $\Pi_{\mathsf{RandTriples}}$ is executed L times in parallel, it outputs packed-double-secret-sharing of $L\cdot(n-t)n^3p^2 = \Theta(L\cdot n^6)$ field elements. Since the communication complexity will be $\mathsf{P2P}(L\cdot n^6)$ elements, apart from 15 calls to $\mathcal{F}_{\mathsf{Rand}}$, it follows that for sufficiently large L, the communication overhead is $\mathcal{O}(1)$ elements.

We describe the steps for $\Pi_{\mathsf{RandTriples}}$ that will be used in simulation where $\Pi_{\mathsf{RandTriples}}$ protocol is used as a primitive.

Simulator F.22: Simulator for the Protocol $\Pi_{\mathsf{RandTriples}}$

– For every P_j not under $\mathcal{A}$'s control, randomly select $\mathbb{A}_j, \mathbb{B}_j, \mathbb{C}_j \in \mathbb{F}^{n^3p^2}$ and simulate the steps for VSS for an honest dealer as per Simulator D.8.

– On behalf of every P_j under the control of $\mathcal{A}$, simulate the steps for VSS for a corrupt dealer as per Simulator D.8.

F.5 Beaver Multiplication with a Constant Overhead

We have already discussed the overview and the complete protocol Π_{Beaver} in Sections 2 and 6 respectively. Here, we prove the properties of the protocol.

Claim F.23. *In protocol Π_{Beaver} if P_i is honest, and if P_i is included in C, then at least $\frac{t}{2}+1$ corrupt parties are included by P_i in Dp_i at the end of Complaint Resolution and Fault Identification, except with a negligible probability.*

Proof. Let P_i be an arbitrary honest party and let $P_i \in \mathsf{C}$. This implies that P_i broadcasted $\mathsf{complain}(i, B_i)$ during Step 5. This further implies that there exists some batch $\beta_i \in B_i$ with β_i being the smallest index in B_i, such that party P_i failed to robustly reconstruct $\mathbf{S}^{(\beta)}(i)$. This happened because at least $\frac{t}{2}+1$ corrupt parties sent incorrect shares of $\langle \mathbf{S}^{(\beta_i)}(i) \rangle_{d+p-1}$ while reconstructing $\mathbf{S}^{(\beta_i)}(i)$ towards P_i. If this is not true then P_i would have error-correct all the incorrect shares of $\langle \mathbf{S}^{(\beta_i)}(i) \rangle_{d+p-1}$ and robustly reconstruct $\mathbb{S}^{(\beta_i)}(i)$, implying $\beta_i \notin B_i$, which is a contradiction. This also means that $\beta_i \in B_i$ is chosen as the smallest index for which the Complaint Resolution is performed corresponding to party P_i.

During step 9, party P_i robustly reconstructs $\mathbf{X}^{(\beta_i,\gamma)} + \mathbf{A}^{(\beta_i,\gamma)}$, $\mathbf{Y}^{(\beta_i,\gamma)} + \mathbf{B}^{(\beta_i,\gamma)}$, $\mathbf{A}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{A}}^{(\beta_i,\gamma)}$, $\mathbf{B}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{B}}^{(\beta_i,\gamma)}$, $\mathbf{C}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{C}}^{(\beta_i,\gamma)}$ and $\mathbf{R}^{(\beta_i,\gamma)} + R \cdot \hat{\mathbf{R}}^{(\beta_i,\gamma)}$, where $\gamma \in \{1, \ldots, t+1\}$, except with a negligible probability, from the ${}^{A}\langle \cdot \rangle_d$-sharings of these values. This is because P_i will be able to identify the correct shares due to the presence of MAC-tags. Moreover, P_i will be able to recompute the entire PSS (namely $\langle \cdot \rangle_d$-sharing) of each of these values. Consequently, P_i will be able to compute the entire sharing $\langle \mathbf{S}^{(\beta_i)}(i) + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1}$ during step 10. This is because $\langle \mathbf{S}^{(\beta_i)}(i) + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1}$ is computed deterministically from the $\langle \cdot \rangle_d$-sharings recomputed by P_i. Then by comparing the shares of each party in the recomputed $\langle \mathbf{S}^{(\beta_i)(i)} + R \cdot \hat{\mathbf{S}}^{(\beta_i)}(i) \rangle_{d+p-1}$ with the shares received from various parties during step 5 and step 8, party P_i will be able to identify all corrupt parties who sent incorrect shares to P_i in step 5 during the reconstruction of $\mathbf{S}^{(\beta_i)}(i)$ and include them in Dp_i. Since more than $\frac{t}{2}+1$ corrupt parties sent incorrect shares to P_i during step 5, it follows that $|\mathsf{Dp}_i| \geq \frac{t}{2}+1$. □

Based on Claim F.23, we next show that for any *honest* party that complains, its complaints are resolved and it will reconstruct $\mathbb{S}(i)$.

Claim F.24. *In protocol Π_{Beaver} if P_i is honest, and $P_i \in \mathsf{C}$ then by the end of the Complaint Resolution and Fault Identification phase, P_i will reconstruct $\mathbb{S}(i)$, except with a negligible probability.*

Proof. Let P_i be an arbitrary honest party such that $P_i \in \mathsf{C}$. Then from Claim F.23, except with a negligible probability, party P_i includes at least $\frac{t}{2}+1$ corrupt parties in the set Dp_i during Step 10.

To see that P_i will be able to robustly reconstruct $\mathbb{S}(i)$, consider any arbitrary vector $\hat{\mathbf{s}} \in \mathbb{S}(i)$ where $\hat{\mathbf{s}} \in \mathbb{F}^p$. Note that as part of $\langle \mathbb{S}(i) \rangle_{d+p-1}$, the vector $\hat{\mathbf{s}}$ is degree-$(d+p-1)$-packed-secret-shared and the parties hold $\langle \hat{\mathbf{s}} \rangle_{d+p-1}$. During step 5, while reconstructing $\mathbb{S}(i)$ towards P_i, every party will send their respective share of $\langle \hat{\mathbf{s}} \rangle_{d+p-1}$ to P_i. Note that $|\mathsf{Dp}_i| \geq \frac{t}{2}+1$ and hence P_i will ignore the shares of at least $\frac{t}{2}+1$ corrupt parties. Then among the remaining shares, there could be at most $\frac{t}{2}-1$ unknown corrupt shares. Moreover, the underlying sharing-polynomial of $\langle \hat{\mathbf{s}} \rangle_{d+p-1}$ is of degree-$(t+\frac{t}{2})$. Since $n = 3t+1$, it follows that P_i can easily error-correct the remaining $\frac{t}{2}-1$ incorrect shares by using RS error-correction and robustly reconstruct the vector $\hat{\mathbf{s}}$. □

Now based on all the previous results, we show that protocol Π_{Beaver} will generate the desired outcome.

Lemma F.25 (Correctness). *Given inputs* $\{{}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d\}$ *and auxiliary inputs* $\{{}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d\}$, $\{{}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{C}}^{(\beta,\gamma)} \rangle_d\}$, $\{{}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1}\}$ *and* $\{{}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1}\}$. *Then except with a negligible probability,* Π_{Beaver} *generates* $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d\}$, *where* $\mathbb{Z}^{(\gamma)} = \mathbb{X}^{(\gamma)} \times \mathbb{Y}^{(\gamma)}$ *component-wise, if and only if* $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ *constitute multiplication-triples component-wise. Here* $\gamma \in \{1, \ldots, t+1\}$.

Proof. From Claim F.24, we have that each honest party reconstructs its $\mathbb{S}(i)$, either during Step 5 or after Complaint Resolution and Fault Identification during Step 10. Consequently, it must hold that all parties execute step 11a of the protocol, where every P_i acts as dealer and supposedly generates ${}^{A}\langle \mathbb{S}(i) \rangle_{p-1}$ through $\Pi_{{}^{A}\langle \cdot \rangle}$ by acting as a dealer. Note that if P_i is corrupt then it might generate ${}^{A}\langle \cdot \rangle_{p-1}$-sharings of incorrect $\mathbb{S}(i)$ and there could be up to t such corrupt parties. Now similar to the proof of Lemma F.10, as observed in [23], the shares (points of the underlying polynomials) of each party corresponding to ${}^{A}\langle \cdot \rangle_{p-1}$-sharing of $\mathbb{S}(1), \ldots, \mathbb{S}(n)$ themselves constitute codewords of length-n, of which up to t may be erroneous due to the corrupt parties. Thus, each P_i can locally identify the error positions using RS-error correction procedure. For simplicity let the error positions be $\{1, \ldots, t\}$. It can now linearly combine its shares and MAC-tags of ${}^{A}\langle \mathbb{S}(t+1) \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}(n) \rangle_{p-1}$ to obtain its correct shares corresponding to ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$. The linear combination is as per Lagrange's interpolation. To allow P_j to adjust its verification information for MACs, P_i broadcasts an n-length bit vector indicating whose resharing is used. Different P_i may use different set of resharings. But the correctness of two honest parties share of ${}^{A}\langle \mathbb{S}^{(1)} \rangle_{p-1}, \ldots, {}^{A}\langle \mathbb{S}^{(t+1)} \rangle_{p-1}$ holds due to correctness of RS error-correction, Lagrange's interpolation and linearity of MACs.

Now we have that the output of the protocol will be $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d\}$, where $\gamma \in \{1, \ldots, t+1\}$. To complete the proof, we just need to show that component-wise $\mathbb{Z}^{(\gamma)} = \mathbb{X}^{(\gamma)} \times \mathbb{Y}^{(\gamma)}$ holds, if and only if $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ constitute multiplication-triples component-wise; i.e. $\mathbb{C}^{(\gamma)} = \mathbb{A}^{(\gamma)} \times \mathbb{B}^{(\gamma)}$ holds. However, this simply follows

from the fact that $\mathbb{Z}^{(\gamma)} = (\mathbb{X}^{(\gamma)}+\mathbb{A}^{(\gamma)})\times(\mathbb{Y}^{(\gamma)}+\mathbb{B}^{(\gamma)})-(\mathbb{X}^{(\gamma)}+\mathbb{A}^{(\gamma)})\times\mathbb{B}^{(\gamma)}-(\mathbb{Y}^{(\gamma)}+\mathbb{B}^{(\gamma)})\times\mathbb{A}^{(\gamma)}+\mathbb{C}^{(\gamma)}$. Hence if $\mathbb{C}^{(\gamma)} = \mathbb{A}^{(\gamma)}\times\mathbb{B}^{(\gamma)}$, then $\mathbb{Z}^{(\gamma)} = (\mathbb{X}^{(\gamma)}+\mathbb{A}^{(\gamma)}-\mathbb{A}^{(\gamma)})\times(\mathbb{Y}^{(\gamma)}+\mathbb{B}^{(\gamma)}-\mathbb{B}^{(\gamma)})$, which further implies that $\mathbb{Z}^{(\gamma)} = \mathbb{X}^{(\gamma)}\times\mathbb{Y}^{(\gamma)}$. On the other hand, if $\mathbb{C}^{(\gamma)} \neq \mathbb{A}^{(\gamma)}\times\mathbb{B}^{(\gamma)}$ component-wise then clearly $\mathbb{Z}^{(\gamma)} \neq \mathbb{X}^{(\gamma)}\times\mathbb{Y}^{(\gamma)}$. □

Lemma F.26 (Privacy). *In* Π_{Beaver}*, the adversary's view remains independent of* $\{\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}\}$*, provided* $\{\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)}, \hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)}, \mathbb{R}^{(\gamma)}, \hat{\mathbb{R}}^{(\gamma)}\}$ *are random for the adversary. Here* $\gamma \in \{1, \ldots, t+1\}$.

Proof. During step 2, adversary completely learns $\mathbb{X}^{(\gamma)} + \mathbb{A}^{(\gamma)}$ and $\mathbb{Y}^{(\gamma)} + \mathbb{B}^{(\gamma)}$. If $\mathbb{A}^{(\gamma)}$ and $\mathbb{B}^{(\gamma)}$ are random for the adversary, then it follows that $\mathbb{X}^{(\gamma)}$ and $\mathbb{Y}^{(\gamma)}$ still remain random for the adversary.

During the protocol, adversary may learn the vector of polynomials $\mathbb{S}(X)$. However, this does not reveal additional information to the adversary. This is because, from $\mathbb{S}(X)$, the adversary may learn each $\mathbb{S}^{(\gamma)}$. However, $\mathbb{S}^{(\gamma)} = \mathbb{Z}^{(\gamma)} + \mathbb{R}^{(\gamma)}$. If $\mathbb{R}^{(\gamma)}$ is random, then it follows that $\mathbb{Z}^{(\gamma)}$ remains random.

Further, for each corrupt party P_i, the adversary may additionally learn $\hat{\mathbb{S}}(i)$ and hence each $\hat{\mathbb{S}}^{(\gamma)}$. However, $\hat{\mathbb{S}}^{(\gamma)} = \hat{\mathbb{Z}}^{(\gamma)} + \hat{\mathbb{R}}^{(\gamma)}$ and if $\hat{\mathbb{R}}^{(\gamma)}$ is random for the adversary, it follows that $\hat{\mathbb{Z}}^{(\gamma)}$ remains random from the adversary.

Finally, for each corrupt P_i and specific index β_i, the adversary learns $\mathbf{X}^{(\beta_i,\gamma)} + \mathbf{A}^{(\beta_i,\gamma)}, \mathbf{Y}^{(\beta_i,\gamma)} + \mathbf{B}^{(\beta_i,\gamma)}, \mathbf{A}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{A}}^{(\beta_i,\gamma)}, \mathbf{B}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{B}}^{(\beta_i,\gamma)}, \mathbf{C}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{C}}^{(\beta_i,\gamma)}$, and $\mathbf{R}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{R}}^{(\beta_i,\gamma)}$ in Step 10. From this, the adversary learns $\mathbf{Z}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{Z}}^{(\beta_i,\gamma)} = (\mathbf{S}^{(\beta_i,\gamma)} + R\cdot\hat{\mathbf{S}}^{(\beta_i,\gamma)}) - (\mathbf{R}^{(\beta_i,\gamma,\delta_r)} + R\cdot\hat{\mathbf{R}}^{(\beta_i,\gamma,\delta_r)})$. However, $\mathbf{Z}^{(\beta_i,\gamma)}$ remains random for the adversary if $\mathbf{A}^{(\beta_i,\gamma)}, \mathbf{B}^{(\beta_i,\gamma)}$ is random and unknown to the adversary. □

Lemma F.27 (Complexity). *Protocol* Π_{Beaver} *requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3\log^2 n)$ *bits and makes at most* 31 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

If the protocol is executed L *times in parallel where* $L \geq 1$*, then it requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(L\cdot n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3\log^2 n)$ *bits and makes at most* 31 *calls to* $\mathcal{F}_{\mathsf{Rand}}$.

Proof. Consider the first stage (namely steps 1-3). In step 2, the parties execute an instance of Π_{RoDegRed}. From Lemma F.11, this requires a constant expected number of rounds and a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus an expected $\mathsf{P2P}(n^3\log^2 n)$ bits, apart from at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$.

Next consider the second stage (steps 4-7). In step 5, each P_i needs to reconstruct a $\langle\cdot\rangle_{d+p-1}$-sharing of n^3p^2 values. For *one* P_i this requires a single round and a communication of $\mathsf{P2P}(n^5\kappa)$ bits and hence this requires a *total* communication of $\mathsf{P2P}(n^6\kappa)$ bits. Additionally, each P_i may broadcast a set B_i, which can be encoded as a Boolean vector of size n bits. This will require a communication of $n\times\mathsf{BC}(n)$ bits.

Next consider the third stage (namely steps 8-10). In step 8, each P_i needs to reconstruct a $\langle\cdot\rangle_{d+p-1}$-sharing of n^3p^2 elements from $\mathbb{F}$. This requires one round and a total communication of $\mathsf{P2P}(n^6\kappa)$ bits. Step 9 involves a single invocation of $\mathcal{F}_{\mathsf{Rand}}$. Additionally, in Step 9, each P_i may need to reconstruct $6\cdot(t+1)$

number of ${}^{A}\langle\cdot\rangle_d$-sharings of n^2p^2 elements from $\mathbb{F}$. This will require a single round and a total communication of $\mathsf{P2P}(n^6\kappa)$ bits.

Finally consider the last stage (steps 11a-11c). Here each P_i generates a ${}^{A}\langle\cdot\rangle_{p-1}$-sharing of n^3p^2 elements by running an instance of $\Pi_{{}^{A}\langle\cdot\rangle}$. From Lemma D.7, after securely realizing the broadcasts over point-to-point channels, this requires a constant expected number of rounds and a total communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^4\log^2 n)$ bits, apart from at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$. □

Communication Overhead of Protocol Π_{Beaver}. If Π_{Beaver} is executed L times in parallel, it outputs the product of $L \cdot (t+1)n^3p^2 = \Theta(L \cdot n^6)$ pairs of elements. Since the communication complexity will be $\mathsf{P2P}(L \cdot n^6)$ elements, apart from at most 31 calls to $\mathcal{F}_{\mathsf{Rand}}$, it follows that for sufficiently large L, the communication overhead is a constant.

We next describe the simulation steps for Π_{Beaver} which will be used when the protocol is used as a primitive.

Simulator F.28: Simulator for the Protocol Π_{Beaver}

- On behalf of each honest party P_i, the initialize $\mathsf{C} = \emptyset$ and $\mathsf{Dp}_i = \phi$.
- On behalf of each honest party, perform the local computations on the shares as an honest party would and simulate Π_{RoDegRed} as in Simulation F.12 using the shares already chosen on behalf of the honest parties.
- Send shares of $\langle\mathbb{S}(k)\rangle_{d+p-1}$ to each P_k under $\mathcal{A}$'s control to allow it to reconstruct $\mathbb{S}(k)$ towards P_k. Receive from $\mathcal{A}$, the shares corresponding to $\langle\mathbb{S}(i)\rangle_{d+p-1}$ for each honest P_i.
- If more than $\frac{t}{2}$ erroneous shares are received from $\mathcal{A}$ corresponding to some honest party P_i, then emulate broadcast, and send $\mathsf{complain}(i, B_i)$ to each P_k under the control of $\mathcal{A}$. Additionally, if $\mathsf{complain}(k, B_k)$ is received from $\mathcal{A}$ corresponding to some corrupt P_k, then send it to all.
- Compute the steps locally on behalf of the honest parties as per the protocol specification.
- Receives Rand from $\mathcal{A}$ and emulate $\mathcal{F}_{\mathsf{Rand}}$ by sending R to all.
- For each $P_k \in \mathsf{C}$ such that P_k is under the control of $\mathcal{A}$, send the honest parties' shares corresponding to the following to $\mathcal{A}$:
  - ${}^{A}\langle\mathbf{X}^{(\beta_k,\gamma)} + \mathbf{A}^{(\beta_k,\gamma)}\rangle_d$, ${}^{A}\langle\mathbf{Y}^{(\beta_k,\gamma)} + \mathbf{B}^{(\beta_k,\gamma)}\rangle_d$
  - ${}^{A}\langle\mathbf{A}^{(\beta_k,\gamma)} + R\cdot\hat{\mathbf{A}}^{(\beta_k,\gamma)}\rangle_d$, ${}^{A}\langle\mathbf{B}^{(\beta_k,\gamma)} + R\cdot\hat{\mathbf{B}}^{(\beta_k,\gamma)}\rangle_d$, ${}^{A}\langle\mathbf{C}^{(\beta_k,\gamma)} + R\cdot\hat{\mathbf{C}}^{(\beta_k,\gamma)}\rangle_d$
  - ${}^{A}\langle\mathbf{R}^{(\beta_k,\gamma)} + R\cdot\hat{\mathbf{R}}^{(\beta_k,\gamma)}\rangle_d$

  Also receive the shares corresponding to each honest $P_i \in \mathsf{C}$ from $\mathcal{A}$.
- Compute the steps on behalf of the honest parties as per the protocol specification.
- On behalf of each honest P_i, simulate the steps of VSS for an honest dealer as in Simulator D.8 and for each P_k under the control of $\mathcal{A}$, simulate the steps of VSS for a corrupt dealer as in Simulator D.8.

F.6 Triple Sharing with a Constant Overhead

Our next protocol Π_{TripSh} (Protocol F.29) allows a dealer D to verifiably secret-share random multiplication-triples, with a constant overhead. The verifiability here guarantees that even if D is *corrupt*, it has secret-shared multiplication-triples, except with a negligible probability. Moreover, if the dealer is *honest*, then the privacy of dealer's triples in maintained.

Given protocols Π_{VSS} and Π_{Beaver}, we can design Π_{TripSh} as follows; we explain the protocol overview over PSS of one-dimensional vectors, but to get $\mathcal{O}(1)$ overhead, the protocol is executed over APSS of three-dimensional vectors. D picks n vectors of random multiplication-triples $(\mathbf{u}^{(i)}, \mathbf{v}^{(i)}, \mathbf{w}^{(i)})$ and generates $(\langle \mathbf{u}^{(i)} \rangle_d, \langle \mathbf{v}^{(i)} \rangle_d, \langle \mathbf{w}^{(i)} \rangle_d)$ using VSS. To verify if D has secret-shared multiplication-triples, the parties "transform" $(\langle \mathbf{u}^{(i)} \rangle_d, \langle \mathbf{v}^{(i)} \rangle_d, \langle \mathbf{w}^{(i)} \rangle_d)$ into "correlated" triples $(\langle \boldsymbol{x}^{(i)} \rangle_d, \langle \boldsymbol{y}^{(i)} \rangle_d, \langle \boldsymbol{z}^{(i)} \rangle_d)$ such that the following hold:

- There exist vectors of polynomials $\boldsymbol{X}(\cdot), \boldsymbol{Y}(\cdot)$ and $\boldsymbol{Z}(\cdot)$ of degree-$\frac{n-1}{2}$, $\frac{n-1}{2}$ and degree-$(n-1)$ respectively, such that $\boldsymbol{X}(i) = \boldsymbol{x}^{(i)}, \boldsymbol{Y}(i) = \boldsymbol{y}^{(i)}$ and $\boldsymbol{Z}(i) = \boldsymbol{z}^{(i)}$ for each $i \in \{1, \ldots, n\}$.
- The vectors of polynomials $(\boldsymbol{X}(\cdot), \boldsymbol{Y}(\cdot), \boldsymbol{Z}(\cdot))$ satisfy the multiplicative relationship $\boldsymbol{Z}(\cdot) = \boldsymbol{X}(\cdot) \cdot \boldsymbol{Y}(\cdot)$ if and only if all the input triples $(\mathbf{u}^{(i)}, \mathbf{v}^{(i)}, \mathbf{w}^{(i)})$ are multiplication-triples.

The transformation works as follows: the first $\frac{n-1}{2}+1$ transformed triples are set to be the first $\frac{n-1}{2}+1$ input triples. That is, for each $i \in \{1, \ldots, \frac{n-1}{2}+1\}$, the parties set $\langle \boldsymbol{x}^{(i)} \rangle_d = \langle \mathbf{u}^{(i)} \rangle_d$, $\langle \boldsymbol{y}^{(i)} \rangle_d = \langle \mathbf{v}^{(i)} \rangle_d$ and $\langle \boldsymbol{z}^{(i)} \rangle_d = \langle \mathbf{w}^{(i)} \rangle_d$. The polynomials in $\boldsymbol{X}(\cdot)$ and $\boldsymbol{Y}(\cdot)$ are then defined through the first and second components of the first $\frac{n-1}{2}+1$ transformed triples. That is, let $\boldsymbol{X}(\cdot)$ and $\boldsymbol{Y}(\cdot)$ be the vectors of degree-$\frac{n-1}{2}$ polynomials, passing through the vectors of $\frac{n-1}{2}+1$ points $\{(i, \mathbf{u}^{(i)})\}$ and $\{(i, \mathbf{v}^{(i)})\}$, where $i \in \{1, \ldots, \frac{n-1}{2}+1\}$. The polynomials in $\boldsymbol{X}(\cdot)$ and $\boldsymbol{Y}(\cdot)$ are next "extended" for $\frac{n-1}{2}$ new points in a secret-shared fashion, which are then set to be the first and second components of the remaining $\frac{n-1}{2}$ transformed triples. That is, for each $i \in \{\frac{n-1}{2}+2, \ldots, n\}$, let $\boldsymbol{x}^{(i)} = \boldsymbol{X}(i)$ and $\boldsymbol{y}^{(i)} = \boldsymbol{Y}(i)$. Note that each of these $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(i)}$ is a publicly-known linear combination of the vectors $\{\boldsymbol{x}^{(j)}\}_{j \in \{1, \ldots, \frac{n-1}{2}+1\}}$ and $\{\boldsymbol{y}^{(j)}\}_{j \in \{1, \ldots, \frac{n-1}{2}+1\}}$ respectively. So using the linearity property of PSS, the parties can non-interactively compute $\langle \boldsymbol{x}^{(i)} \rangle_d$ and $\langle \boldsymbol{y}^{(i)} \rangle_d$ from $\{\langle \boldsymbol{x}^{(j)} \rangle_d\}_{j \in \{1, \ldots, \frac{n-1}{2}+1\}}$ and $\{\langle \boldsymbol{y}^{(j)} \rangle_d\}_{j \in \{1, \ldots, \frac{n-1}{2}+1\}}$ respectively. Finally, the parties try to compute the product of these extended points, using the remaining triples shared by D (which are not touched till now). That is, for every $j \in \{\frac{n-1}{2}+2, \ldots, n\}$, the parties run the protocol Π_{Beaver} on inputs $\langle \boldsymbol{x}^{(j)} \rangle_d$ and $\langle \boldsymbol{y}^{(j)} \rangle_d$, using the shared triples $(\langle \mathbf{u}^{(j)} \rangle_d, \langle \mathbf{v}^{(j)} \rangle_d, \langle \mathbf{w}^{(j)} \rangle_d)$ as the auxiliary triple and compute $\langle \boldsymbol{z}^{(j)} \rangle_d$. It follows that $\boldsymbol{z}^{(j)} = \boldsymbol{x}^{(j)} \times \boldsymbol{y}^{(j)}$ component-wise if and only if $(\mathbf{a}^{(j)}, \mathbf{b}^{(j)}, \mathbf{c}^{(j)})$ constitute vector of multiplication-triples. The polynomials in $\boldsymbol{Z}(\cdot)$ are then defined through the third components of all the n transformed triples. That is, $\boldsymbol{Z}(i) = \boldsymbol{z}^{(i)}$, for every $i \in \{1, \ldots, n\}$. Note that if D is *honest*, then throughout this transformation, the privacy of the polynomials in $\boldsymbol{X}(\cdot), \boldsymbol{Y}(\cdot)$ and $\boldsymbol{Z}(\cdot)$ is maintained.

It follows that $\boldsymbol{Z}(\cdot) = \boldsymbol{X}(\cdot) \cdot \boldsymbol{Y}(\cdot)$ holds iff all the n vectors of triples shared by D are multiplication-triples. Hence to verify the triples of D, it is sufficient to verify the multiplicative-relationship among the polynomials. For this, the parties publicly check a random point on these polynomials, leaving $\frac{n-1}{2}+1-1$ "degree of freedom" in the polynomials in $\boldsymbol{X}(\cdot), \boldsymbol{Y}(\cdot)$ and $\boldsymbol{Z}(\cdot)$. Consequently, the parties (locally) compute PSS of $\frac{n-1}{2}$ new points on the polynomials in $\boldsymbol{X}(\cdot), \boldsymbol{Y}(\cdot)$ and $\boldsymbol{Z}(\cdot)$, which are taken as the output on behalf of D.

We extend the above idea to three dimensional vectors. In more detail, D generates ${}^{A}\langle\cdot\rangle_d$-sharings of vectors of multiplication-triples $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$, where $\gamma \in \{1, \ldots, 2t+1\}$. To prove that these vectors of triples are indeed multiplication-triples, the parties transform them into vectors of triples $\{(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})\}$, such that the following holds.

- There exist vectors of degree-t, t and degree-$2t$ polynomials $\mathbb{X}(\cdot), \mathbb{Y}(\cdot)$ and $\mathbb{Z}(\cdot)$ respectively, passing through the elements in $\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}$ and $\mathbb{Z}^{(\gamma)}$ respectively, where $\gamma \in \{1, \ldots, 2t+1\}$.
- Component-wise, $\mathbb{X}(\cdot) \times \mathbb{Y}(\cdot) = \mathbb{Z}(\cdot)$ if and only if the triples shared by the dealer are multiplication-triples.

Next to verify the multiplicativity relationship among $\mathbb{X}(\cdot), \mathbb{Y}(\cdot)$ and $\mathbb{Z}(\cdot)$, the parties publicly check the relationship at a random point r generated through $\mathcal{F}_{\mathsf{Rand}}$. If the verification is successful, then the parties extract vectors of secret-shared random multiplication-triples on behalf of the dealer, by extending the vectors of polynomials $\mathbb{X}(\cdot), \mathbb{Y}(\cdot)$ and $\mathbb{Z}(\cdot)$ at new points.

Protocol F.29: Packed Triple Sharing – Π_{TripSh}

(Sharing Uncorrelated Triples)

1. For $\gamma \in \{1, \ldots, 2t+1\}$, D randomly selects $\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)}$ from $\mathbb{F}$ where each $\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$ and where component-wise, $\mathbb{C}^{(\gamma)} = \mathbb{A}^{(\gamma)} \times \mathbb{B}^{(\gamma)}$. D generates ${}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d$, ${}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d$ by invoking instances of $\Pi_{A\langle\cdot\rangle}$ (protocol 5.1).
2. D randomly selects $\hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$ and generates ${}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d$, ${}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_d$ by acting as a dealer and invoking instances of $\Pi_{A\langle\cdot\rangle}$.

(Generating Secret-Shared Random Pairs)

3. The parties invoke instances of Π_{RandPair} (Protocol F.13) to generate $2 \cdot t$ number of vectors of secret-shared random double pairs, denoted by $({}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1})$ and $({}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1})$, where $\gamma \in \{t+2, \ldots, 2t+1\}$ and where $\mathbb{R}^{(\gamma)}, \hat{\mathbb{R}}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$.

(Transformation to Correlated Triples)

4. For each $\gamma \in \{1, \ldots, t+1\}$, parties locally set
   - ${}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d$,

- ${}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d$, and
- ${}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d$.

5. Let $\mathbb{X}(\cdot)$ and $\mathbb{Y}(\cdot)$ be the vectors of n^3p^2 degree-t polynomials, passing through the vectors of $t+1$ points $(1, \mathbb{X}^{(1)}), \ldots, (t+1, \mathbb{X}^{(t+1)})$ and $(1, \mathbb{Y}^{(1)}), \ldots, (t+1, \mathbb{Y}^{(t+1)})$ respectively.
6. For each $\gamma \in \{t+2, \ldots, 2t+1\}$, parties locally compute ${}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{X}(\gamma) \rangle_d$ from $\{{}^{A}\langle \mathbb{X}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{X}^{(t+1)} \rangle_d\}$.
7. For each $\gamma \in \{t+2, \ldots, 2t+1\}$, parties locally compute ${}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d = {}^{A}\langle \mathbb{Y}(\gamma) \rangle_d$ from $\{{}^{A}\langle \mathbb{Y}^{(1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{Y}^{(t+1)} \rangle_d\}$.
8. For each $\gamma \in \{t+2, \ldots, 2t+1\}$, the parties invoke instances of Π_{Beaver} with inputs $\{{}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d\}$ and auxiliary inputs $\{{}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d\}$, $\{{}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1}\}$, $\{{}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1}\}$ to compute $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d\}$ where $\mathbb{Z}^{(\gamma)} = \mathbb{X}^{(\gamma)} \times \mathbb{Y}^{(\gamma)}$ should supposedly hold.
9. Let $\mathbb{Z}(\cdot)$ be the vectors of n^3p^2 degree-$2t$ polynomials passing through the vectors of points $(1, \mathbb{Z}^{(1)}), \ldots, (2t+1, \mathbb{Z}^{(2t+1)})$.

(Verification of Triples and Output Computation)

10. Parties call $\mathcal{F}_{\mathsf{Rand}}$ and receive a random value $r \in \mathbb{F}$.
11. Parties locally compute ${}^{A}\langle \mathbb{X}(r) \rangle_d, {}^{A}\langle \mathbb{Y}(r) \rangle_d$ and ${}^{A}\langle \mathbb{Z}(r) \rangle_d$ from $\{{}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d\}$, $\{{}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d\}$ respectively, where $\gamma \in \{1, \ldots, 2t+1\}$.
12. Parties publicly reconstruct $\mathbb{X}(r), \mathbb{Y}(r)$ and $\mathbb{Z}(r)$ and check if component-wise $\mathbb{Z}(r) = \mathbb{X}(r) \times \mathbb{Y}(r)$ holds.
13. If $\mathbb{Z}(r) \neq \mathbb{X}(r) \times \mathbb{Y}(r)$ then the parties discard D and output default ${}^{A}\langle \cdot \rangle_d$-sharings of 0's on behalf of D.
14. Else let $\eta_1, \ldots, \eta_t$ be distinct elements from $\mathbb{F} \backslash \{1, \ldots, n, r\}$. For $i \in \{1, \ldots, t\}$, let $\mathbb{X}_i = \mathbb{X}(\eta_i), \mathbb{Y}_i = \mathbb{Y}(\eta_i)$ and $\mathbb{Z}_i = \mathbb{Z}(\eta_i)$. Parties locally compute ${}^{A}\langle \mathbb{X}_i \rangle_d, {}^{A}\langle \mathbb{Y}_i \rangle_d$ and ${}^{A}\langle \mathbb{Z}_i \rangle_d$ from $\{{}^{A}\langle \mathbb{X}^{(\gamma)} \rangle_d\}, \{{}^{A}\langle \mathbb{Y}^{(\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \mathbb{Z}^{(\gamma)} \rangle_d\}$ respectively, where $\gamma \in \{1, \ldots, 2t+1\}$ and output ${}^{A}\langle \mathbb{X}_i \rangle_d, {}^{A}\langle \mathbb{Y}_i \rangle_d$ and ${}^{A}\langle \mathbb{Z}_i \rangle_d$, where $i \in \{1, \ldots, t\}$.

Lemma F.30 (Correctness and Privacy). *If D is honest, then except with a negligible probability, parties output $\{{}^{A}\langle \mathbb{X}_i \rangle_d, {}^{A}\langle \mathbb{Y}_i \rangle_d, {}^{A}\langle \mathbb{Z}_i \rangle_d\}$, where $i \in \{1, \ldots, t\}$, such that $\mathbb{X}_i, \mathbb{Y}_i, \mathbb{Z}_i \in \mathbb{F}^{n^3p^2}$ and component-wise $\mathbb{Z}_i = \mathbb{X}_i \times \mathbb{Y}_i$. The view of the adversary will be independent of $\mathbb{X}_i, \mathbb{Y}_i$ and $\mathbb{Z}_i$.*

Proof. If D is *honest*, then it will select vectors of random multiplication-triples $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ during step 1 and vectors of random multiplication-triples $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ during step 2. From the correctness of $\Pi_{A\langle\cdot\rangle}$ (Lemma D.5), parties will hold ${}^{A}\langle \cdot \rangle_d$-sharing of these values, except with a negligible probability. Moreover, parties will hold ${}^{A}\langle \cdot \rangle_d$-sharings of the vectors of triples $(\hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)})$. From the correctness of Π_{RandPair} (Lemma F.14), at the end of step 3, parties will hold $({}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\gamma)} \rangle_{d+p-1})$ and $({}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\gamma)} \rangle_{d+p-1})$, except with a negligible probability. From the protocol steps, it follows that for each $\gamma \in \{1, \ldots, t+1\}$,

the vectors of triples $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ constitute multiplication-triples. The correctness property of Π_{Beaver} (Lemma F.25) guarantees that the same hold even for $\gamma \in \{t+2, \ldots, 2t+1\}$, except with a negligible probability. Consequently, the vectors of polynomials $(\mathbb{X}(\cdot), \mathbb{Y}(\cdot), \mathbb{Z}(\cdot))$ will satisfy the multiplicative relation and hence component-wise, $\mathbb{Z}(r) = \mathbb{X}(r) \times \mathbb{Y}(r)$ will hold for any $r \in \mathbb{F}$. Since the parties will robustly reconstruct $\mathbb{X}(r), \mathbb{Y}(r)$ and $\mathbb{Z}(r)$ except with a negligible probability, it follows that D will not be discarded. From the protocol steps, it now follows that parties will output $\{{}^{A}\langle \mathbb{X}_i \rangle_d, {}^{A}\langle \mathbb{Y}_i \rangle_d, {}^{A}\langle \mathbb{Z}_i \rangle_d\}$ during step 14. Moreover, the vectors of triples $(\mathbb{X}_i, \mathbb{Y}_i, \mathbb{Z}_i)$ will be multiplication-triples, as they constitute vectors of points on the vectors of polynomials $(\mathbb{X}(\cdot), \mathbb{Y}(\cdot), \mathbb{Z}(\cdot))$ which satisfy the multiplicative relation.

To argue the privacy of the output multiplication-triples $(\mathbb{X}_i, \mathbb{Y}_i, \mathbb{Z}_i)$, we need to show that throughout the protocol, adversary will learn at most one vector of points on the vectors of polynomials $(\mathbb{X}_i, \mathbb{Y}_i, \mathbb{Z}_i)$. Since the polynomials in $\mathbb{X}(\cdot)$ and $\mathbb{Y}(\cdot)$ are of degree-t, the privacy follows.

From the privacy of $\Pi_{{}^{A}\langle \cdot \rangle}$ (Lemma D.3), the vectors of multiplication-triples $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ and the vectors of triples $(\hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)})$ will remain random for the adversary, at the end of step 1 and 2 respectively. From the privacy of Π_{RandPair} (Lemma F.15), the vectors of elements $\mathbb{R}^{(\gamma)}$ and $\hat{\mathbb{R}}^{(\gamma)}$ will remain random for the adversary. From the protocol steps and from the privacy of protocol Π_{Beaver} (Lemma F.26), it follows that the view of the adversary will be independent of the multiplication-triples $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$, for every $\gamma \in \{1, \ldots, 2t+1\}$. Hence till the end of step 9, the view of the adversary will be independent of the vectors of polynomials $(\mathbb{X}(\cdot), \mathbb{Y}(\cdot), \mathbb{Z}(\cdot))$, except that they satisfy the multiplicative relationship. Finally during step 12, adversary may learn the vectors of points $\mathbb{X}(r), \mathbb{Y}(r)$ and $\mathbb{Z}(r)$, as they are publicly reconstructed. This completes the proof of the lemma. □

Lemma F.31 (Commitment). *If D is corrupt, then except with a negligible probability, parties output $\{{}^{A}\langle \mathbb{X}_i \rangle_d, {}^{A}\langle \mathbb{Y}_i \rangle_d, {}^{A}\langle \mathbb{Z}_i \rangle_d\}$, where $i \in \{1, \ldots, t\}$, such that $\mathbb{X}_i, \mathbb{Y}_i, \mathbb{Z}_i \in \mathbb{F}^{n^3p^2}$ and $\mathbb{Z}_i = \mathbb{X}_i \times \mathbb{Y}_i$.*

Proof. If D is discarded during step 13 then the lemma holds trivially, since in this case $\mathbb{X}_i, \mathbb{Y}_i$ and $\mathbb{Z}_i$ are all 0's. So we consider the case when D is not discarded. In this case, we show that except with a negligible probability, the vectors of polynomials $\mathbb{X}(\cdot), \mathbb{Y}(\cdot)$ and $\mathbb{Z}(\cdot)$ defined by the end of step 9, satisfy the multiplicative relationship. On contrary, if this is not the case, then component-wise $\mathbb{Z}(r) \neq \mathbb{X}(r) \times \mathbb{Y}(r)$ will hold, except with a negligible probability. This is because r is selected randomly from $\mathbb{F}$ and not known to the adversary at the end of step 9, when the polynomials are determined. Moreover, the vectors of polynomials $\mathbb{Z}(\cdot) - \mathbb{X}(\cdot) \times \mathbb{Y}(\cdot)$ can have at most $2t$ roots, since each polynomial here is a degree-$2t$ polynomial.

To show that $\mathbb{X}(\cdot), \mathbb{Y}(\cdot)$ and $\mathbb{Z}(\cdot)$ satisfy the multiplicative relationship, we need to show that for every $\gamma \in \{1, \ldots, 2t+1\}$, the vectors of triples $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ computed in the protocol are multiplication-triples. Note that $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ is the same as $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ shared by D, when $\gamma \in \{1, \ldots, t+1\}$. If $\gamma \in \{t+$

$2, \ldots, 2t+1\}$, then $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ depends upon the triples $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ shared by D.

Consider an arbitrary $\gamma \in \{1, \ldots, t+1\}$. If $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ do not constitute multiplication-triples then $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ will also be non-multiplication-triples. On the other hand, consider any $\gamma \in \{t+2, \ldots, 2t+1\}$. Here, $\mathbb{Z}^{(\gamma)}$ is computed from the inputs $\mathbb{X}^{(\gamma)}$ and $\mathbb{Y}^{(\gamma)}$ through Π_{Beaver} using the triple $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ shared by D. It follows from the correctness of Π_{Beaver} (Lemma F.25) that if $(\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)})$ is not a multiplication-triple, then $(\mathbb{X}^{(\gamma)}, \mathbb{Y}^{(\gamma)}, \mathbb{Z}^{(\gamma)})$ will also be a non-multiplication-triple. □

Lemma F.32 (Complexity). Π_{TripSh} *requires a constant expected rounds and makes at most* 47 *number of calls to* $\mathcal{F}_{\mathsf{Rand}}$. *It generates* tn^3p^2 *packed-secret-shared multiplication-triples, incurring a communication of* $\mathsf{P2P}(n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits.*

If the protocol is executed L *times in parallel, then it outputs* $L \cdot tn^3p^2$ *packed-secret-shared multiplication-triples over* $\mathbb{F}$, *requires a constant expected number of rounds, incurs a communication of* $\mathsf{P2P}(L \cdot n^6\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits and calls* $\mathcal{F}_{\mathsf{Rand}}$ *at most* 47 *times.*

Proof. During steps 1-2, D needs to generate ${}^{A}\langle\cdot\rangle_d$-sharings of $\mathcal{O}(n)$ number of vectors of n^3p^2 elements from $\mathbb{F}$ through instances of $\Pi_{{}^{A}\langle\cdot\rangle}$. From Lemma D.7, after securely realizing broadcasts over point-to-point channels, this incurs a constant expected number of rounds and a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$. Note that all the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ can be executed in parallel and so the calls to $\mathcal{F}_{\mathsf{Rand}}$ inside various instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ can be synchronized across various stages of the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$. Hence, the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ will require at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$.

During step 3, the parties need to run a constant number of instances of Π_{RandPair} to generate the required number of double-secret-shared random values. From Lemma F.16, this requires a constant expected number of rounds and incur a communication of $\mathsf{P2P}(n^6\kappa)$ bits, apart from at most 16 calls to $\mathcal{F}_{\mathsf{Rand}}$ since all the instances can be executed in parallel. We also note that steps 1-3 can be executed in parallel and so all the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$, required as part of steps 1-2 and the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ inside Π_{RandPair} can be executed in parallel. Consequently, the calls to $\mathcal{F}_{\mathsf{Rand}}$ during various stages of various instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ can be synchronized. So the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ across steps 1-3 is at most 16.

To transform the dealer's secret-shared triples, parties need to run an instance of Π_{Beaver}. From Lemma F.27, this requires a constant expected number of rounds, communication of $\mathsf{P2P}(n^6\kappa)$ bits and at most 31 number of calls to $\mathcal{F}_{\mathsf{Rand}}$. Finally during step 12, the parties need to publicly reconstruct a ${}^{A}\langle\cdot\rangle_d$-shared vectors of n^3p^2 values, which costs one round and incurs a communication of $\mathsf{P2P}(n^6\kappa)$ bits.

From the above analysis, Π_{TripSh} requires a constant expected rounds, incurs a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits and requires

at most 47 number of calls to $\mathcal{F}_{\mathsf{Rand}}$. If Π_{TripSh} is executed L times in parallel, then the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be synchronized across all the L instances and the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is at most 47. Moreover, $L \cdot tn^3p^2$ multiplication-triples generated will be generated on the behalf of the dealer. □

Communication Overhead of Π_{TripSh}. If Π_{TripSh} is executed L times in parallel, it outputs $L \cdot tn^3p^2 = \Theta(n^6)$ multiplication-triples, with communication complexity $\mathsf{P2P}(L \cdot n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 47 calls to $\mathcal{F}_{\mathsf{Rand}}$. It follows that for sufficiently large L, the communication overhead is a constant.

We next describe the simulation steps for Π_{TripSh} which will be used when the protocol is used as a primitive.

Simulator F.33: Simulator for the Protocol Π_{TripSh}

Case 1: Honest Dealer

- On behalf of the dealer, for each $\gamma \in \{1, \ldots, 2t+1\}$, select vectors of random elements $\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)}$ where each $\mathbb{A}^{(\gamma)}, \mathbb{B}^{(\gamma)}, \mathbb{C}^{(\gamma)} \in \mathbb{F}^{n^3p^2}$ and where component-wise, $\mathbb{C}^{(\gamma)} = \mathbb{A}^{(\gamma)} \times \mathbb{B}^{(\gamma)}$.
- Simulate the steps of VSS for an honest dealer as per Simulator D.8 to generate ${}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d$, ${}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d$.
- On behalf of the dealer, for each $\gamma \in \{t+2, \ldots, 2t+1\}$, select vectors of random elements $\hat{\mathbb{A}}^{(\gamma)}, \hat{\mathbb{B}}^{(\gamma)}, \hat{\mathbb{C}}^{(\gamma)}$ from $\mathbb{F}^{n^3p^2}$.
- Simulate the steps of VSS for an honest dealer as per Simulator D.8 to generate ${}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d$, ${}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_d$.
- Simulate the steps of Π_{RandPair} as in Simulation F.17.
- Perform the local computations as in the protocol steps on behalf of the honest parties and invoke the simulation steps of Π_{Beaver} as in Simulation F.28.
- Select a random r to emulate $\mathcal{F}_{\mathsf{Rand}}$ and send r to each P_k under $\mathcal{A}$'s control.
- Send the honest parties shares corresponding to ${}^{A}\langle \mathbb{X}(r) \rangle_d$, ${}^{A}\langle \mathbb{Y}(r) \rangle_d$ and ${}^{A}\langle \mathbb{Z}(r) \rangle_d$.

Case 2: Corrupt Dealer

- Simulate the steps of VSS for a corrupt dealer as per Simulator D.8 to obtain ${}^{A}\langle \mathbb{A}^{(\gamma)} \rangle_d$, ${}^{A}\langle \mathbb{B}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \mathbb{C}^{(\gamma)} \rangle_d$ for each $\gamma \in \{1, \ldots, 2t+1\}$ and ${}^{A}\langle \hat{\mathbb{A}}^{(\gamma)} \rangle_d$, ${}^{A}\langle \hat{\mathbb{B}}^{(\gamma)} \rangle_d$ and ${}^{A}\langle \hat{\mathbb{C}}^{(\gamma)} \rangle_d$ for each $\gamma \in \{t+2, \ldots, 2t+1\}$.
- Simulate the steps of Π_{RandPair} as in Simulation F.17.
- Perform the local computations as in the protocol steps on behalf of the honest parties and invoke the simulation steps of Π_{Beaver} as in Simulation F.28.
- Select a random r to emulate $\mathcal{F}_{\mathsf{Rand}}$ and send r to each P_k under the control of $\mathcal{A}$.
- Send the honest parties shares corresponding to ${}^{A}\langle \mathbb{X}(r) \rangle_d$, ${}^{A}\langle \mathbb{Y}(r) \rangle_d$ and ${}^{A}\langle \mathbb{Z}(r) \rangle_d$.
- If $\mathbb{Z}(r) \neq \mathbb{X}(r) \times \mathbb{Y}(r)$ then discard the dealer on behalf of the honest parties and assume a default sharing of 0's corresponding to the corrupt dealer.

F.7 Triple-Extraction with Constant Overhead

Our final component is a triple extraction protocol Π_{TripExt} (Protocol F.34). The protocol takes input vectors of multiplication-triples shared by individual parties where it will be guaranteed that the multiplication-triples shared by the honest parties are truly random and unknown to the adversary. The protocol outputs vectors of random multiplication-triples unknown to the adversary.

In more detail, for $\gamma \in \{1, \ldots, \frac{3t}{2}+1\}$, the input of party P_γ will be ${}^{A}\langle\cdot\rangle_d$-shared vectors of multiplication-triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$, where $\beta \in \{1, \ldots, n\}$. On the other hand, for $\gamma \in \{\frac{3t}{2}+2, \ldots, n\}$, party P_γ will have input ${}^{A}\langle\cdot\rangle_d$-shared random multiplication-triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ and ${}^{A}\langle\cdot\rangle_d$-shared random triples $(\hat{\mathbb{A}}^{(\beta,\gamma)}, \hat{\mathbb{B}}^{(\beta,\gamma)}, \hat{\mathbb{C}}^{(\beta,\gamma)})$. It would be ensured that if P_γ is *honest* then adversary's view remains independent of P_γ's inputs. The idea is to transform the multiplication-triples in $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ into multiplication-triples $(\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)})$, such that the elements in $\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}$ and $\mathbb{Z}^{(\beta,\gamma)}$ define vectors of degree-$\frac{3t}{2}$, $\frac{3t}{2}$ and degree-$3t$ polynomials $\mathbb{X}^{(\beta)}(\cdot), \mathbb{Y}^{(\beta)}(\cdot)$ and $\mathbb{Z}^{(\beta)}(\cdot)$ respectively, with adversary learning at most t vectors of points on these polynomials. This is done using a similar idea as used in the protocol Π_{TripSh}. Note that the polynomials will satisfy the multiplicative relationship as the input triples of all the parties will be guaranteed to be multiplication-triples. Once the polynomials $\mathbb{X}^{(\beta)}(\cdot), \mathbb{Y}^{(\beta)}(\cdot)$ and $\mathbb{Z}^{(\beta)}(\cdot)$ are set, the parties extend these polynomials at $\frac{t}{2}+1$ new points and output ${}^{A}\langle\cdot\rangle_d$-sharings of these vectors of elements.

Protocol F.34: Packed Triple Extraction – Π_{TripExt}

Input:

- For $(\beta,\gamma) \in \{1,\ldots,n\}\times\{1,\ldots,\frac{3t}{2}+1\}$, party P_γ has the vectors of multiplication-triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ and parties hold their corresponding sharings ${}^{A}\langle\mathbb{A}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\beta,\gamma)}\rangle_d$.
- For $(\beta,\gamma) \in \{1,\ldots,n\}\times\{\frac{3t}{2}+2,\ldots,n\}$, P_γ has the vectors of multiplication-triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ and triples $(\hat{\mathbb{A}}^{(\beta,\gamma)}, \hat{\mathbb{B}}^{(\beta,\gamma)}, \hat{\mathbb{C}}^{(\beta,\gamma)})$ and parties hold ${}^{A}\langle\mathbb{A}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\beta,\gamma)}\rangle_d$ and ${}^{A}\langle\hat{\mathbb{A}}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}^{(\beta,\gamma)}\rangle_d$.

(Generating Double-Packed-Secret-Shared Random Values)

1. For each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{\frac{3t}{2}+2,\ldots,n\}$, parties generate $({}^{A}\langle\mathbb{R}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{R}^{(\beta,\gamma)}\rangle_{d+p-1})$ and $({}^{A}\langle\hat{\mathbb{R}}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{R}}^{(\beta,\gamma)}\rangle_{d+p-1})$ through instances of protocol Π_{RandPair}.

(Transformation to Correlated Triples)

2. For each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{1,\ldots,\frac{3t}{2}+1\}$, parties locally set
   - ${}^{A}\langle\mathbb{X}^{(\beta,\gamma)}\rangle_d = {}^{A}\langle\mathbb{A}^{(\beta,\gamma)}\rangle_d$, ${}^{A}\langle\mathbb{Y}^{(\beta,\gamma)}\rangle_d = {}^{A}\langle\mathbb{B}^{(\beta,\gamma)}\rangle_d$, ${}^{A}\langle\mathbb{Z}^{(\beta,\gamma)}\rangle_d = {}^{A}\langle\mathbb{C}^{(\beta,\gamma)}\rangle_d$.
3. For each $\beta \in \{1,\ldots,n\}$, let $\mathbb{X}^{(\beta)}(\cdot)$ and $\mathbb{Y}^{(\beta)}(\cdot)$ be the vectors of n^3p^2 degree-$\frac{3t}{2}$ polynomials, passing through the vectors of points from $\{(\gamma, \mathbb{X}^{(\beta,\gamma)})\}$ and $\{(\gamma, \mathbb{Y}^{(\beta,\gamma)})\}$ respectively, where $\gamma \in \{1,\ldots,\frac{3t}{2}+1\}$.

4. For each $\gamma \in \{\frac{3t}{2}+2, \ldots, n\}$, parties locally compute
   - ${}^{A}\langle \mathbb{X}^{(\beta,\gamma)} \rangle_d = {}^{A}\langle \mathbb{X}^{(\beta)}(\gamma) \rangle_d$ from ${}^{A}\langle \mathbb{X}^{(\beta,1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{X}^{(\beta,\frac{3t}{2}+1)} \rangle_d$, and
   - ${}^{A}\langle \mathbb{Y}^{(\beta,\gamma)} \rangle_d = {}^{A}\langle \mathbb{Y}^{(\beta)}(\gamma) \rangle_d$ from ${}^{A}\langle \mathbb{Y}^{(\beta,1)} \rangle_d, \ldots, {}^{A}\langle \mathbb{Y}^{(\beta,\frac{3t}{2}+1)} \rangle_d$.
5. For each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{\frac{3t}{2}+2, \ldots, n\}$, parties invoke instances of Π_{Beaver} (two times as described below) with inputs ${}^{A}\langle \mathbb{X}^{(\beta,\gamma)} \rangle_d$, ${}^{A}\langle \mathbb{Y}^{(\beta,\gamma)} \rangle_d$ and auxiliary inputs $\{{}^{A}\langle \mathbb{A}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \mathbb{B}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \mathbb{C}^{(\beta,\gamma)} \rangle_d\}$, $\{{}^{A}\langle \mathbb{R}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \mathbb{R}^{(\beta,\gamma)} \rangle_{d+p-1}\}$, $\{{}^{A}\langle \hat{\mathbb{A}}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{B}}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{C}}^{(\beta,\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \hat{\mathbb{R}}^{(\beta,\gamma)} \rangle_d, {}^{A}\langle \hat{\mathbb{R}}^{(\beta,\gamma)} \rangle_{d+p-1}\}$ to compute ${}^{A}\langle \mathbb{Z}^{(\beta,\gamma)} \rangle_d$, where $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} \times \mathbb{Y}^{(\beta,\gamma)}$ component-wise.
   The first and second instance of Π_{Beaver} will take inputs indexed from $\gamma \in \{\frac{3t}{2}+2, \frac{5t}{2}+2\}$ and $\gamma \in \{\frac{5t}{2}+3, \ldots, n\}$ respectively.
6. For each $\beta \in \{1,\ldots,n\}$, let $\mathbb{Z}^{(\beta)}(\cdot)$ be the vectors of n^3p^2 degree-$(n-1)$ polynomials defined by the vectors of points $(1, \mathbb{Z}^{(\beta,1)}), \ldots, (n, \mathbb{Z}^{(\beta,n)})$.

(Output Computation)

7. For each $(\beta,i) \in \{1,\ldots,n\} \times \{1, \frac{t}{2}+1\}$, let $\mathbb{A}_i^{(\beta)} = \mathbb{X}^{(\beta)}(n+i), \mathbb{B}_i^{(\beta)} = \mathbb{Y}^{(\beta)}(n+i)$ and $\mathbb{C}_i^{(\beta)} = \mathbb{Z}^{(\beta)}(n+i)$. Parties locally compute ${}^{A}\langle \mathbb{A}_i^{(\beta)} \rangle_d$, ${}^{A}\langle \mathbb{B}_i^{(\beta)} \rangle_d$ and ${}^{A}\langle \mathbb{C}_i^{(\beta)} \rangle_d$ from $\{{}^{A}\langle \mathbb{X}^{(\beta,\gamma)} \rangle_d\}$, $\{{}^{A}\langle \mathbb{Y}^{(\beta,\gamma)} \rangle_d\}$ and $\{{}^{A}\langle \mathbb{Z}^{(\beta,\gamma)} \rangle_d\}$ respectively, where $\gamma \in \{1,\ldots,n\}$ and output ${}^{A}\langle \mathbb{A}_i^{(\beta)} \rangle_d$, ${}^{A}\langle \mathbb{B}_i^{(\beta)} \rangle_d$ and ${}^{A}\langle \mathbb{C}_i^{(\beta)} \rangle_d$.

Lemma F.35 (Correctness). *For each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{1,\ldots,\frac{3t}{2}+1\}$, let $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ be multiplication-triples which are ${}^{A}\langle \cdot \rangle_d$-shared among the parties. Moreover, for each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{\frac{3t}{2}+2, \ldots, n\}$, let $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ be multiplication-triples which are ${}^{A}\langle \cdot \rangle_d$-shared among the parties. Then except with a negligible probability, for each $(\beta,i) \in \{1,\ldots,n\} \times \{1,\ldots,\frac{t}{2}+1\}$, protocol Π_{TripExt} outputs ${}^{A}\langle \cdot \rangle_d$-sharing of multiplication-triples $(\mathbb{A}_i^{(\beta)}, \mathbb{B}_i^{(\beta)}, \mathbb{C}_i^{(\beta)})$.*

Proof. We show that for each $\beta \in \{1,\ldots,n\}$, the condition $\mathbb{Z}^{(\beta)}(\cdot) = \mathbb{X}^{(\beta)}(\cdot) \times \mathbb{Y}^{(\beta)}(\cdot)$ holds for the vectors of polynomials $\mathbb{X}^{(\beta)}(\cdot), \mathbb{Y}^{(\beta)}(\cdot)$ and $\mathbb{Z}^{(\beta)}(\cdot)$.

From the protocol steps, $\mathbb{Z}^{(\beta)}(\gamma) = \mathbb{X}^{(\beta)}(\gamma) \times \mathbb{Y}^{(\beta)}(\gamma)$ holds component-wise, for each $\gamma \in \{1,\ldots,\frac{3t}{2}+1\}$. This is because for each $\gamma \in \{1,\ldots,\frac{3t}{2}+1\}$, we have $(\mathbb{X}^{(\beta)}(\gamma), \mathbb{Y}^{(\beta)}(\gamma), \mathbb{Z}^{(\beta)}(\gamma)) = (\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)}) = (\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ and it is given that $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ is a multiplication-triple. On the other hand, for each $\gamma \in \{\frac{3t}{2}+2, \ldots, n\}$, we have $(\mathbb{X}^{(\beta)}(\gamma), \mathbb{Y}^{(\beta)}(\gamma), \mathbb{Z}^{(\beta)}(\gamma)) = (\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)})$, where $\mathbb{Z}^{(\beta,\gamma)}$ is computed from inputs $\mathbb{X}^{(\beta,\gamma)}$ and $\mathbb{Y}^{(\beta,\gamma)}$ using protocol Π_{Beaver}. During the instance of Π_{Beaver}, the triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ are used as auxiliary triples, which are given to be multiplication-triples. From the correctness of Π_{Beaver}, it follows that except with a negligible probability, $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} \times \mathbb{Y}^{(\beta,\gamma)}$ will hold. Consequently, $(\mathbb{X}^{(\beta)}(\gamma), \mathbb{Y}^{(\beta)}(\gamma), \mathbb{Z}^{(\beta)}(\gamma))$ will constitute multiplication-triples, even for $\gamma \in \{\frac{3t}{2}+2, \ldots, n\}$. □

Lemma F.36 (Privacy). *For each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{1,\ldots,\frac{3t}{2}+1\}$, let $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ be multiplication-triples held by party P_γ, which are ${}^{A}\langle \cdot \rangle_d$-shared among the parties. Moreover, for each $(\beta,\gamma) \in \{1,\ldots,n\} \times \{\frac{3t}{2}+2, \ldots, n\}$,*

let $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ *be the multiplication-triples and let* $(\hat{\mathbb{A}}^{(\beta,\gamma)}, \hat{\mathbb{B}}^{(\beta,\gamma)}, \hat{\mathbb{C}}^{(\beta,\gamma)})$ *be the triples held by party* P_γ*, which are* ${}^{A}\langle\cdot\rangle_d$*-shared among the parties. Furthermore, if* P_γ *is honest then let the triples and multiplication-triples held by* P_γ *be random for the adversary. Then for each* $(\beta, i) \in \{1, \ldots, n\} \times \{1, \ldots, \frac{t}{2} + 1\}$*, the view of the adversary in the protocol* Π_{TripExt} *will be independent of the output multiplication-triples* $(\mathbb{A}_i^{(\beta)}, \mathbb{B}_i^{(\beta)}, \mathbb{C}_i^{(\beta)})$*.*

Proof. From the proof of Lemma F.35, for every $\beta \in \{1, \ldots, n\}$, the vectors of polynomials $\mathbb{X}^{(\beta)}(\cdot), \mathbb{Y}^{(\beta)}(\cdot)$ and $\mathbb{Z}^{(\beta)}(\cdot)$ satisfy the multiplicative relationship, where these polynomials have degree-$\frac{3t}{2}$, $\frac{3t}{2}$ and degree-$3t$ respectively. Also, from the protocol steps, for every $\gamma \in \{1, \ldots, n\}$, the conditions $\mathbb{X}^{(\beta)}(\gamma) = \mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta)}(\gamma) = \mathbb{Y}^{(\beta,\gamma)}$ and $\mathbb{Z}^{(\beta)}(\gamma) = \mathbb{Z}^{(\beta,\gamma)}$ hold. To prove the lemma, we show that throughout the protocol, adversary learns at most t points on these vectors of polynomials. For this consider any arbitrary *honest* party P_γ. We claim that throughout the protocol, the view of the adversary remains independent of the vectors of multiplication-triples $(\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)})$. This will imply that adversary may learn the vectors of multiplication-triples $(\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)})$ only if P_γ is corrupt. Since there can be at most t corrupt parties, the adversary may learn at most t points on the vectors of polynomials.

To prove the above claim, we consider two cases. If $\gamma \in \{1, \ldots, \frac{3t}{2}+1\}$, then it is obvious that the view of the adversary remains independent of $(\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)})$. This is because from the protocol steps, we have $(\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}, \mathbb{Z}^{(\beta,\gamma)}) = (\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$, where the elements of $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ are given to be random for the adversary as per the lemma conditions. The other possibility is when $\gamma \in \{\frac{3t}{2}+2, \ldots, n\}$. In this case, $\mathbb{Z}^{(\beta,\gamma)}$ is computed from $\mathbb{X}^{(\beta,\gamma)}$ and $\mathbb{Y}^{(\beta,\gamma)}$ through an instance of Π_{Beaver}. During the instance of Π_{Beaver}, parties use the multiplication-triples $(\mathbb{A}^{(\beta,\gamma)}, \mathbb{B}^{(\beta,\gamma)}, \mathbb{C}^{(\beta,\gamma)})$ and the triples $(\hat{\mathbb{A}}^{(\beta,\gamma)}, \hat{\mathbb{B}}^{(\beta,\gamma)}, \hat{\mathbb{C}}^{(\beta,\gamma)})$ as auxiliary information. From the lemma conditions, these are given to be random for the adversary. Apart from the above auxiliary information, the parties also use vectors of double-packed-secret-shared elements $\mathbb{R}^{(\beta,\gamma)}$ and $\hat{\mathbb{R}}^{(\beta,\gamma)}$, which are generated through instances of Π_{RandPair}. From the privacy property of Π_{RandPair} (Lemma F.15), these elements will be random for the adversary. It now follows from the privacy property of Π_{Beaver} (Lemma F.26) that during the instance of Π_{Beaver}, the view of the adversary will remain independent of $\mathbb{X}^{(\beta,\gamma)}, \mathbb{Y}^{(\beta,\gamma)}$ and $\mathbb{Z}^{(\beta,\gamma)}$, thus completing the proof. □

Lemma F.37 (Complexity). *Given inputs* $\{{}^{A}\langle\mathbb{A}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\beta,\gamma)}\rangle_d\}$ *where* $(\beta, \gamma) \in \{1, \ldots, n\} \times \{1, \ldots, \frac{3t}{2} + 1\}$ *and inputs* $\{{}^{A}\langle\mathbb{A}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\beta,\gamma)}\rangle_d\}$ *and* $\{{}^{A}\langle\hat{\mathbb{A}}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}^{(\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}^{(\beta,\gamma)}\rangle_d\}$*, where* $(\beta, \gamma) \in \{1, \ldots, n\} \times \{\frac{3t}{2} + 2, \ldots, n\}$*, protocol* Π_{TripExt} *outputs* ${}^{A}\langle\cdot\rangle_d$*-shared vectors of multiplication-triples* $(\mathbb{A}_i^{(\beta)}, \mathbb{B}_i^{(\beta)}, \mathbb{C}_i^{(\beta)})$*, where* $\beta \in \{1, \ldots, n\}$*,* $i \in \{1, \ldots, \frac{t}{2} + 1\}$ *and where each* $\mathbb{A}_i^{(\beta)}, \mathbb{B}_i^{(\beta)}, \mathbb{C}_i^{(\beta)} \in \mathbb{F}^{n^3p^2}$*. The protocol requires a constant expected number of rounds and a communication of* $\mathsf{P2P}(n^7\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits, apart from at most* 47 *number of calls to* $\mathcal{F}_{\mathsf{Rand}}$*.*

If the protocol is executed L times in parallel where $L \geq 1$, then the protocol outputs $L \cdot n(\frac{t}{2}+1)n^3p^2$ secret-shared multiplication-triples over $\mathbb{F}$, incurring a communication of $\mathsf{P2P}(L \cdot n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 47 number of calls to $\mathcal{F}_{\mathsf{Rand}}$.

Proof. In the protocol, the parties interact only during the instances of Π_{RandPair} and Π_{Beaver}, both of which require a constant expected number of rounds. Parties need to run $\Theta(n)$ instances of Π_{RandPair}, since they need to generate $\Theta(n^2)$ double-packed-secret-shared vectors of random elements and each instance can generate $\Theta(n)$ such vectors. All these instances can be executed in parallel and the calls to $\mathcal{F}_{\mathsf{Rand}}$ can be reused across various stages of different instances of Π_{RandPair}. Hence, from Lemma F.16, the instances of Π_{RandPair} will require a constant expected number of rounds and incur a communication of $\mathsf{P2P}(n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 16 calls to $\mathcal{F}_{\mathsf{Rand}}$.

Next, there will be $2n$ instances of Π_{Beaver}, both of which can be executed in parallel. From Lemma F.27, this requires a constant expected number of rounds, incurs a communication of $\mathsf{P2P}(n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits and requires at most 31 number of calls to $\mathcal{F}_{\mathsf{Rand}}$.

If the protocol is executed L times in parallel, then it outputs $L \cdot n(\frac{t}{2}+1)n^3p^2$ secret-shared multiplication-triples over $\mathbb{F}$. The maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ remains 47, as they can be synchronized across all L instances. □

Communication Overhead of Π_{TripExt}. If Π_{TripExt} is executed L times in parallel, it outputs $L \cdot n(\frac{n}{2}+1)n^3p^2 = \Theta(n^7)$ secret-shared multiplication-triples, with communication complexity $\mathsf{P2P}(L \cdot n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, apart from at most 47 calls to $\mathcal{F}_{\mathsf{Rand}}$. So for a large L, the communication overhead is a constant.

We describe the simulation steps for Π_{TripExt} when used as a primitive.

Simulator F.38: Simulator for the Protocol Π_{TripExt}

- Simulate the steps of Π_{RandPair} as in Simulation F.17.
- On behalf of each honest party P_i not controlled by $\mathcal{A}$, perform the local computations as an honest party would as per the protocol specifications.
- Simulate the steps of Π_{Beaver} as in Simulation F.28 and perform local computations as per the protocol specifications on behalf of each honest party P_i not under the control of $\mathcal{A}$.

F.8 The Preprocessing Phase Protocol

We now give a protocol Π_{Off} in the $\mathcal{F}_{\mathsf{MacKeys}}$-hybrid model for the preprocessing phase, as a composition of the building blocks defined so far. Parties begin by invoking the $\mathcal{F}_{\mathsf{MacKeys}}$ functionality for setup generation, that is, establishing the MAC key required for further computation. The setup is used across all the

instances of $\Pi_{A\langle\cdot\rangle}$ used inside various sub-protocols and thus guarantees that all the ${}^{A}\langle\cdot\rangle_d$-sharings generated are key-consistent.[16] Subsequently, the goal is to generate shares of ${}^{A}\langle\cdot\rangle_d$-shared random multiplication triples, random triples and double-packed-secret-shared vectors of random elements. Recall that $\mathcal{F}_{\mathsf{Off}}$ generates $c_M n(t+1)$ vectors of random multiplication-triples and random triples, each of size n^3p^2. Along with this, the functionality also generates $2c_M n(t+1)$ vectors of double-packed-secret-shared random values, each of size n^3p^2.

The multiplication-triples are generated by applying the framework of [16], extending it for authenticated packed-secret-sharing. Namely, each party generates ${}^{A}\langle\cdot\rangle_d$-sharing of "sufficiently many" random multiplication-triples by acting as a dealer through instances of Π_{TripSh}. The multiplication-triples shared by honest parties will be random for the adversary. However, the identity of honest parties will not be known and consequently, parties run instances of Π_{TripExt} on the multiplication-triples shared by different parties and extract ${}^{A}\langle\cdot\rangle_d$-sharing of truly random multiplication-triples, unknown to the adversary. To calculate the number of vectors of multiplication-triples that each party should provide, let us first compute the number of instances of Π_{TripExt} that the parties need to run to generate the desired number of multiplication-triples.

Recall that one instance of Π_{TripExt} generates $n(\frac{t}{2}+1)n^3p^2$-secret-shared random multiplication-triples. Consequently, $2c_M$ instances of Π_{TripExt} will suffice to generate $c_M n(t+1)n^3p^2$-secret-shared random multiplication-triples. To run one instance of Π_{TripExt}, we need n number of ${}^{A}\langle\cdot\rangle_d$-sharing of vectors of n^3p^2 multiplication-triples and triples from each party and so for $2c_M$ instances, we need $2nc_M$ such triples from each party.[17] Now based on this calculation, let us calculate the number of instances of Π_{TripSh} that each party has to run as a dealer.

Recall that one instance of Π_{TripSh} generates tn^3p^2 number of secret-shared multiplication-triples on behalf of the designated dealer. Consequently, each party has to invoke $\frac{2nc_M}{t} = \Theta(c_M)$ instances of Π_{TripSh} as a dealer, so that $2nc_M n^3p^2$ secret-shared multiplication-triples are available on behalf of that party.

To generate $c_M n(t+1)$ vectors of random secret-shared triples, the parties run $\frac{c_M}{n-t}n(t+1)$ instances of the protocol $\Pi_{\mathsf{RandTriples}}$. This is because one instance of $\Pi_{\mathsf{RandTriples}}$ generates $n-t$ vectors of secret-shared random triples. Finally, to generate $2c_M n(t+1)$ vectors of double-packed-secret-shared vectors of random values, the parties need to run the protocol Π_{RandPair} $\frac{2c_M}{(n-t)}n(t+1)$ times, as one instance of Π_{RandPair} generates $(n-t)$ such vectors.

Protocol F.39: The Preprocessing Phase Protocol – Π_{Off}

[16] Recall that $\Pi_{A\langle\cdot\rangle}$ requires this setup to be already available.

[17] We need the first $\frac{3t}{2}+1$ parties to provide only n number of vectors of secret-shared multiplication-triples and the remaining $n-(\frac{3t}{2}+1)$ parties to provide n number of vectors of secret-shared multiplication-triples and triples. For uniformity, we ask for n number of vectors of secret-shared multiplication-triples and triples from each party. Asymptotically, the communication cost still remains the same.

1. **Mack-Key Setup Generation:** Every $P_i \in \mathcal{P}$ interacts with $\mathcal{F}_{\mathsf{MacKeys}}$ (Functionality F.2) with input SetUp and receives from it the following.
   - Vectors $\{\boldsymbol{\mu}_{ij}\}_{j=1}^{n}$ where each $\boldsymbol{\mu}_{ij} \in \mathbb{F}^{n^2p}$ and where $\boldsymbol{\mu}_{ij}$ is designated to be used in the MAC-key for party P_j.
   - For every (P_j, P_k), the twisted-shares of P_i corresponding to $[\boldsymbol{\mu}_{jk}]_t^k$.

   The parties use the above setup in all the instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ involved in the rest of the protocol.
2. **Generating double-packed-secret-shared Random Values**: The parties execute $\frac{2c_M}{(n-t)}n(t+1)$ instances of Π_{RandPair} (Protocol F.13) to generate packed-double-secret-sharing of $2c_Mn(t+1)n^3p^2$ random elements from $\mathbb{F}$.
3. **Generating Packed-Secret-Shared Random Triples**: The parties execute $\frac{c_M}{(n-t)}n(t+1)$ instances of $\Pi_{\mathsf{RandTriples}}$ (Protocol F.18) to generate packed-secret-sharing of $c_Mn(t+1)n^3p^2$ random triples from $\mathbb{F}$.
4. **Generating Packed-Secret-Shared Random Multiplication-Triples**:
   - Each $P_i \in \mathcal{P}$ acts as a dealer and invokes $\frac{2nc_M}{t}$ instances of Π_{TripSh} (Protocol F.29) and generates packed-secret-sharing of $2nc_Mn^3p^2$ random multiplication-triples over $\mathbb{F}$.
   - The multiplication-triples and the triples shared by different parties are grouped into $2c_M$ disjoint groups, where in each group, there are nn^3p^2 number of packed-secret-shared multiplication-triples from each party. The parties execute $2c_M$ instances of Π_{TripExt} (Protocol F.34), one on behalf of each group and generate packed-secret-sharing of $c_Mn(t+1)n^3p^2$ random multiplication-triples over $\mathbb{F}$ [18].

Lemma F.40 (Correctness). *Protocol Π_{Off} generates the following, except with a negligible probability:*

- $c_Mn(t+1)n^3p^2$ ${}^{A}\langle\cdot\rangle_d$*-shared multiplication-triples over* $\mathbb{F}$.
- $c_Mn(t+1)n^3p^2$ ${}^{A}\langle\cdot\rangle_d$*-shared triples over* $\mathbb{F}$.
- $2c_Mn(t+1)n^3p^2$ *double-packed-secret-shared elements over* $\mathbb{F}$.

Proof. In the protocol, the parties execute $\frac{2c_M}{(n-t)}n(t+1)$ instances of Π_{RandPair}. From Lemma F.14, except with a negligible probability, each instance outputs packed-double-secret-sharing of $(n-t)n^3p^2$ elements over $\mathbb{F}$. So $\frac{2c_M}{(n-t)}n(t+1)$ instances will output $2c_Mn(t+1)n^3p^2$ such elements. Additionally, the parties execute $\frac{c_M}{(n-t)}n(t+1)$ instances of $\Pi_{\mathsf{RandTriples}}$. From Lemma F.19, except with a negligible probability, each instance outputs $(n-t)n^3p^2$ ${}^{A}\langle\cdot\rangle_d$-shared triples over $\mathbb{F}$. So $\frac{c_M}{(n-t)}n(t+1)$ instances will output $c_Mn(t+1)n^3p^2$ such triples.

Finally, in the protocol, the parties execute $2c_M$ instances of Π_{TripExt} on the multiplication-triples secret-shared by various parties. From Lemma F.35, except

[18] Parties actually generate packed-secret-sharing of $c_Mn(t+2)n^3p^2$ multiplication-triples, however, we use only $c_Mn(t+1)n^3p^2$ as per our notation for simplicity.

with a negligible probability, each instance outputs $n(\frac{t}{2}+1)n^3p^2$ ${}^{A}\langle\cdot\rangle_d$-secret-shared multiplication-triples over $\mathbb{F}$. So $2c_M$ instances will output $c_M n(t+1)n^3p^2$ such multiplication-triples [19]. □

Lemma F.41 (Privacy). *In protocol* Π_{Off}*, the view of the adversary remains independent of the output secret-shared triples, secret-shared multiplication-triples and double-secret-shared elements.*

Proof. This follows from the privacy of Π_{RandPair} (Lemma F.15), $\Pi_{\mathsf{RandTriples}}$ (Lemma F.20), Π_{TripSh} (Lemma F.30), Π_{TripExt} (Lemma F.36) and $\Pi_{{}^{A}\langle\cdot\rangle}$ (Lemma D.3). □

Lemma F.42. *Protocol* Π_{Off} *requires a constant expected number of rounds, makes at most* 94 *calls to* $\mathcal{F}_{\mathsf{Rand}}$ *and incurs a communication of* $\mathsf{P2P}(c_M n^7\kappa)$ *bits plus expected* $\mathsf{P2P}(n^3\log^2 n)$ *bits.*

Proof. As each building block in the protocol requires a constant expected number of rounds, the protocol requires a constant expected number of rounds.

The number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is computed as follows: we first note that step 1, 2, 3, 4 can be executed in parallel. Hence, the number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is dominated by step 4, where the parties generate packed-secret-shared multiplication-triples. The instances of Π_{TripSh} require at most 47 calls to $\mathcal{F}_{\mathsf{Rand}}$ for all the dealers (from Lemma F.32). On the other hand, the instances of Π_{TripExt} will call $\mathcal{F}_{\mathsf{Rand}}$ at most 47 times. So the maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ is 94.

During step 2, parties run $\frac{2c_M}{(n-t)}n(t+1)$ instances of Π_{RandPair}. From Lemma F.16, by substituting $L = \frac{2c_M}{(n-t)}n(t+1)$, this incurs a communication of $\mathsf{P2P}(c_M n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits, as $t = \Theta(n)$. During step 3, parties run $L = \frac{c_M}{(n-t)}n(t+1)$ instances of $\Pi_{\mathsf{RandTriples}}$. From Lemma F.21, by substituting L, this incurs a communication of $\mathsf{P2P}(c_M n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits

During step 4, there are $\frac{2c_M n}{t}$ instances of Π_{TripSh} executed by each party. From Lemma F.32, by substituting $L = \frac{2c_M n}{t}$, this incurs a total cost of $\mathsf{P2P}(c_M n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits. Furthermore, there will be $2c_M$ instances of Π_{TripExt}. From Lemma F.37, by substituting $L = 2c_M$, this incurs a communication of $\mathsf{P2P}(c_M n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits. □

As discussed in Section B, N calls to $\mathcal{F}_{\mathsf{Rand}}$ can be emulated by a constant expected round protocol, incurring a communication of $\mathsf{P2P}(N\cdot n^2\kappa)+n\times\mathsf{BC}(n^2\kappa)$ bits. After securely realizing parallel broadcast, the total communication will be $\mathsf{P2P}(N\cdot n^2\kappa + n^4\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits. By substituting $N = 94$, the communication involved to emulate broadcasts is $\mathsf{P2P}(n^4\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits. The call to $\mathcal{F}_{\mathsf{MacKeys}}$ can be securely realized by using protocol Π_{MacKeys}, requiring a constant expected number of rounds and incurring a communication of $\mathsf{P2P}(n^6\kappa)$ bits plus expected $\mathsf{P2P}(n^3\log^2 n)$ bits.

[19] As mentioned earlier, this actually generates $c_M n(t+2)n^3p^2$ random multiplication-triples, however, we use only $c_M n(t+1)n^3p^2$ for simplicity.

So the overall communication cost of Π_{Off} in the plain model will be $\mathsf{P2P}(c_M n^7 \kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits.

The output of protocol Π_{Off} consists of $\Theta(c_M n^7)$ field elements and hence $\Theta(c_M n^7 \kappa)$ bits. Since the total communication cost is $\mathsf{P2P}(c_M n^7 \kappa)$ bits, it follows that protocol Π_{Off} has a constant communication overhead.

Theorem F.43. *Protocol Π_{Off} securely realizes the functionality $\mathcal{F}_{\mathsf{Off}}$ with statistical security in the $\mathcal{F}_{\mathsf{MacKeys}}$-hybrid model, in the presence of a malicious adversary, corrupting at most $t < \frac{n}{3}$ parties. The protocol requires a constant expected number of rounds and incurs a communication of $\mathsf{P2P}(c_M n^7 \kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits. The communication overhead of the protocol is a constant.*

Proof. We prove the above through a sequence of lemmas and claims below.

Lemma F.44. *Protocol Π_{Off} securely realizes $\mathcal{F}_{\mathsf{Off}}$ in $\mathcal{F}_{\mathsf{MacKeys}}$-hybrid model with statistical security. The protocol calls $\mathcal{F}_{\mathsf{MacKeys}}$ once.*

Proof. The number of calls to $\mathcal{F}_{\mathsf{MacKeys}}$ is straight-forward. We now give the simulator $\mathcal{S}_{\mathsf{Off}}$ for Π_{Off}. During the simulated preprocessing, the simulator itself performs the role of $\mathcal{F}_{\mathsf{MacKeys}}$, allowing it to learn the setup data that is supposed to be generated by the corrupt parties during $\mathcal{F}_{\mathsf{MacKeys}}$. The simulator then invokes $\mathcal{F}_{\mathsf{Off}}$ with this setup data. During simulation of Π_{RandPair} and $\Pi_{\mathsf{RandTriples}}$, whenever adversary invokes instances of $\Pi_{{}^{A}\langle\cdot\rangle}$ on behalf of a corrupt party, the simulator plays the roles of honest parties as per the steps of $\Pi_{{}^{A}\langle\cdot\rangle}$ and learns the entire ${}^{A}\langle\cdot\rangle_d$-sharings of the corrupt party. Moreover, the simulator ensures consistency of the setup information corresponding to the honest parties with that of the corrupt parties fixed in the prior stage. For the honest parties, the simulator picks arbitrary values as their inputs and simulates the steps of the honest parties as per the protocol. Due to the privacy guarantee of each primitive proven in lemmas F.15, F.20, F.30, D.3, the corrupt parties learn shares of random values unknown to them in each primitive in the preprocesing phase. Hence, their view remains independent of the underlying values corresponding to the random pairs, random triples and the multiplication triples. Hence, the simulator can ensure the indistinguishability of the adversary's view.

Simulator F.45: $\mathcal{S}_{\mathsf{Off}}$: Simulator for the Protocol Π_{Off}

$\mathcal{S}_{\mathsf{Off}}$ constructs virtual real-world honest parties and invokes the real-world adversary $\mathcal{A}$ who controls at most t parties. The simulator simulates the view of $\mathcal{A}$, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with $\mathcal{F}_{\mathsf{MacKeys}}$. In order to simulate $\mathcal{Z}$, the simulator forwards every message it receives from $\mathcal{Z}$ to $\mathcal{A}$ and vice-versa. The simulator then simulates the various steps of the protocol as follows.

(Simulating $\mathcal{F}_{\mathsf{MacKeys}}$): Emulate the functionality $\mathcal{F}_{\mathsf{MacKeys}}$ as follows:

- On behalf of every P_j under the control of $\mathcal{A}$:
  - Receive from $\mathcal{A}$ the n vectors $\{\boldsymbol{\mu}_{ji}\}_{i=1}^{n}$ where each $\boldsymbol{\mu}_{ji} \in \mathbb{F}^{n^2 p}$.

  - For every $\boldsymbol{\mu}_{ji}$, receive from $\mathcal{A}$ the sharing $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ such that the twisted-share of P_i is 0. If $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ is not a valid twisted t-sharing with twisted-shares of P_i being not 0, then send 0s as the input to the functionality.
  - Invoke $\mathcal{F}_{\mathsf{Off}}$ with the above values, receive from it the shares of every P_k under the control of $\mathcal{A}$ corresponding to $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ and send it to $\mathcal{A}$.
- On behalf of every party P_j not under the control of $\mathcal{A}$:
  - For every $P_i \in \mathcal{P}$, receive from $\mathcal{F}_{\mathsf{Off}}$ the shares of $\lceil \boldsymbol{\mu}_{ji} \rfloor_t^i$ corresponding to each P_k controlled by $\mathcal{A}$ and send it to $\mathcal{A}$.
  - Randomly pick t-degree polynomials $q_{ji}^{(1)}(\cdot), \ldots, q_{ji}^{(n^2p)}$, such that they are consistent with the shares of each P_k under the control of $\mathcal{A}$ received from $\mathcal{F}_{\mathsf{Off}}$ and $q_{ji}^{(\ell)}(0) = 0$ for all $\ell \in \{1, \ldots, n^2p\}$. Let $u_{ji}^{(\ell)} = q_{ji}^{(\ell)}(i)$.

In each of the following cases, extract the shares of the corrupt parties and forward them to $\mathcal{F}_{\mathsf{Off}}$. Also, choose the second component of the MAC key on behalf of each honest party randomly and simulate the following while ensuring consistency with the entire setup.
(Simulating Π_{RandPair}): Simulate the steps of Π_{RandPair} as in Simulation F.17.
(Simulating $\Pi_{\mathsf{RandTriples}}$): Simulate the steps of $\Pi_{\mathsf{RandTriples}}$ as in Simulation F.22.
(Simulating Packed-Secret-Shared Random Mult-Triples Generation)

- **(Simulating Π_{TripSh}):** Simulate the steps of Π_{TripSh} as in Simulation F.33.
- **(Simulating Random Triple Generation):** Simulate the steps of $\Pi_{A\langle\cdot\rangle}$ as in Simulation D.8 for an honest dealer for each party not under the control of $\mathcal{A}$ and for a corrupt dealer for each party P_k under the control of $\mathcal{A}$.
- **(Simulating Π_{TripExt}):** Simulate the steps of Π_{TripExt} as in Simulation F.38.

□

Claim F.46. *The view of $\mathcal{A}$ in the simulated execution with $\mathcal{S}_{\mathsf{Off}}$ is identically distributed as the view of $\mathcal{A}$ in the real execution of Π_{Off}.*

Proof. The view of $\mathcal{A}$ during the emulation of $\mathcal{F}_{\mathsf{MacKeys}}$ is identically distributed in both the executions, as $\mathcal{A}$ receives no messages from the honest parties and the steps of $\mathcal{F}_{\mathsf{MacKeys}}$ are executed by the simulator itself in the simulated execution. Namely, in both the executions, $\mathcal{A}$'s view consists of the MAC-key components of the parties in Cr and the twisted-shares of the MAC-key components of the honest parties. Let us fix all this data. Now conditioned on this data, for Π_{RandPair} and $\Pi_{\mathsf{RandTriples}}$, $\mathcal{A}$ learns its supposed data corresponding to the sharings $\{{}^A\langle \mathbb{R} \rangle_d, {}^A\langle \mathbb{R} \rangle_{d+p-1}\}$ and $\{{}^A\langle \hat{\mathbb{A}} \rangle_d, {}^A\langle \hat{\mathbb{B}} \rangle_d, {}^A\langle \hat{\mathbb{C}} \rangle_d\}$ during the real execution, while in the simulated execution it learns its supposed data corresponding to the sharings $\{{}^A\langle \widetilde{\mathbb{R}} \rangle_d, {}^A\langle \widetilde{\mathbb{R}} \rangle_{d+p-1}\}$ and $\{{}^A\langle \tilde{\hat{\mathbb{A}}} \rangle_d, {}^A\langle \tilde{\hat{\mathbb{B}}} \rangle_d, {}^A\langle \tilde{\hat{\mathbb{C}}} \rangle_d\}$, where the elements of $\widetilde{\mathbb{R}}, \tilde{\hat{\mathbb{A}}}, \tilde{\hat{\mathbb{B}}}, \tilde{\hat{\mathbb{C}}}$ are randomly chosen by the simulator. Owing to lemmas D.3, F.15, F.20, it is easy to see that the distributions of the data held by $\mathcal{A}$ in both the real and simulated executions are identical, as simulator honestly plays the role of honest parties in the underlying $\Pi_{A\langle\cdot\rangle}$ instances involving honest dealers. So let us fix the data held by $\mathcal{A}$ for both these protocols.

Similarly, during the instances of Π_{TripSh}, $\mathcal{A}$ learns its supposed data corresponding to the sharings $\{{}^{A}\langle\mathbb{X}\rangle_d, {}^{A}\langle\mathbb{Y}\rangle_d, {}^{A}\langle\mathbb{Z}\rangle_d\}$ during the real execution, while in the simulated execution it learns its data corresponding to the sharings $\{{}^{A}\langle\hat{\mathbb{X}}\rangle_d, {}^{A}\langle\hat{\mathbb{Y}}\rangle_d, {}^{A}\langle\hat{\mathbb{Z}}\rangle_d\}$, where $\hat{\mathbb{X}}, \hat{\mathbb{Y}}, \hat{\mathbb{Z}}$ are randomly chosen by the simulator.

As in the prior case, during random triple generation as well, the difference between the real and simulated execution corresponds to the simulator choosing the random triples in the latter case in contrast to the honest party choosing them in the real execution. Due to lemma F.30, it is thus easy to see that the distribution of data held by $\mathcal{A}$ is identical in both the cases, since the simulator follows the protocol steps honestly on behalf of the honest parties.

Finally, during triple extraction, the simulator uses the random triples it chose on behalf of the honest parties as the input, while ensuring that the shares of the corrupt parties are as obtained from the functionality $\mathcal{F}_{\mathsf{Off}}$. Since the simulator performs the role of the honest parties as in the real-world, following lemma F.36, the distribution in both the worlds is identically distributed. □

Claim F.47. *Conditioned on the view of $\mathcal{A}$, the output of the honest parties is identically distributed in the real execution of Π_{Off} involving $\mathcal{A}$, as well as in the ideal execution involving $\mathcal{S}_{\mathsf{Off}}$ and $\mathcal{F}_{\mathsf{Off}}$.*

Proof. Let view be an arbitrary view of $\mathcal{A}$. According to view, for every gate in each SIMD circuit, there exist double-packed secret-shared random values. Specifically, there exist $2c_M n^3(t+1)$ vectors of random values $\{\mathbb{R}\}$ where $\mathbb{R} \in \mathbb{F}^{n^3p^2}$, such that the parties hold $({}^{A}\langle\mathbb{R}\rangle_d, {}^{A}\langle\mathbb{R}\rangle_{d+p-1})$. Further, there exist $c_M n^3(t+1)$ vectors of random triples of the form $\{\hat{\mathbb{A}}, \hat{\mathbb{B}}, \hat{\mathbb{C}}\}$ where $\hat{\mathbb{A}}, \hat{\mathbb{B}}, \hat{\mathbb{C}} \in \mathbb{F}^{n^3p^2}$, such that parties hold $({}^{A}\langle\hat{\mathbb{A}}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}\rangle_d)$. Finally, there exist $c_M n^3(t+1)$ vectors of multiplication triples of the form $\{\mathbb{A}, \mathbb{B}, \mathbb{C}\}$ where $\mathbb{A}, \mathbb{B}, \mathbb{C} \in \mathbb{F}^{n^3p^2}$, such that parties hold $({}^{A}\langle\mathbb{A}\rangle_d, {}^{A}\langle\mathbb{B}\rangle_d, {}^{A}\langle\mathbb{C}\rangle_d)$. From Claim F.46, we have that the values $\mathbb{R}, (\hat{\mathbb{A}}, \hat{\mathbb{B}}, \hat{\mathbb{C}})$ and $(\mathbb{A}, \mathbb{B}, \mathbb{C})$ are uniformly distributed conditioned on the data for the corresponding sharings that is available to $\mathcal{A}$ as determined by the view.

It is easy to see that the output of the honest parties in the ideal-world correspond to shares of random double-packed pairs, random triples and random multiplication triples. We now show that the output of the honest parties in the real world is the same as that in the ideal world except with negligible probability. This follows directly from the correctness of Π_{Off} given in Lemma F.40, which in turn relies on the correctness of Π_{RandPair} (Lemma F.14), $\Pi_{\mathsf{RandTriples}}$ (Lemma F.19), Π_{TripSh} (Lemmas F.30, F.31), Π_{TripExt} (Lemma F.35). □

This completes the proof of Theorem F.43. □

G The MPC Protocol

We present our MPC protocol to securely realize the functionality $\mathcal{F}_{\mathsf{MPC}}$, which evaluates the circuit `ckt` in SIMD fashion over $n^4(t+1)p^2$ inputs. It receives $n^4(t+1)p^2$ values for the input x_i from every party P_i. These inputs are arranged

into n groups, where within each group there will be $(t+1)$ subgroups, with each subgroup consisting of n^3p^2 elements. The functionality then evaluates each gate over $n^4(t+1)p^2$ inputs in parallel. Once all the gates are evaluated, the functionality sends $n^4(t+1)p^2$ outputs to all.

Functionality G.1: $\mathcal{F}_{\mathsf{MPC}}$ – Evaluating circuit ckt in SIMD fashion

$\mathcal{F}_{\mathsf{MPC}}$ is parametrized by a function $f : \mathbb{F}^n \rightarrow \mathbb{F}$, where each $P_i \in \mathcal{P}$ has an input x_i for f and every party in $\mathcal{P}$ is supposed to receive the function output $o = f(x_1, \ldots, x_n)$. The function f is represented by a circuit ckt over $\mathbb{F}$. The functionality is also parametrized by the set Cr of corrupt parties. The functionality interacts with the parties in $\mathcal{P}$ and an adversary $\mathcal{S}$ as follows.

- For every $P_i \notin \mathsf{Cr}$, upon receiving $n^4(t+1)p^2$ values from $\mathbb{F}$ for the input x_i, arrange them as $\{\mathbb{X}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ and associate them with wire x_i, where $|\mathbb{X}_i^{(\beta,\gamma)}| = n^3p^2$.
- For every $P_i \in \mathsf{Cr}$, upon receiving $n^4(t+1)p^2$ values from $\mathbb{F}$ for the input x_i from $\mathcal{S}$, arrange them in the same way as in the previous step.
- For every gate $g \in$ ckt in the topological order, on having the inputs $\{\mathbb{X}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ and $\{\mathbb{Y}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ associated with x and y respectively, do the following.
  - **Addition Gate**: If g is an addition gate then compute $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} + \mathbb{Y}^{(\beta,\gamma)}$. Associate $\{\mathbb{Z}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ with wire z.
  - **Multiplication Gate**: If g is a multiplication-gate, then compute $\mathbb{Z}^{(\beta,\gamma)} = \mathbb{X}^{(\beta,\gamma)} \times \mathbb{Y}^{(\beta,\gamma)}$. Associate $\{\mathbb{Z}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ with wire z.
- On having the values $\{\mathbb{O}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ associated with the output wire o of ckt, send them to $\mathcal{S}$ and all the parties $P_i \notin \mathsf{Cr}$. .

The MPC protocol Π_{MPC} for securely realizing the functionality $\mathcal{F}_{\mathsf{MPC}}$ is given in Protocol G.2. The protocol is presented in $\mathcal{F}_{\mathsf{Off}}$-hybrid model.

Protocol G.2: Π_{MPC}: The MPC Protocol

Every party $P_i \in \mathcal{P}$ interacts with $\mathcal{F}_{\mathsf{Off}}$ with input $\mathsf{SetUp}, \mathsf{Multiplication-Triples}, \mathsf{Triples}$ and Double and receives the following.

- Vectors $\{\boldsymbol{\mu}_{ij}\}_{j=1}^n$ where each $\boldsymbol{\mu}_{ij} \in \mathbb{F}^{n^2p}$ and where $\boldsymbol{\mu}_{ij}$ is designated to be used in the MAC-key for party P_j.
- For every (P_j, P_k), the twisted-shares of P_i corresponding to $[\boldsymbol{\mu}_{jk}]_t^k$.
- For $\alpha \in \{1, \ldots, c_M\}, \beta \in \{1, \ldots, n\}$ and $\gamma \in \{1, \ldots, t+1\}$, its information corresponding to $({}^A\langle\mathbb{A}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\mathbb{B}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\mathbb{C}^{(\alpha,\beta,\gamma)}\rangle_d)$ and $({}^A\langle\hat{\mathbb{A}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\hat{\mathbb{B}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\hat{\mathbb{C}}^{(\alpha,\beta,\gamma)}\rangle_d)$. Here each value $\mathbb{A}^{(\alpha,\beta,\gamma)}, \mathbb{B}^{(\alpha,\beta,\gamma)}, \mathbb{C}^{(\alpha,\beta,\gamma)}, \hat{\mathbb{A}}^{(\alpha,\beta,\gamma)}, \hat{\mathbb{B}}^{(\alpha,\beta,\gamma)}, \hat{\mathbb{C}}^{(\alpha,\beta,\gamma)} \in \mathbb{F}^{n^3p^2}$ and component-wise $\mathbb{C}^{(\alpha,\beta,\gamma)} = \mathbb{A}^{(\alpha,\beta,\gamma)} \times \mathbb{B}^{(\alpha,\beta,\gamma)}$. % Recall that $\mathcal{F}_{\mathsf{Off}}$ generates $c_M n(t+1)$ number of ${}^A\langle\cdot\rangle_d$-sharing of n^3p^2 random multiplication-triples and random triples each.
- For $\alpha \in \{1, \ldots, c_M\}, \beta \in \{1, \ldots, n\}$, and $\gamma \in \{1, \ldots, t+1\}$, its information corresponding to $({}^A\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$ and the scarificing randomness $({}^A\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^A\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$. Here each $\mathbb{R}^{(\alpha,\beta,\gamma)}, \hat{\mathbb{R}}^{(\alpha,\beta,\gamma)} \in$

$\mathbb{F}^{n^3p^2}$. % Recall that $\mathcal{F}_{\mathsf{Off}}$ generates $2c_M n(t+1)$ number of ${}^{A}\langle\cdot\rangle_d$-sharing and ${}^{A}\langle\cdot\rangle_{d+p-1}$-sharing of n^3p^2 random values.

For $\alpha \in \{1,\ldots,c_M\}$, the parties associate the secret-sharings $({}^{A}\langle\mathbb{A}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\alpha,\beta,\gamma)}\rangle_d)$, $({}^{A}\langle\hat{\mathbb{A}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}^{(\alpha,\beta,\gamma)}\rangle_d)$ and the secret-sharings $({}^{A}\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$, $({}^{A}\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$ with the αth multiplication gate in `ckt`, where $\beta \in \{1,\ldots,n\}$ and $\gamma \in \{1,\ldots,t+1\}$. % Each multiplication gate will be evaluated over $n(t+1)$ number of ${}^{A}\langle\cdot\rangle_d$-shared vectors of inputs of size n^3p^2 using instances of Π_{Beaver}.

The parties then evaluate each gate in `ckt` in topological order as follows.

- **Input Gate**: Every $P_i \in \mathcal{P}$ on having $n^4(t+1)p^2$ different values for input-wire x_i of `ckt` does the following.
  - Divide the inputs into $n(t+1)$ groups, each consisting of n^3p^2 values. For $\beta \in \{1,\ldots,n\}$, $\gamma \in \{1,\ldots,t+1\}$, let these groups be denoted by $\mathbb{X}_i^{(\beta,\gamma)}$, where $\mathbb{X}^{(\beta,\gamma)} = (\mathbf{X}^{(\beta,\gamma,1)},\ldots,\mathbf{X}^{(\beta,\gamma,n)})$. Moreover, each $\mathbf{X}^{(\beta,\gamma,u)} = (\boldsymbol{x}^{(\beta,\gamma,u,1)},\ldots,\boldsymbol{x}^{(\beta,\gamma,u,n^2p)})$. Further, each $\boldsymbol{x}^{(\beta,\gamma,u,v)} \in \mathbb{F}^p$.
  - Generate ${}^{A}\langle\mathbb{X}_i^{(\beta,\gamma)}\rangle_d$ by acting as a dealer with input $\mathbb{X}_i^{(\beta,\gamma)}$ and invoking an instance of protocol $\Pi_{A\langle\cdot\rangle}$ (Protocol 5.1).

  $\forall P_i$, the parties associate $\{{}^{A}\langle\mathbb{X}_i^{(\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ with wire x_i.
- **Addition Gate**: For every addition gate in `ckt` with input wires x, y and output wire z, the parties in $\mathcal{P}$ do the following.
  - For $\beta \in \{1,\ldots,n\}$, $\gamma \in \{1,\ldots,t+1\}$, on having the inputs ${}^{A}\langle\mathbb{X}^{(\beta,\gamma)}\rangle_d$ and ${}^{A}\langle\mathbb{Y}^{(\beta,\gamma)}\rangle_d$ associated with wires x and y resp., locally compute ${}^{A}\langle\mathbb{Z}^{(\beta,\gamma)}\rangle_d = {}^{A}\langle\mathbb{X}^{(\beta,\gamma)}\rangle_d + {}^{A}\langle\mathbb{Y}^{(\beta,\gamma)}\rangle_d$.
  - Associate the sharings $\{{}^{A}\langle\mathbb{Z}^{(\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ with wire z.
- **Multiplication Gate**: For αth multiplication gate in `ckt` with input wires $x^{(\alpha)}, y^{(\alpha)}$ and output wire $z^{(\alpha)}$ where $\alpha \in \{1,\ldots,c_M\}$, parties with inputs $\{{}^{A}\langle\mathbb{X}^{(\alpha,\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ and $\{{}^{A}\langle\mathbb{Y}^{(\alpha,\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ associated with wires $x^{(\alpha)}$ and $y^{(\alpha)}$ respectively where $\mathbb{X}^{(\alpha,\beta,\gamma)}, \mathbb{Y}^{(\alpha,\beta,\gamma)} \in \mathbb{F}^{n^3p^2}$, do the following.
  - For each $k \in \{1,\ldots,n\}$, run an instance of Π_{Beaver} (Protocol 6.1) with inputs $\{{}^{A}\langle\mathbb{X}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{Y}^{(\alpha,\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$, and $({}^{A}\langle\mathbb{A}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{B}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{C}^{(\alpha,\beta,\gamma)}\rangle_d)$, $({}^{A}\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\mathbb{R}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$ and auxiliary inputs $({}^{A}\langle\hat{\mathbb{A}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{B}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{C}}^{(\alpha,\beta,\gamma)}\rangle_d)$ and the pairs $({}^{A}\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_d, {}^{A}\langle\hat{\mathbb{R}}^{(\alpha,\beta,\gamma)}\rangle_{d+p-1})$ associated with the αth multiplication-gate where $\beta \in \{(k-1)n+1,\ldots,kn\}$ and $\gamma \in \{1,\ldots,t+1\}$.[20]
    Let $\{{}^{A}\langle\mathbb{Z}^{(\alpha,\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ be the output from Π_{Beaver}.
  - The parties associate the sharings ${}^{A}\langle\mathbb{Z}^{(\alpha,\beta,\gamma)}\rangle_d$ with the output wite $z^{(\alpha)}$, where $\beta \in \{1,\ldots,n\}$ and $\gamma \in \{1,\ldots,t+1\}$.

[20] A single instance of Π_{Beaver} can compute product of $(t+1)n^3p^2$ inputs. Since we want to compute product of $n(t+1)n^3p^2$ inputs, we need n such instances.

– **Output Gate**: Let o be the output-wire of $\texttt{ckt}$. For $\beta \in \{1, \ldots, n\}$ and $\gamma \in \{1, \ldots, t+1\}$, parties on having the input ${}^{A}\langle \mathbb{O}^{(\beta,\gamma)} \rangle_d$ associated with wire o publicly reconstruct $\mathbb{O}^{(\beta,\gamma)}$, output $\mathbb{O}^{(\beta,\gamma)}$ and terminate.

Lemma G.3 (Complexity). *Protocol* Π_{MPC} *requires* $\mathcal{O}(D_M)$ *expected number of rounds and incurs a communication of* $\mathsf{P2P}(c_M n^7 \kappa + n^8 \kappa)$ *bits plus expected* $\mathsf{P2P}(n^3 \log^2 n)$ *bits. Here* c_M *is the number of multiplication gates and* D_M *is the multiplicative depth of* $\texttt{ckt}$.

Proof. During the input sharing phase, each party $P_i \in \mathcal{P}$ invokes $\Pi_{{}^{A}\langle\cdot\rangle}$ $n(t+1)$ times. From Lemma D.7, after securely realizing the broadcasts over point-to-point channels, this requires a constant expected number of rounds and a communication of $\mathsf{P2P}(n^8\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, and at most 15 calls to $\mathcal{F}_{\mathsf{Rand}}$.

Addition gates are computed locally. For each multiplication gate $\alpha \in \{1, \ldots, c_M\}$, parties invoke n instances of Π_{Beaver}. From Lemma F.27, this requires a constant expected number of rounds, communication of $\mathsf{P2P}(n^7\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, and at most 31 calls to $\mathcal{F}_{\mathsf{Rand}}$. The multiplication gates at the same multiplicative depth can be evaluated in parallel. The total communication cost for evaluating all the multiplication gates will be $\mathsf{P2P}(c_M n^7 \kappa)$ bits. On the other hand, the expected number of rounds required for evaluating all the multiplication gates will be $\mathcal{O}(D_M)$. The maximum number of calls to $\mathcal{F}_{\mathsf{Rand}}$ for evaluating all the multiplication gates could be $31c_M$. The calls to $\mathcal{F}_{\mathsf{Rand}}$ can be securely realized using a constant expected round protocol, which will incur a total communication of $\mathsf{P2P}(c_M n^2 \kappa + n^4 \kappa)$ bits Finally, the output gate values are publicly reconstructed which incurs a communication of $\mathsf{P2P}(n^8\kappa)$ bits. □

Theorem G.4. *Protocol* Π_{MPC} *securely realizes the functionality* $\mathcal{F}_{\mathsf{MPC}}$ *with statistical security in the* $\mathcal{F}_{\mathsf{Off}}$*-hybrid model, in the presence of a malicious adversary, corrupting at most* $t < \frac{n}{3}$ *parties. The protocol requires* $\mathcal{O}(D_M)$ *expected number of rounds and incurs a communication of* $\mathsf{P2P}(c_M n^7 \kappa + n^8 \kappa)$ *bits. The communication overhead of the protocol is a constant.*

Proof. We have already analyzed the complexity of Protocol Π_{MPC} in lemma G.3. We present a simulator $\mathcal{S}_{\mathsf{MPC}}$ for the protocol Π_{MPC}.

Simulator G.5: $\mathcal{S}_{\mathsf{MPC}}$: Simulator for Protocol Π_{MPC}

$\mathcal{S}_{\mathsf{MPC}}$ constructs virtual real-world honest parties and invokes the real-world adversary $\mathcal{A}$ who controls the set of parties Cr. The simulator simulates the view of $\mathcal{A}$, namely its communication with $\mathcal{Z}$, the messages sent by the honest parties and the interaction with $\mathcal{F}_{\mathsf{Off}}$. In order to simulate $\mathcal{Z}$, the simulator forwards every message it receives from $\mathcal{Z}$ to $\mathcal{A}$ and vice-versa. The simulator then simulates the various steps of the protocol as follows.

Interaction with $\mathcal{F}_{\mathsf{Off}}$: The simulator itself playing the role of $\mathcal{F}_{\mathsf{Off}}$. Namely, the simulator receives from $\mathcal{A}$ all the MAC-key components and their twisted

t-sharings that the parties in Cr would like to have. Similarly, the simulator receives from $\mathcal{A}$ the shares of the parties in Cr for all multiplication-triples and double-sharings. The simulator then executes the steps of $\mathcal{F}_{\mathsf{Off}}$ and generates key-consistent secret-sharings of random multiplication-triples and random double-sharings, which are consistent with the shares corresponding to the parties in Cr, as provided by $\mathcal{A}$. At the end of simulation of this step, the simulator will know the entire vectors of MAC-key components, their twisted t-sharings, all secret-shared multiplication-triples and all secret-shared random double-sharings.

Input Gates: The simulator simulates the evaluation of input gates as follows.

- For every $P_i \notin \mathsf{Cr}$, the simulator picks $n^4(t+1)p^2$ random inputs from $\mathbb{F}$, groups them as $\{\widetilde{\mathbb{X}}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ where each $\widetilde{\mathbb{X}}_i^{(\beta,\gamma)} \in \mathbb{F}^{n^3p^2}$. For every $\mathbb{X}_i^{(\beta,\gamma)}$, the simulator then run an instance of $\Pi_{A\langle\cdot,\rangle}$ on behalf of P_i by acting as a dealer and interacts with $\mathcal{A}$ to generate ${}^{A}\langle\widetilde{\mathbb{X}}_i^{(\beta,\gamma)}\rangle_d$.
- For every $P_i \in \mathsf{Cr}$, the simulator $\mathcal{S}_{\mathsf{MPC}}$ plays the role of the honest parties as per the steps of $\Pi_{A\langle\cdot,\rangle}$ and interacts with $\mathcal{A}$ on behalf of the honest parties in the instances of $\Pi_{A\langle\cdot,\rangle}$ where P_i plays the role of the dealer. Let $\{\widetilde{\mathbb{X}}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ be the inputs whose ${}^{A}\langle\cdot\rangle_d$-sharings are generated during these instances where each $\widetilde{\mathbb{X}}_i^{(\beta,\gamma)} \in \mathbb{F}^{n^3p^2}$. Note that $\mathcal{S}_{\mathsf{MPC}}$ will know the complete ${}^{A}\langle\widetilde{\mathbb{X}}_i^{(\beta,\gamma)}\rangle_d$.

Interaction with $\mathcal{F}_{\mathsf{MPC}}$: Once $\mathcal{S}_{\mathsf{MPC}}$ learns $\{\widetilde{\mathbb{X}}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ for every $P_i \in \mathsf{Cr}$, it sends to $\mathcal{F}_{\mathsf{MPC}}$ these inputs on behalf of the parties $P_i \in \mathsf{Cr}$ and receives the outputs $\{\mathbb{O}^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$, where each $\mathbb{O} \in \mathbb{F}^{n^3p^2}$.

Circuit-Evaluation: The simulator simulates the evaluation of each gate in `ckt` in topological order as follows.

- **Addition Gate**: Since this step involves local computation, the simulator does not have to simulate any messages on the behalf of the honest parties. The simulator locally adds ${}^{A}\langle\cdot\rangle_d$-sharings corresponding to the gate-input and obtain the ${}^{A}\langle\cdot\rangle_d$-sharings corresponding to the gate-outputs.
- **Multiplication Gate**: The simulator takes the multiplication-triples and the double-pairs associated with the multiplication-gate (generated as part of interaction with $\mathcal{F}_{\mathsf{Off}}$), along with the ${}^{A}\langle\cdot\rangle_d$-sharings corresponding to the gate-inputs and computes the messages of the honest parties as per the steps of the protocol Π_{Beaver}. The simulator then sends these messages to $\mathcal{A}$. Once the simulation of the gate-evaluation is done, the simulator will know the complete ${}^{A}\langle\cdot\rangle_d$-sharings corresponding to the gate-outputs.

Output Gate: Let $\{{}^{A}\langle\widetilde{\mathbb{O}}^{(\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ be the sharings, corresponding to the output gate, available with $\mathcal{S}_{\mathsf{MPC}}$. The simulator computes the sharings $\{{}^{A}\langle\mathbb{O}^{(\beta,\gamma)}\rangle_d\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$, such that the shares, MAC-keys and MAC-tags of the parties in Cr for ${}^{A}\langle\mathbb{O}^{(\beta,\gamma)}\rangle_d$ are identical as in ${}^{A}\langle\widetilde{\mathbb{O}}^{(\beta,\gamma)}\rangle_d$. On behalf of each $P_i \notin \mathsf{Cr}$, the simulator sends to $\mathcal{A}$ the shares of ${}^{A}\langle\widetilde{\mathbb{O}}^{(\beta,\gamma)}\rangle_d$ held

by P_i and the MAC-tags on these shares corresponding to each $P_j \in \mathsf{Cr}$.

Claim G.6. *The view of $\mathcal{A}$ in the simulated execution with $\mathcal{S}_{\mathsf{MPC}}$ is identically distributed as the view of $\mathcal{A}$ in the real execution of Π_{MPC}.*

Proof. It is easy to see that the view of $\mathcal{A}$ during the preprocessing phase is identically distributed in both executions. This is because in both executions, $\mathcal{A}$ receives no messages from the honest parties and the steps of $\mathcal{F}_{\mathsf{Off}}$ are executed by the simulator itself in the simulated execution. Namely, in both executions, $\mathcal{A}$'s view consists of information of the parties in Cr for key-consistent secret-shared random multiplication-triples and random double-pairs. Along with this, adversary's view also consists of the MAC-key components of the parties in Cr and the twisted-shares of the MAC-key components of the honest parties. Let us fix all this data. Conditioned on this data, for the input gate, $\mathcal{A}$ learns its supposed data corresponding to the sharings $\{{}^{A}\langle \mathbb{X}_i^{(\beta,\gamma)} \rangle_d\}_{P_i \notin \mathsf{Cr}}$ during the real execution, while in the simulated execution it learns its supposed data corresponding to the sharings $\{{}^{A}\langle \widetilde{\mathbb{X}}_i^{(\beta,\gamma)} \rangle_d\}_{P_i \notin \mathsf{Cr}}$, where the elements of $\widetilde{\mathbb{X}}^{(\beta,\gamma)}$ are randomly chosen by the simulator. It is easy to see that the distributions of the data held by $\mathcal{A}$ in both the real and simulated executions are identical, as simulator honestly plays the role of honest parties in the underlying $\Pi_{{}^{A}\langle\cdot\rangle}$ instances involving honest dealers. So let us fix the data held by $\mathcal{A}$ for input gates.

During the evaluation of addition gates, no communication is involved. During the evaluation of multiplication gates, in the simulated execution, the simulator will know the complete ${}^{A}\langle\cdot\rangle_d$-sharings associated with gate-inputs. It will also known the complete ${}^{A}\langle\cdot\rangle_d$-sharings of the associated multiplication-triples and also $({}^{A}\langle\cdot\rangle_d, {}^{A}\langle\cdot\rangle_{d+p-1})$-sharings of the associated double-pairs. Hence the simulator can correctly simulate the messages sent by the honest parties during the execution of the corresponding instance of Π_{Beaver}. This ensures that the $\mathcal{A}$'s view during the evaluation of multiplications gates is identically distributed, both in the real execution and the simulated execution.

For the output gate, the data received by $\mathcal{A}$ in the real execution from the honest parties correspond to ${}^{A}\langle\cdot\rangle_d$-sharings of the circuit-outputs $\mathbb{O}^{(\beta,\gamma)}$. From the steps of $\mathcal{S}_{\mathsf{MPC}}$ (as described in the informal discussion), it is easy to see that the same holds in the simulated execution, since the simulator computes ${}^{A}\langle\cdot\rangle_d$-sharings of $\mathbb{O}^{(\beta,\gamma)}$, where the data to be held by the parties in Cr is identical to the simulated execution. Hence $\mathcal{A}$'s view is identically distributed in both executions during the evaluation of the output gate as well. □

Claim G.7. *Conditioned on the view of $\mathcal{A}$, the output of the honest parties is identically distributed in the real execution of Π_{MPC} involving $\mathcal{A}$, as well as in the ideal execution involving $\mathcal{S}_{\mathsf{MPC}}$ and $\mathcal{F}_{\mathsf{MPC}}$.*

Proof. Let view be an arbitrary view of $\mathcal{A}$. According to view, for every $P_i \in \mathcal{P}$, there exist inputs $\{\mathbb{X}_i^{(\beta,\gamma)}\}_{\beta\in\{1,\ldots,n\},\gamma\in\{1,\ldots,t+1\}}$ where $\mathbb{X}_i^{(\beta,\gamma)} \in \mathbb{F}^{n^3p^2}$, such that the parties hold ${}^{A}\langle \mathbb{X}_i^{(\beta,\gamma)} \rangle_d$. Furthermore, from Claim G.6, if $P_i \in \mathsf{Cr}$ then the

corresponding ${}^{A}\langle \mathbb{X}_i^{(\beta,\gamma)} \rangle_d$ is included in view while for $P_i \notin \mathsf{Cr}$, the corresponding $\mathbb{X}_i^{(\beta,\gamma)}$ are uniformly distributed, conditioned on the data for ${}^{A}\langle \mathbb{X}_i^{(\beta,\gamma)} \rangle_d$ available with $\mathcal{A}$ as determined by view. Let us fix the $\mathbb{X}_i^{(\beta,\gamma)}$ values corresponding to the parties in $\mathcal{P}$ and consider the vector of values $(\mathbb{X}_1^{(\beta,\gamma)}, \ldots, \mathbb{X}_n^{(\beta,\gamma)})$, where $\beta \in \{1, \ldots, n\}$ and $\gamma \in \{1, \ldots, t+1\}$.

It is easy to see that in the ideal-world, the output of the honest parties is $\{\mathbb{O}^{(\beta,\gamma)}\}_{\beta \in \{1,\ldots,n\}, \gamma \in \{1,\ldots,t+1\}}$, where $\mathbb{O}^{(\beta,\gamma)}$ is the output of the circuit-evaluation over input $(\mathbb{X}_1^{(\beta,\gamma)}, \ldots, \mathbb{X}_n^{(\beta,\gamma)})$. We now show that the honest parties output $\mathbb{O}^{(\beta,\gamma)}$ even in the real-world, except with a negligible probability. For this, we argue that all the values during the circuit-evaluation are correctly ${}^{A}\langle \cdot \rangle_d$-shared, except with a negligible probability.

Since the evaluation of addition gates needs only local computation, it follows that the output of addition gates will be correctly ${}^{A}\langle \cdot \rangle_d$-shared. Since multiplication-gates are evaluated using instances of Π_{Beaver}, it follows from the correctness of Π_{Beaver} (Lemma F.25) that the output of multiplication-gates will be correctly ${}^{A}\langle \cdot \rangle_d$-shared, except with a negligible probability. Since all the gates are evaluated correctly except with a negligible probability, it follows that for the output-gates, the parties will correctly hold ${}^{A}\langle \mathbb{O}^{(\beta,\gamma)} \rangle_d$ which the honest parties reconstruct correctly, except with a negligible probability. □

The constant communication overhead is computed as follows: *each* multiplication gate is evaluated over $n(t+1)n^3p^2 = \Theta(n^7)$ field elements, represented by $\Theta(n^7\kappa)$ bits. The total cost for evaluating the multiplication gates is $\mathsf{P2P}(c_M n^7 \kappa)$ bits and so the communication overhead per multiplication gate is a constant.

Each party has $n(t+1)n^3p^2 = \Theta(n^7)$ field elements as inputs, hence there are *total* $\Theta(n^8)$ field elements as inputs for $\mathtt{ckt}$, represented by $\Theta(n^8\kappa)$ bits. Since the *total* cost of secret-sharing of all inputs is $\mathsf{P2P}(n^8\kappa)$ bits plus expected $\mathsf{P2P}(n^3 \log^2 n)$ bits, the communication overhead per input gate is a constant. Finally, there are $n(t+1)n^3p^2 = \Theta(n^7)$ field elements as outputs, which should be learned by *all* n parties. So, $\mathtt{ckt}$ consists of $\Theta(n^8)$ field elements as output, represented by $\Theta(n^8\kappa)$ bits. Since the total cost of reconstructing all output values is $\mathsf{P2P}(n^8\kappa)$ bits, the communication cost per output gate is a constant This completes the proof of Theorem G.4. □

From Theorem F.43, we can securely realize the functionality $\mathcal{F}_{\mathsf{Off}}$ using protocol Π_{Off} over point-to-point channels. Consequently, we get the following theorem.

Theorem 7.1. *Let $n = 3t+1$ and let $f : \mathbb{F}^n \to \mathbb{F}$ be a function, consisting of c_M multiplication gates and has multiplicative depth D, where f has a single input from each P_i and a single output for all n parties. Then there exists a statistically-secure MPC for computing f in an SIMD fashion $\Theta(n^7)$ times. The protocol requires $\mathcal{O}(D)$ expected rounds and incurs a communication of $\mathcal{O}(c_M n^7 + n^8)$ field elements, implying an overhead of $\mathcal{O}(1)$ field elements per gate.*