Round and Communication Efficient Unconditionally-secure MPC with $t < n/3$ in Partially Synchronous Network

Abstract

In this work, we study unconditionally-secure multi-party computation (MPC) tolerating $t < n/3$ corruptions, where n is the total number of parties involved. In this setting, it is well known that if the underlying network is *completely asynchronous*, then one can achieve only *statistical* security; moreover it is *impossible* to ensure *input provision* and consider inputs of all the honest parties. The best known statistically-secure asynchronous MPC (AMPC) with $t < n/3$ requires a communication of $\Omega(n^5)$ field elements per multiplication. We consider a *partially synchronous* setting, where the parties are assumed to be globally synchronized initially for few rounds and then the network becomes completely asynchronous. In such a setting, we present a MPC protocol, which requires $\mathcal{O}(n^2)$ communication per multiplication while ensuring input provision. Our MPC protocol relies on a new four round, communication efficient statistical *verifiable secret-sharing* (VSS) protocol with broadcast communication complexity *independent* of the number of secret-shared values.

1 Introduction

Threshold unconditionally-secure multiparty computation (MPC) is a fundamental problem in secure distributed computing [40, 26, 8, 12, 38, 2]. Informally, an MPC protocol enables a set of n mutually distrusting parties to jointly and securely compute a publicly known function f of their private inputs over some finite field $\mathbb{F}$, even in the presence of a *computationally unbounded active adversary* Adv, who can corrupt any t out of the n parties. Let the parties be connected by pair-wise secure (private and authentic) channels. Then in the *synchronous* communication setting, where the parties are assumed to be synchronized through a global clock, it is known that *perfectly-secure* MPC is possible if and only if $t < n/3$ [8]. If a common broadcast channel is also available to the parties in addition to the pair-wise secure channels, then one can tolerate upto $t < n/2$ corruptions, albeit with *statistical security*[1] [38]. The resilience bounds become different if one considers a completely *asynchronous* setting, where parties are not synchronized and messages can be arbitrarily delayed. Specifically, perfectly-secure asynchronous MPC (AMPC) is possible if and only if $t < n/4$ [7], while statistically-secure AMPC is possible if and only if $t < n/3$ [9].

Feasibility Results for Unconditionally-secure MPC: In any general MPC protocol [8, 12, 38, 2, 27, 3, 20, 4, 5, 10, 15], the function f is usually expressed as an arithmetic circuit (consisting of addition and multiplication gates) over $\mathbb{F}$ and then the protocol "securely" evaluates each gate in the circuit in a shared/distributed fashion. More specifically, each party secret-shares its inputs among the parties using a linear secret-sharing scheme (LSS) [17], say Shamir [39], with threshold[2] t. The parties then interact to maintain the following invariant for each gate: *given the gate inputs in a secret-shared fashion, the gate output is computed in a secret-shared fashion*. Finally the (shared) circuit output is publicly reconstructed. Intuitively, the privacy follows

[1] The outcome of a perfectly-secure protocol is error-free, while a negligible error is allowed in a statistically-secure protocol.

[2] Informally such a scheme ensures that the shared value remains information-theoretically secure even if upto t shares are revealed. Shamir sharing of a secret with threshold t is done by selecting a random polynomial of degree at most t with the secret as the constant term and defining the individual shares as distinct evaluations of the polynomial.

since each intermediate value in the above process remains secret-shared with threshold t. Due to the *linearity* of the LSS, the addition (linear) gates are evaluated *locally* by the parties. However, maintaining the above invariant for the multiplication (non-linear) gates requires interaction among the parties. The focus therefore is rightfully placed on measuring the communication complexity (namely the *total* number of field elements communicated) required to evaluate the multiplication gates in the circuit. In the recent past, a lot of work has been done to design communication-efficient MPC protocols; we summarize the relevant works here.

With $t < n/3$, [5] presents a perfectly-secure MPC protocol with $\mathcal{O}(n)$ amortized communication complexity[3] per multiplication, while [10] presents a statistically-secure MPC protocol with $t < n/2$ with almost $\mathcal{O}(n)$ communication complexity per multiplication. Both these results are in the synchronous setting and require *non-constant* number of rounds of interaction among the parties. While the protocol of [5] requires $\Theta(n + \mathcal{D})$ rounds, the protocol of [10] requires $\Theta(n^2 + \mathcal{D})$ rounds, where $\mathcal{D}$ denotes the *multiplicative depth* of the circuit.

A major drawback of the synchronous setting is that it does not model real life networks like the Internet accurately where it is very hard to ensure that the users are synchronized through a global clock and that there exists a strict a priori known upper bound on the message delivery. Real life networks can be modelled more appropriately by the asynchronous setting, where there are no known upper bounds and messages are delivered arbitrarily (the only guarantee given in this model is that the messages sent by the honest parties will reach to their desitination eventually). Hence designing AMPC protocols is practically motivated. However, an inherent challenge in designing protocols in a completely asynchronous setting is that it is impossible to distinguish between a *slow*, but honest party (whose messages are delayed arbitrarily) and a corrupt party (who do not send any message at all). Hence in a completely asynchronous protocol, no party can afford to receive messages from all the n parties, as the wait may turn out to be an endless wait. So as soon a party receives messages from $n - t$ parties, it has to proceed to the next "step" of the protocol. However, in this process, messages from t potentially honest, but slow parties may get ignored. Due to this inherent phenomena, designing efficient AMPC protocols is a challenging task, as evident from the known feasibility results for AMPC protocols summarized below.

In a completely asynchronous setting, [36] presents a perfectly-secure AMPC protocol with $t < n/4$ and $\mathcal{O}(n^2)$ communication per multiplication, while [34] presents a statistically-secure AMPC with $t < n/3$ and $\mathcal{O}(n^5)$ communication per multiplication. As it is clear, there is a significant gap in the communication complexity of MPC and AMPC protocols. In addition, any AMPC protocol cannot ensure *input provision*, namely the inputs of *all* the honest parties may not be considered for the circuit evaluation, as this may turn out to be an endless wait and so inputs of upto t potentially honest parties may get ignored. With an aim to bridge the gap in the communication complexity of synchronous and asynchronous MPC and to enforce input provision, the works of [4, 14] motivate and consider *hybrid* asynchronous setting, where the network is assumed to be synchronized for few initial rounds and then it becomes completely asynchronous. This is a practically motivated communication setting, which has been well considered in the recent past for bridging the efficiency gap between synchronous and asynchronous protocols for various distributed computing tasks [30, 4, 23, 6, 14].

With $t < n/4$, a perfectly-secure hybrid MPC protocol with one synchronous round is presented in [14], with $\mathcal{O}(n)$ amortized communication complexity per multiplication. In [15], four MPC protocols in the hybrid setting are proposed with $t < n/3$; while two of these protocols are perfectly-secure, the remaining two are statistically-secure. These protocols are obtained by instantiating the efficient framework for unconditionally-secure MPC proposed in [15] with existing VSS schemes with $t < n/3$ (more on this later). Among the perfectly-secure protocols, the first one requires less number of synchronous rounds, namely[4] $(12, 3)$, but requires a higher communication of $\mathcal{O}(n^5)$ per multiplication. The second perfectly-secure protocol requires more number of synchronous rounds, namely $(21, 7)$, but provides a better communication complexity of $\mathcal{O}(n^4)$ per multiplication. So a tradeoff is attained between the amount of synchrony required and communication achieved per multiplication. The statistically-secure hybrid protocols of [15] with $t < n/3$ retain the same

[3]The amortized communication complexity is derived under the assumption that the circuit is large enough so that the terms that are independent of the circuit size can be ignored.

[4]We say a protocol requires (r, r') (synchronous) rounds, if it requires a total of r rounds of interaction among the parties and out of these r rounds, r' rounds require broadcast by the parties, where $r' \leq r$.

communication complexity as their perfect counterparts, but reduces the number of synchronous rounds. Namely the first statistically-secure protocol requires $(7, 2)$ rounds and $\mathcal{O}(n^5)$ communication per multiplication, the second statistically-secure protocol requires $(16, 6)$ rounds and $\mathcal{O}(n^4)$ communication per multiplication. As it is clear from these results, with $t < n/3$, significant improvement in the communication complexity is not achieved, even if partial synchrony is provided in the network. Our goal is to design more efficient hybrid MPC protocol with $t < n/3$ using minimal level of synchrony.

Our Results. We present a hybrid MPC protocol with $t < n/3$. Our protocol is *statistically-secure*, requires $(4, 3)$ synchronous rounds and $\mathcal{O}(n^2)$ communication per multiplication. Moreover, our protocol also ensures input provision. Our protocol outperforms the existing hybrid MPC protocols with $t < n/3$, both in terms of communication complexity as well as in terms of the number of synchronous rounds required in the protocol.

To design our protocol, we follow the standard offline-online paradigm, based on Beaver's circuit-randomization technique [2] and which is now the de facto style of designing efficient MPC protocols [3, 4, 5, 10, 14, 15]. In this paradigm, an MPC protocol is divided into two phases, a *circuit-independent* offline phase and a *circuit-dependent* online phase. While the offline phase generates "raw data", independent of the circuit and actual inputs for the computation, the online phase utilizes this raw data for the circuit evaluation. In a more detail, the offline phase generates random *multiplication triples* of the form (a, b, c), Shamir-shared with threshold t; here a, b are random and private and $c = ab$ holds. Later, using such triples, multiplication gates are evaluated in a shared fashion. For each multiplication gate, one multiplication triple from the offline phase is utilized and the multiplication gate is evaluated at the cost of publicly reconstructing two Shamir-shared values. Reconstructing a Shamir-shared valued (with threshold t) can be done efficiently with $t < n/3$ using the standard Reed-Solomon (RS) error correction [31], even in a *completely asynchronous* setting [11, 7]. Hence we shift the focus to design an efficient offline phase in the hybrid setting for generating multiplication triples. For this we follow the recent framework of [15], which shows how to efficiently generate Shamir-shared multiplication triples in offline phase, using *any* (polynomial based) *verifiable secret-sharing* (VSS) protocol [13] as a black-box. Informally, a VSS protocol allows a designated party called *dealer* (D) to *verifiably* Shamir-share a secret with threshold t. Thus at the end of the VSS protocol it is ensured that there exists some polynomial of degree at most t with the secret as the constant term and every share-holder has a distinct evaluation of this polynomial. Moreover this is ensured irrespective of whether the dealer is under the influence of the adversary or not. In addition, if the dealer is *honest* then it is ensured that the secret remains information-theoretically secure from t corrupted share-holders.

In this work, our proposed VSS protocol in the setting of $t < n/3$ is plugged into the framework of [15] and the result is a more efficient hybrid MPC protocol. Communication-wise, our VSS protocol stands out with an amortized overhead of $\mathcal{O}(n^2)$ per secret-shared value, whereas the best known bound is only $\mathcal{O}(n^3)$ [25, 28]. The improvement comes from the fact that our VSS protocol requires a broadcast complexity that is *independent* of the number of secrets shared, a property that is not achieved by the known constructions [25, 28]. To induce a better complexity over point-to-point channels, we use the best known *broadcast amplification* protocols (aka multi-valued broadcast protocols) [22] to simulate the broadcast invocations in the VSS protocols of [25, 28]. Informally, in a multi-valued protocol, broadcasting a "sufficiently large" message of size ℓ has communication complexity of $\mathcal{O}(n\ell)$ over point-to-point channels and a broadcast complexity of poly(n). With $t < n/3$, the most efficient multi-valued broadcast protocol is due to [37]. The protocol requires a communication complexity of $\mathcal{O}(n\ell)$ over point-to-point channels and broadcast of n^2 bits for broadcasting an ℓ-bit message. Detailed analysis and comparison of our VSS with existing ones is deferred later in the paper (see section 4.3). In Table 1, we compare our MPC and VSS protocols with their previous best counter parts.

Other Related Work. In the *synchronous* setting, MPC protocols with $\mathcal{O}(n)$ communication per multiplication has been reported in [5] with perfect security and $t < n/3$ and in [10] with statistical security and $t < n/2$. These protocols deploy *non-robust* secret-sharing protocols in the player-elimination and dispute-control fra-

Figure 1

(a) Communication complexity (in bits) per multiplication of various AMPC and hybrid MPC protocols with $t < n/3$

Security	Underlying Network		VSS Deployed in the Offline Phase	Communication Complexity
	Offline Phase	Online Phase		
Statistical	Asynchronous	Asynchronous	[34]	$\mathcal{O}(n^5 \log \lvert\mathbb{F}\rvert)$ [34]
Perfect	Hybrid $(12, 3)$ rounds	Asynchronous	[28]	$\mathcal{O}(n^5 \log \lvert\mathbb{F}\rvert)$ [15]
Perfect	Hybrid $(21, 7)$ rounds	Asynchronous	[25]	$\mathcal{O}(n^4 \log \lvert\mathbb{F}\rvert)$ [15]
Statistical	Hybrid $(7, 2)$ rounds	Asynchronous	[28]	$\mathcal{O}(n^5 \log \lvert\mathbb{F}\rvert)$ [15]
Statistical	Hybrid $(16, 6)$ rounds	Asynchronous	[25]	$\mathcal{O}(n^4 \log \lvert\mathbb{F}\rvert)$ [15]
Statistical	Hybrid $(4, 3)$ rounds	Asynchronous	**[This work]**	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$ **[This work]**

(b) Amortized communication complexity per shared-secret of the underlying VSS deployed in the offline phase.

Security	Network Type	Overhead
Statistical	Asynchronous	$\mathcal{O}(n^4)$[34]
Perfect	Synchronous $(7, 2)$ rounds	$\mathcal{O}(n^4)$ [28]
Perfect	Synchronous $(16, 6)$ rounds	$\mathcal{O}(n^3)$ [25]
Statistical	Synchronous $(4, 3)$ rounds	$\mathcal{O}(n^2)$ **[This work]**

mework. The non-robustness of the underlying primitives inflates the round complexity of their offline phase to $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ respectively. The naive approach of adopting these protocols in hybrid setting will lead to protocols with $\mathcal{O}(n)$ or $\mathcal{O}(n^2)$ synchronous (broadcast) rounds to execute the offline phase. The online phase of these protocols can be executed asynchronously. Our hybrid MPC protocol on the other hand requires only a constant number of synchronous broadcast rounds.

The reported works [19, 18] in the synchronous setting with polylogarithmic (in n) communication per gate (denoted as $\widetilde{\mathcal{O}}(n)$) [5] are only *non-optimally* resilient. While [19] works with $t < (\frac{1}{2} - \epsilon)n$ and provides statistical security, [18] works with $t < (\frac{1}{3} - \epsilon)n$ and provides perfect security, where $\epsilon > 0$. These protocols also evaluate the underlying circuit in a secret-shared fashion. However, instead of Shamir secret-sharing, they use packed secret-sharing [24] taking advantage of the presence of larger subset of honest parties (due to the non-optimal resilience). Due to the use of packed secret-sharing, "multiple" gates can be evaluated simultaneously by doing a fixed set of operations on the shares. However, this requires "special" structure from the underlying circuit being available at each layer, maintaining which, demands additional circuitry to be incorporated between different layers of the circuit. Evaluating the overall circuit using packed secret-sharing makes these protocols highly non-trivial and complex. It is not known how to adapt these protocols in a completely asynchronous or a partially synchronous setting. Specifically, it is not clear whether these protocols can be executed in a hybrid setting, with a *constant* number of synchronous rounds. Therefore, while treating VSS as an MPC functionality and evaluating the resultant "VSS circuit" using the MPC protocols of [19, 18] may lead to sublinear (namely $\widetilde{\mathcal{O}}(n)$) overhead per secret-shared value, it is not clear if the resultant protocols runs with a *constant* number of synchronous rounds in hybrid setting.

New Techniques. Our VSS protocol is built upon a new primitive called *information checking with succinct proof of possession* (ICPoP) that takes motivation from *information checking protocol* (ICP) introduced in [38, 16, 35]. An ICP allows a D to privately authenticate some data for an intermediary INT, who can later publicly reveal this data and prove that it originated from D. On the other hand, in an ICPoP protocol INT gives a proof of possession publicly of the data originated from D, instead of publicly revealing the data. The proof preserves data privacy and is "succinct" i.e. its size is *independent* of the size of the data. The succinctness of the proof makes the broadcast complexity of our VSS protocol independent of the number of shared secrets. Our ICPoP also offers *transferability* that allows any designated party to take possession of INT's authenticated

[5] The actual complexity (communication, computation and round) of these protocols are of the form $\mathcal{O}((\log^k n \cdot \text{poly}(\log |C|)) \cdot |C|) + \mathcal{O}(\text{poly}(n, \log |C|, \mathcal{D}))$, where $\mathcal{D}$ is the multiplicative depth of the underlying circuit C.

(by D) data and to be able to give a proof of possession on the "behalf" of INT. The existing ICPs do not support transferability.

We next give a high level overview of our VSS. To share a secret s, we embed s in the constant term of a random bivariate polynomial $F(x, y)$ of degree t in x and y. Every party P_i then obtains a *row polynomial* $f_i(x) = F(x, \alpha_i)$. The parties then publicly verify whether the row polynomials of at least $n - t$ parties called VCORE define a unique bivariate polynomial. The standard way to do this is to perform the "pair-wise checking", where every pair of parties (P_i, P_j) is asked to verify the consistency of the common values on their respective polynomials and publicly complain if there is any inconsistency, in which case D publicly resolves the complaint by making the common value public [25, 21, 28]. This approach will lead to a broadcast complexity of $\mathcal{O}(n^2)$ per secret-shared value; instead we use a statistical protocol called Poly-Check (section 4.1), adapted from [36], which performs the same task in parallel for ℓ secrets (and hence ℓ bivariate polynomials), but keeping the broadcast complexity independent of ℓ. Once VCORE is found, it is ensured that D has committed a unique $F(x, y)$ and the secret $F(0, 0)$ to the parties in VCORE. To enable the parties to obtain their shares, the goal will be to enable each party P_j to compute its *column polynomial* $g_j(y) = F(\alpha_j, y)$. For this each party $P_i \in$ VCORE transfers its common value on $g_j(y)$ (namely $f_i(\alpha_j)$) to P_j. To ensure that correct values are transferred, P_j publicly gives a proof of possession of all the transferred values originated from D via the intermediary parties in VCORE. This is done in parallel for ℓ secrets (and hence ℓ bivariate polynomials); the succinctness of the proof ensures that this step has broadcast complexity, independent of ℓ.

2 Network Model, Definitions and Existing Tools

We consider a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n parties, connected by pair-wise private and authentic channels. For simplicity we assume $n = 3t + 1$, so $t = \Theta(n)$. There exists a computationally unbounded adversary Adv who can maliciously corrupt any t parties and may force them to behave in any arbitrary fashion during the execution of a protocol. We assume the adversary to be static, who decides the set of corrupted parties at the beginning of the protocol execution. We assume a partially synchronous network, where the first four rounds are synchronous, after which the entire communication is done *asynchronously*. Moreover, we assume that the parties have access to a broadcast channel during the second, third and fourth synchronous round. Our protocols operate over a finite field $\mathbb{F}$, where $|\mathbb{F}| > 2n$. We assume that there exists $2n$ distinct non-zero elements $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ in $\mathbb{F}$. Each element of $\mathbb{F}$ can be represented by $\mathcal{O}(\log |\mathbb{F}|)$ bits. The communication complexity of any protocol is defined to be the total number of field elements communicated by the *honest* parties in that protocol. We denote the point-to-point communication complexity by $\mathcal{PC}()$ and the broadcast communication complexity as $\mathcal{BC}()$.

Without loss of generality, we assume that the parties want to securely compute the function $f : \mathbb{F}^n \rightarrow \mathbb{F}$ via an MPC protocol, where $f(x_1, \ldots, x_n) = y$, such that $x_i \in \mathbb{F}$ is the input of P_i and every party is supposed to receive the output $y \in \mathbb{F}$. The function f is assumed to be represented by a publicly known arithmetic circuit C over $\mathbb{F}$. The circuit C consists of n input gates, two-input addition (linear) and multiplication (non-linear) gates, zero-input random gates (for generating random values during the computation) and one output gate. We denote by c_M and c_R the number of multiplication and random gates in C respectively. By $[X]$ and $[X, Y]$ for $Y \geq X$, we denote the sets $\{1, \ldots, X\}$ and $\{X, X + 1, \ldots, Y\}$, respectively. We use $i \in [k]$ to denote that i can take a value from the set $\{1, 2 \ldots k\}$. We will also require that $|\mathbb{F}| > 4n^4(c_M + c_R)(3t + 1)2^{\kappa}$ to ensure that the error-probability of our MPC protocol is at most $2^{-\kappa}$, for a given error parameter κ.

2.1 Definitions

Definition 2.1 (d**-sharing [3, 20, 5]).** *A value $s \in \mathbb{F}$ is said to be d-shared if there exists a polynomial over $\mathbb{F}$, say $f(\cdot)$, of degree at most d, such that $f(0) = s$ and every (honest) party $P_i \in \mathcal{P}$ holds a share s_i of s, where $s_i = f(\alpha_i)$. We denote by $[s]_d$, the vector of shares of s corresponding to the (honest) parties in $\mathcal{P}$.*

A vector $\vec{S} = (s^{(1)}, \ldots, s^{(\ell)}) \in \mathbb{F}^\ell$ is said to be d-shared if *each* $s^{(i)}$ is d-shared. Note that d-sharings are *linear*: given $[a]_d$ and $[b]_d$, then $[a+b]_d = [a]_d + [b]_d$ and $[c \cdot a]_d = c \cdot [a]_d$ holds, for a public constant c. In general, given ℓ sharings $[x^{(1)}]_d, \ldots, [x^{(\ell)}]_d$ and a public linear function $g : \mathbb{F}^\ell \rightarrow \mathbb{F}^m$, where $g(x^{(1)}, \ldots, x^{(\ell)}) = (y^{(1)}, \ldots, y^{(m)})$, then $g([x^{(1)}]_d, \ldots, [x^{(\ell)}]_d) = ([y^{(1)}]_d, \ldots, [y^{(m)}]_d)$. We say that the parties *locally compute* $([y^{(1)}]_d, \ldots, [y^{(m)}]_d) = g([x^{(1)}]_d, \ldots, [x^{(\ell)}]_d)$ to mean that every P_i (locally) computes $(y_i^{(1)}, \ldots, y_i^{(m)}) = g(x_i^{(1)}, \ldots, x_i^{(\ell)})$, where $y_i^{(l)}$ and $x_i^{(l)}$ denotes the i^{th} share of $y^{(l)}$ and $x^{(l)}$ respectively.

Definition 2.2 ((Polynomial-based) Verifiable Secret Sharing (VSS) [3, 4, 5]). *Let $\vec{S} = (s^{(1)}, \ldots, s^{(L)}) \in \mathbb{F}^L$ be a set of L values that a dealer $\mathsf{D} \in \mathcal{P}$ wants to t-share among $\mathcal{P}$. Let Sh be a protocol for the n parties, where D has the input $\vec{S}$. Then Sh is a VSS scheme if the following holds for every possible Adv, on all possible inputs:* **(1) Correctness***: If D is honest then $\vec{S}$ is t-shared among $\mathcal{P}$ at the end of Sh. Moreover even if D is corrupted there exists a set of L values, say $(\overline{s}^{(1)}, \ldots, \overline{s}^{(L)})$, which is t-shared among $\mathcal{P}$ at the end of Sh.* **(2) Privacy***: If D is honest then Sh reveals no information about $\vec{S}$ to Adv in the information-theoretic sense; i.e. Adv's view is identically distributed for all possible $\vec{S}$.*

If Sh satisfies all its properties without any error then it is called perfectly-secure. If the correctness is satisfied with probability at least $1 - \epsilon$, for a given error parameter ϵ, then it is called statistically-secure.

Unconditionally-secure MPC: Recent papers on efficient unconditionally-secure MPC follow a simpler "property based" security definition of secure MPC [3, 20, 5, 10], instead of the more rigorous "real-world/ideal-world" paradigm based definition [29, 1]. As our main goal is to provide an efficient VSS and MPC, to avoid blurring the main focus of the paper and to avoid additional technicalities, we also use the property based security definition. However, we confirm that using standard techniques, like the above efficient protocols, our MPC protocol can be also proved secure according to the simulation based definition.

Let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a publicly known function and party P_i has input $x_i \in \mathbb{F}$. In any (unconditionally-secure) multiparty computation, each party P_i t-shares its input. Let x_i be the value shared by P_i. If P_i is honest then $\mathsf{x}_i = x_i$. The parties then compute f as $y = f(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ and everyone receives y.

Definition 2.3 (Unconditionally-secure MPC). *A protocol Π among the n parties securely computes f, if it satisfies the following for every possible Adv, on all possible inputs:* **(1) Correctness***: All honest parties obtain y at the end of Π.* **(2) Privacy***: Adv obtains no additional information about the inputs of the honest parties, other than what is inferred from the inputs of the corrupted parties and y.*

Protocol Π is called perfectly-secure if it satisfies all its properties without any error. If the correctness is satisfied with probability at least $1 - 2^{-\kappa}$, for a given error parameter κ, then Π is called statistically-secure.

Information Checking with Succinct Proof of Possession (ICPoP): An ICPoP protocol involves three entities: a designated dealer $\mathsf{D} \in \mathcal{P}$ who holds a set of L private values $\mathcal{S} = \{s^{(1)}, \ldots, s^{(L)}\}$, an intermediary $\mathsf{INT} \in \mathcal{P}$ and the set of parties $\mathcal{P}$ acting as verifiers (note that D and INT will also play the role of verifiers, apart from their designated role of dealer and intermediary respectively). The protocol proceeds in three phases, each of which is implemented by a dedicated sub-protocol: **(1) Distribution Phase**: Here D, sends $\mathcal{S}$ to INT along with some *auxiliary information*. For the purpose of verification, some *verification information* is additionally sent to each individual verifier. **(2) Authentication Phase**: This phase is initiated by INT who interacts with D and the verifiers to ensure that the information it received from D is consistent with the verification information distributed to the individual verifiers. If D wants it can publicly abort this phase, which is interpreted as if D is accusing INT of malicious behaviour. **(3) Revelation Phase**: This phase is carried out by INT and the verifiers in $\mathcal{P}$ only if D has not aborted the previous phase. Here INT reveals a proof of possession of the values received from D. The verifiers in $\mathcal{P}$ check this proof with respect to their verification information. Then based on certain criteria, each verifier either outputs `AcceptProof` (indicating that it accepts the proof) or `RejectProof` (indicating that it rejects the proof).

Definition 2.4 (ICPoP). *A triplet of protocols $(\mathsf{Distr}, \mathsf{AuthVal}, \mathsf{RevealPoP})$ (implementing the distribution, authentication and revelation phase respectively) is a (1 - ϵ)-secure ICPoP, for an error parameter ϵ, if the*

following holds: **(1)** ICPoP-**Correctness1***: If* D *and* INT *are honest, then each honest verifier* $P_i \in \mathcal{P}$ *outputs* `AcceptProof` *at the end of* RevealPoP. **(2)** ICPoP-**Correctness2***: If* D *is corrupted and* INT *is honest and if* ICPoP *proceeds to* RevealPoP*, then except with probability at most* ϵ*, all honest verifiers output* `AcceptProof` *at the end of* RevealPoP. **(3)** ICPoP-**Correctness3***: If* D *is honest,* INT *is corrupted,* ICPoP *proceeds to* RevealPoP *and if the honest verifiers output* `AcceptProof`*, then except with probability at most* ϵ*, the proof produced by* INT *corresponds*[6] *to the values in* $\mathcal{S}$. **(4)** ICPoP-**Privacy***: If* D *and* INT *are honest, then information obtained by* Adv *during* ICPoP *is independent of* $\mathcal{S}$. **(5)** ICPoP-**Succinctness of the Proof***: The size of the proof produced by* INT *during* RevealPoP *is independent of* L.

Properties of Polynomials: A bivariate polynomial $F(x, y)$ of degree at most t is of the form $F(x, y) = \sum_{i,j=0}^{i,j=t} r_{ij} x^i y^j$, where $r_{ij} \in \mathbb{F}$. Let $f_i(x) \stackrel{\text{def}}{=} F(x, \alpha_i), g_i(y) \stackrel{\text{def}}{=} F(\alpha_i, y)$ for $i \in [n]$. We call $f_i(x)$ and $g_i(y)$ as ith *row polynomial* and *column polynomial* respectively of $F(x, y)$. We say that a row polynomial $\overline{f}_i(x)$ *lies* on a bivariate polynomial $F(x, y)$ of degree at most t if $F(x, \alpha_i) = \overline{f}_i(x)$ holds. Similarly we will say that a column polynomial $\overline{g}_i(y)$ *lies* on $F(x, y)$ if $F(\alpha_i, y) = \overline{g}_i(y)$ holds. We will use some well known standard properties of bivariate and univariate polynomials, which are stated in Appendix A.

3 Efficient ICPoP

We present a $(1-\epsilon)$-secure ICPoP protocol, where $|\mathcal{S}| = L = \ell \times \mathsf{pack}$, with $\ell \geq 1$ and $1 \leq \mathsf{pack} \leq n - t$; moreover $\epsilon = \max\{\frac{n\ell}{|\mathbb{F}|-1}, \frac{n(n-1)}{|F|-\mathsf{pack}}\}$. The protocol has communication complexity $\mathcal{PC}(\mathcal{O}(n\ell))$ and $\mathcal{BC}(\mathcal{O}(n))$. Hence the broadcast complexity is *independent* of ℓ. Our ICPoP is similar to the asynchronous ICP of [35], adapted to the synchronous setting with the following differences: in ICP the whole $\mathcal{S}$ is revealed during the revelation phase, as only its authenticity is required during the revelation phase. We require INT to be able to publicly prove the possession of $\mathcal{S}$ while maintaining its privacy. Hence the auxiliary information distributed in our ICPoP differs and also used differently; the details follow.

Let $\mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pack})}), \ldots, (s^{(\ell,1)}, \ldots, s^{(\ell,\mathsf{pack})})\}$. During the distribution phase, D embeds the values $(s^{(k,1)}, \ldots, s^{(k,\mathsf{pack})})$ for $k \in [\ell]$ in a random degree d *secret-encoding* polynomial $G^{(k)}(x)$ at $x = \beta_1, \ldots, \beta_{\mathsf{pack}}$, where $\mathsf{d} = \mathsf{pack}+t-1$. In addition, D picks a *masking set* $\mathcal{M}$, consisting of $2{\cdot}\mathsf{pack}$ random values $\{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pack})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pack})})\}$, which are embedded in two random degree d polynomials $H^{(1)}(x)$ and $H^{(2)}(x)$ respectively at $x = \beta_1, \ldots, \beta_{\mathsf{pack}}$; we call these polynomials as *masking polynomials*. The polynomials are sent to INT, while each verifier P_i receives the values $v_{1,i}, \ldots, v_{\ell,i}, m_{1,i}, m_{2,i}$ of these polynomials at a secret evaluation point γ_i. This distribution achieves ICPoP-**Privacy**, as each secret-encoding polynomial has degree d and adversary may get at most t values on these polynomials; so it will lack pack values on each polynomial to uniquely interpolate them.

During revelation phase, to give a proof of possession of $\mathcal{S}$, INT produces a random linear combination of the values in $\mathcal{S} \cup \mathcal{M}$ by making public a random linear combiner, say e and a linear combination $C(x) \stackrel{\text{def}}{=} eH^{(1)}(x) + e^2H^{(2)}(x) + e^3G^{(1)}(x) + \ldots + e^{\ell+2}G^{(\ell)}(x)$. The values $C(\beta_1), \ldots, C(\beta_{\mathsf{pack}})$ define pack linear combinations of $\mathcal{S} \cup \mathcal{M}$ with respect to e. The pair $(e, C(x))$ is considered as a *proof of possession* of $\mathcal{S}$ (union $\mathcal{M}$) and verified as follows: each verifier locally verifies if the corresponding linear combination $em_{1,i} + e^2m_{2,i} + e^3v_{1,i} + \ldots + e^{\ell+2}v_{\ell,i}$ satisfies $C(x)$ at $x = \gamma_i$ and accordingly broadcast an `Accept` or a `Reject` message. If more than t verifiers broadcast `Accept` then the proof $(e, C(x))$ is said to be accepted, otherwise it is rejected. The proof will always be accepted for an *honest* D and INT, implying ICPoP-**Correctness1**. The size of the proof is $\mathcal{O}(n)$ (as $\mathsf{d} = \mathcal{O}(n)$), which is independent of ℓ, implying ICPoP-**Succinctness of the Proof**. No additional information about the secret-encoding polynomials is revealed from $C(x)$, thanks to the masking polynomials. If D is *honest* and INT is *corrupted* then the evaluation points of the honest verifiers will be private. So if INT gives a proof of possession of $\mathcal{S}^\star \cup \mathcal{M}^\star \neq \mathcal{S} \cup \mathcal{M}$ by revealing a linear combination of

[6] The interpretation of a proof corresponding to a set of values will be clear later during the formal presentation of our ICPoP.

$\mathcal{S}^\star \cup \mathcal{M}^\star$ through $(e, C^\star(x))$ where $C^\star(x) \neq C(x)$, then with high probability, every honest verifier will reject the proof. This is because the corresponding linear combination of the values possessed by the honest verifiers will fail to satisfy $C^\star(x)$; this implies ICPoP-**Correctness 3**.

The above mechanism, however, fails to achieve ICPoP-**Correctness 2**, as a *corrupted* D can distribute "inconsistent" polynomials and values to an *honest* INT and honest verifiers respectively; later on the proof produced by INT will be rejected by every honest verifier. To verify the consistency of the distributed information, during the authentication phase, INT "challenges" D by making public a random linear combination $A(x)$ of the received polynomials. In response, D either instructs to abort the protocol or continue, after verifying whether the $A(x)$ polynomial satisfies the corresponding random linear combination of the values held by each verifier. The idea here is that if D distributed inconsistent data, then with very high probability, any random linear combination of the distributed polynomials would fail to satisfy the corresponding linear combination of the values given to the honest verifiers. And this will be locally learned by the honest verifiers after $A(x)$ is made public. So if D still instructs to continue the protocol, then clearly D is corrupted; so later even if the proof produced in the revelation phase turns out to be inconsistent with the information held by the honest verifiers, the proof is accepted by adding an additional acceptance condition to deal with this particular case. We stress that the additional acceptance condition never gets satisfied for an *honest* D and a *corrupted* INT. The privacy of the secret-encoding polynomials is still preserved during the authentication phase (for an honest INT and D), thanks to the masking polynomials[7]. The formal steps of ICPoP are given in Fig. 3. In the protocol, if the output is `AcceptProof` then we additionally let the parties output pack linear combinations of the values in $\mathcal{S} \cup \mathcal{M}$ possessed by INT; looking ahead this will be useful in our VSS.

In Fig. 2 we present a pictorial representation of the values distributed and revealed in ICPoP.

Figure 2: Pictorial representation of the information generated and communicated during ICPoP protocol.

(a) The values communicated during Distr. The two masking polynomials of degree d are $H^{(1)}(x)$ and $H^{(2)}(x)$ (shown in blue) which embeds the masking values $\{m^{(1,1)} \ldots m^{(1,\mathsf{pack})}\}$ and $\{m^{(2,1)} \ldots m^{(2,\mathsf{pack})}\}$ respectively. The ℓ secret-encoding polynomials of degree d are $G^{(1)}(x) \cdots G^{(\ell)}(x)$ (shown in red) where $G^{(k)}(x)$ embeds pack secrets i.e $\{s^{(k,1)} \ldots s^{(k,\mathsf{pack})}\}$. All embeddings are done at $x = \beta_1, \ldots, \beta_{\mathsf{pack}}$.

$$\begin{array}{cc|ccc} H^{(1)}(x) & \Rightarrow & m^{(1,1)} & \cdots & m^{(1,\mathsf{pack})} \\ H^{(2)}(x) & \Rightarrow & m^{(2,1)} & \cdots & m^{(2,\mathsf{pack})} \\ G^{(1)}(x) & \Rightarrow & s^{(1,1)} & \cdots & s^{(1,\mathsf{pack})} \\ \vdots & & \vdots & \vdots & \vdots \\ G^{(\ell)}(x) & \Rightarrow & s^{(\ell,1)} & \cdots & s^{(\ell,\mathsf{pack})} \end{array}$$

(b) The output vector $(\mathsf{comb}_1, \ldots, \mathsf{comb}_{\mathsf{pack}}) = (C(\beta_1), \ldots, C(\beta_{\mathsf{pack}}))$ that is revealed by INT during RevealPoP (shown in blue color) via the proof $(e, C(x))$. We note that comb_k is a linear combination of the kth value embedded in $H^{(1)}(x), H^{(2)}(x), G^{(1)}(x), \ldots, G^{(\ell)}(x)$ with respect to the combiner e, for $k = 1, \ldots, \mathsf{pack}$. This is represented as the kth column in the matrix representation (shown in green color).

$$\begin{array}{cc|ccccc} H^{(1)}(x) & \Rightarrow & m^{(1,1)} & \cdots & m^{(1,k)} & \cdots & m^{(1,\mathsf{pack})} \\ H^{(2)}(x) & \Rightarrow & m^{(2,1)} & \cdots & m^{(2,k)} & \cdots & m^{(2,\mathsf{pack})} \\ G^{(1)}(x) & \Rightarrow & s^{(1,1)} & \cdots & s^{(1,k)} & \cdots & s^{(1,\mathsf{pack})} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ G^{(\ell)}(x) & \Rightarrow & s^{(\ell,1)} & \cdots & s^{(\ell,k)} & \cdots & s^{(\ell,\mathsf{pack})} \\ & & \Downarrow & & \Downarrow & & \Downarrow \\ & & \mathsf{comb}_1 & & \mathsf{comb}_k & & \mathsf{comb}_{\mathsf{pack}} \end{array}$$

$$\begin{aligned} \mathsf{comb}_k &= em^{(1,k)} + e^2 m^{(2,k)} + e^3 s^{(1,k)} + \cdots + e^{\ell+2} s^{(\ell,k)} \\ &= eH^{(1)}(\beta_k) + e^2 H^{(2)}(\beta_k) + e^3 G^{(1)}(\beta_k) + \cdots + e^{\ell+2} G^{(\ell)}(\beta_k) \\ &= C(\beta_k), \text{ where} \\ C(x) &= eH^{(1)}(x) + e^2 H^{(2)}(x) + e^3 G^{(1)}(x) + \ldots + e^{\ell+1} G^{(\ell)}(x) \end{aligned}$$

In ICPoP, the *correspondence* between a proof and a set of values is defined as follows: Let $\mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pack})}), \ldots, (s^{(\ell,1)}, \ldots, s^{(\ell,\mathsf{pack})})\}$ and $\mathcal{M} = \{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pack})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pack})})\}$. We say that a proof $(e, C(x))$ *corresponds* to $\mathcal{S} \cup \mathcal{M}$ if $C(x)$ embeds linear combination of $\mathcal{S} \cup \mathcal{M}$ with respect to e at $x = \beta_1, \ldots, \beta_{\mathsf{pack}}$; i.e. if $C(\beta_i) = em^{(1,i)} + e^2 m^{(2,i)} + e^3 s^{(1,i)} + \ldots + e^{(\ell+2)} s^{(\ell,i)}$ holds for $i \in [\mathsf{pack}]$.
The following theorem, stating the properties of ICPoP is proved in Appendix B.

[7]This explains the need for two masking polynomials: one is used to preserve the privacy of the secret-encoding polynomials during the authentication phase while the other is used to maintain the privacy during the revelation phase.

Figure 3: Efficient ICPoP protocol where $\ell \geq 1$ and $1 \leq \mathsf{pack} \leq n - t$.

$\mathsf{ICPoP}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pack}, \mathcal{S}) : \mathcal{S} = \{(s^{(1,1)}, \ldots, s^{(1,\mathsf{pack})}), \ldots, (s^{(\ell,1)}, \ldots, s^{(\ell,\mathsf{pack})})\}$

$\mathsf{Distr}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pack}, \mathcal{S} \cup \mathcal{M})$

Round 1:

- D defines a *masking set* $\mathcal{M} \stackrel{\text{def}}{=} \{(m^{(1,1)}, \ldots, m^{(1,\mathsf{pack})}), (m^{(2,1)}, \ldots, m^{(2,\mathsf{pack})})\}$ consisting of $2 \cdot \mathsf{pack}$ random elements from $\mathbb{F}$. Let $\mathsf{d} \stackrel{\text{def}}{=} \mathsf{pack} + t - 1$. Dealer D selects ℓ random *secret-encoding polynomials* $G^{(1)}(x), G^{(2)}(x), \ldots G^{(\ell)}(x)$ of degree at most d, such that $G^{(k)}(\beta_1) = s^{(k,1)}, \ldots, G^{(k)}(\beta_{\mathsf{pack}}) = s^{(k,\mathsf{pack})}$ for $k \in [\ell]$. In addition, D selects two random *masking polynomials* $H^{(1)}(x), H^{(2)}(x)$ of degree d, such that $H^{(k)}(\beta_1) = m^{(k,1)}, \ldots, H^{(k)}(\beta_{\mathsf{pack}}) = m^{(k,\mathsf{pack})}$ for $k \in [2]$. For each verifier $P_i \in \mathcal{P}$, dealer D selects a random *evaluation point* γ_i such that $\gamma_i \in \mathbb{F} \setminus \{\beta_1, \ldots, \beta_{\mathsf{pack}}\}$.
- D *gives* $\mathcal{S} \cup \mathcal{M}$ to INT by sending $G^{(1)}(x), \ldots G^{(\ell)}(x), H^{(1)}(x)$ and $H^{(2)}(x)$ to INT. To each verifier $P_i \in \mathcal{P}$, dealer D sends $(\gamma_i, v_{1,i}, v_{2,i}, \ldots v_{\ell,i}, m_{1,i}, m_{2,i})$, where $v_{k,i} \stackrel{\text{def}}{=} G^{(k)}(\gamma_i)$ for $k \in [\ell]$ and $m_{k,i} \stackrel{\text{def}}{=} H^{(k)}(\gamma_i)$ for $k \in [2]$.

Local Computation by INT: Let $\overline{G}^{(1)}(x), \ldots \overline{G}^{(\ell)}(x), \overline{H}^{(1)}(x)$ and $\overline{H}^{(2)}(x)$ be the polynomials received from D (if D is honest then these will be the same polynomials as selected by D). INT sets $\overline{\mathcal{S}} = \{(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,\mathsf{pack})}), \ldots, (\overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,\mathsf{pack})})\}$ and $\overline{\mathcal{M}} = \{(\overline{m}^{(1,1)}, \ldots, \overline{m}^{(1,\mathsf{pack})}), (\overline{m}^{(2,1)}, \ldots, \overline{m}^{(2,\mathsf{pack})})\}$, where $\overline{s}^{(k,1)} = \overline{G}^{(k)}(\beta_1), \ldots, \overline{s}^{(k,\mathsf{pack})} = \overline{G}^{(k)}(\beta_{\mathsf{pack}})$ for $k \in [\ell]$ and $\overline{m}^{(k,1)} = \overline{H}^{(k)}(\beta_1), \ldots, \overline{m}^{(k,\mathsf{pack})} = \overline{H}^{(k)}(\beta_{\mathsf{pack}})$ for $k \in [2]$; $\overline{\mathcal{S}} \cup \overline{\mathcal{M}}$ are considered to be *received* by INT from D.

Local Computation Each Verifier P_i: Let $(\overline{\gamma}_i, \overline{v}_{1,i}, \overline{v}_{2,i}, \ldots \overline{v}_{\ell,i}, \overline{m}_{1,i}, \overline{m}_{2,i})$ be the tuple received from D (if D is honest then this will be the same tuple as computed by D).

$\mathsf{AuthVal}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pack}, \overline{\mathcal{S}} \cup \overline{\mathcal{M}})$

Round 1: INT selects a random element $d \in \mathbb{F} \setminus \{0\}$ and broadcasts $(d, A(x))$, where $A(x) \stackrel{\text{def}}{=} d\overline{H}^{(1)}(x) + d^2\overline{H}^{(2)}(x) + d^3\overline{G}^{(1)}(x) + d^4\overline{G}^{(2)}(x) + \ldots d^{\ell+2}\overline{G}^{(\ell)}(x)$.

Round 2: Upon receiving $(d, A(x))$ from the broadcast of INT, D checks if $A(\gamma_i) = dm_{1,i} + d^2 m_{2,i} + d^3 v_{1,i} + d^4 v_{2,i} \ldots d^{\ell+2} v_{\ell,i}$ holds for every $P_i \in \mathcal{P}$. If not then it broadcasts an `Abort` messages, else it broadcasts an `OK` message.

$\mathsf{RevealPoP}(\mathsf{D}, \mathsf{INT}, \mathcal{P}, \ell, \mathsf{pack}, \overline{\mathcal{S}} \cup \overline{\mathcal{M}})$: Executed only if D broadcasted `OK` during AuthVal.

Round 1: INT chooses a random element $e \in \mathbb{F} \setminus \{0\}$ and broadcasts $(e, C(x))$ as a *proof of possession* of $\overline{\mathcal{S}} \cup \overline{\mathcal{M}}$, where $C(x) \stackrel{\text{def}}{=} e\overline{H}^{(1)}(x) + e^2\overline{H}^{(2)}(x) + e^3\overline{G}^{(1)}(x) + e^4\overline{G}^{(2)}(x) \ldots e^{\ell+2}\overline{G}^{(\ell)}(x)$.

Round 2: Upon receiving the broadcast of $(e, C(x))$ from INT, every verifier $P_i \in \mathcal{P}$ locally verifies the following conditions:

- $C(\overline{\gamma}_i) \stackrel{?}{=} e\overline{m}_{i,1} + e^2\overline{m}_{i,2} + e^3\overline{v}_{1,i} + \ldots e^{\ell+2}\overline{v}_{\ell,i}$ — we call this condition **C1**.
- $A(\gamma_i) \neq d\overline{m}_{1,i} + d^2\overline{m}_{2,i} + d^3\overline{v}_{1,i} + d^4\overline{v}_{2,i} + \ldots d^{\ell+2}\overline{v}_{\ell,i}$ holds during AuthVal — we call this condition **C2**.

Verifier P_i broadcasts `Accept` if condition **C1** or **C2** is true for P_i, else it broadcasts `Reject`.

Output Determination: If more than t verifiers broadcast `Accept` then each verifier P_i outputs `AcceptProof` along with the vector $(\mathsf{comb}_1, \ldots, \mathsf{comb}_{\mathsf{pack}}) \stackrel{\text{def}}{=} (C(\beta_1), \ldots, C(\beta_{\mathsf{pack}}))$, else each verifier P_i outputs `RejectProof`.

Theorem 3.1. *Protocols* $(\mathsf{Distr}, \mathsf{AuthVal}, \mathsf{RevealPoP})$ *constitute a* $(1-\epsilon)$*-secure* ICPoP *for* $L = \ell \times \mathsf{pack}$ *values with* $\ell \geq 1$ *and* $1 \leq \mathsf{pack} \leq n - t$*, where* $\epsilon = \max\{\frac{n\ell}{|\mathbb{F}|-1}, \frac{n\mathsf{d}}{|F|-\mathsf{pack}}\}$ *and* $\mathsf{d} = \mathsf{pack} + t - 1$*. The protocol has communication complexity* $\mathcal{PC}(\mathcal{O}(n\ell))$ *and* $\mathcal{BC}(\mathcal{O}(n))$.

Transferability of ICPoP: In our VSS protocol we will use ICPoP as follows: after receiving $\mathcal{S} \cup \mathcal{M}$ from D via the secret-encoding and masking polynomials, INT will send these polynomials (and hence $\mathcal{S} \cup \mathcal{M}$) to another designated party, say $P_R \in \mathcal{P}$ (if INT is corrupted then it can send incorrect polynomials to P_R). Later on, party P_R will act as an INT and produce a proof of possession of $\mathcal{S} \cup \mathcal{M}$, which got "transferred" to P_R from INT; the proof gets verified with respect to the verification information held by the verifiers. This transfer of $\mathcal{S} \cup \mathcal{M}$ will satisfy all the properties of ICPoP, imagining P_R as the new INT. Specifically if D is *honest* and both INT and P_R are *honest*, then the privacy will hold. Moreover if P_R produces a proof of possession of incorrect sets (this can be the case if either INT or P_R is corrupted), then the proof gets rejected. If D is

corrupted and both INT and P_R are honest then the proof given by P_R will be accepted.

4 Statistical VSS with a Quadratic Overhead

We present a 4-round VSS protocol Sh to t-share $\ell \times (n-t) = \Theta(n\ell)$ values with communication complexity $\mathcal{PC}(\mathcal{O}(n^3\ell))$ and $\mathcal{BC}(\mathcal{O}(n^3))$. So for sufficiently large ℓ, the broadcast complexity will be *independent* of ℓ. For simplicity, we will present a 5-round statistical VSS protocol Sh-Single for sharing a single secret. We will then explain how to reduce the number of rounds of Sh-Single from five to four. Finally we extend this four round Sh-Single to get Sh. We first discuss a protocol Poly-Check adapted from [36], used in our VSS.

4.1 Verifiably Distributing Values on Bivariate Polynomials of Degree at most t

In our VSS protocol we will come across the following situation: D will select L bivariate polynomials $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$, each of degree at most t and send the ith row polynomials $f_i^{(1)}(x),\ldots,f_i^{(L)}(x)$ of $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$ respectively to each P_i; we stress that the corresponding column polynomials are retained by D. The parties now want to publicly verify if there is a set of at least $t+1$ *honest* parties, who received row polynomials, lying on L unique bivariate polynomials of degree at most t without revealing any additional information about the polynomials. For this we use a two round protocol Poly-Check (see Fig 4), which is adapted from an asynchronous protocol for the same purpose, presented in [36].

In the protocol Poly-Check, there is a designated *verifier* V, who challenges D to broadcast a random linear combination of the n column polynomials of all the bivariate polynomials selected by D. Specifically V provides a challenge combiner, say r and in response D makes public a linear combination of its column polynomials with respect to r; to maintain the privacy of the column polynomials, this linear combination is blinded by a random degree t *blinding polynomial* $B(y)$, selected by D, with each party P_i having a value on this polynomial. Corresponding to the linear combination of the column polynomials produced by D, each party P_i makes public a linear combination of n values of all its row polynomials, with respect to the combiner r, which is blinded by the value of $B(y)$ possessed by it. The idea here is the following: if indeed there exists a set of $t+1$ honest parties that we are looking for, then the values of the row polynomials possessed by these parties will define degree t column polynomials. And these column and row polynomials will be "pair-wise consistent". Based on this idea we check if the blinded linear combination of the column polynomials produced by D is of degree t. Moreover it is also checked if there exists a *witness set* $\mathcal{W}^{(\mathsf{V})}$ of at least $2t+1$ parties, such that their blinded linear combination of row polynomial values satisfies the linear combination produced by D. If any one of the above conditions is not satisfied the parties output $\perp$, otherwise they output $\mathcal{W}^{(\mathsf{V})}$. It is ensured that if V is *honest*, then except with probability $\frac{nL}{|\mathbb{F}|}$, the honest parties in $\mathcal{W}^{(\mathsf{V})}$ constitute the desired set of row polynomial holders.
The properties of Poly-Check are stated in Lemma 4.1; we refer to [36] for the complete proof.

Lemma 4.1 (**Properties of Protocol** Poly-Check **[36]**). *In protocol* Poly-Check*, the following holds:*

- *If* D *is honest then every honest party outputs a $\mathcal{W}^{(\mathsf{V})}$ set which includes all the honest parties. Moreover the row polynomials of the honest parties in $\mathcal{W}^{(\mathsf{V})}$ will lie on $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$. Furthermore* Adv *gets no additional information about $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$ in the protocol.*
- *If* D *is corrupted and* V *is honest and if the parties output a $\mathcal{W}^{(\mathsf{V})}$, then except with probability at most $\frac{nL}{|\mathbb{F}|}$, there exists L bivariate polynomials, say $\overline{F}^{(1)}(x,y),\ldots,\overline{F}^{(L)}(x,y)$, of degree at most t, such that the row polynomials of the honest parties in $\mathcal{W}^{(\mathsf{V})}$ lie on $\overline{F}^{(1)}(x,y),\ldots,\overline{F}^{(L)}(x,y)$.*
- *The protocol requires two rounds and has communication complexity $\mathcal{BC}(\mathcal{O}(n))$.*

Figure 4: Checking the consistency of row polynomials distributed by D under the supervision of a designated verifier V. The inputs for (an honest) D are L secret bivariate polynomials $F^{(1)}(x,y),\ldots,F^{(L)}(x,y)$ of degree at most t and a secret blinding polynomial $B(y)$ of degree at most t. The inputs for (an honest) party P_i are L row polynomials $\overline{f}_i^{(1)}(x),\ldots,\overline{f}_i^{(L)}(x)$ of degree at most t and a share $\overline{b}_i$ of blinding polynomial. If D and P_i are honest then these values are private and $\overline{f}_i^{(k)}(x)=F^{(k)}(x,\alpha_i)$ and $\overline{b}_i=B(\alpha_i)$ will hold for each $k\in[L]$.

Poly-Check(D, V, $\mathcal{P}$, L, $\{F^{(1)}(x,y),\ldots,F^{(L)}(x,y),B(y)\}$, $\{\overline{f}_i^{(1)}(x),\ldots,\overline{f}_i^{(L)}(x),\overline{b}_i\}_{i\in[n]}$)

Round 1: Verifier V selects a random combiner $r\in\mathbb{F}\setminus\{0\}$ and broadcasts r.

Round 2: The parties on receiving r from the broadcast of V do the following:

- D broadcasts the polynomial $E(y)\stackrel{\text{def}}{=}B(y)+rg_1^{(1)}(y)+r^2g_2^{(1)}(y)+\ldots+r^ng_n^{(1)}(y)+r^{(n+1)}g_1^{(2)}(y)+r^{(n+2)}g_2^{(2)}(y)+\ldots+r^{2n}g_n^{(2)}(y)+\ldots+r^{(L-1)n+1}g_1^{(L)}(y)+r^{(L-1)n+2}g_2^{(L)}(y)+\ldots+r^{Ln}g_n^{(L)}(y)$. Here $g_i^{(k)}(y)=F^{(k)}(\alpha_i,y)$ for $k\in[L]$ and $i\in[n]$.
- Each party $P_i\in\mathcal{P}$ (including D) broadcasts the linear combination $e_i\stackrel{\text{def}}{=}\overline{b}_i+r\overline{f}_i^{(1)}(\alpha_1)+r^2\overline{f}_i^{(1)}(\alpha_2)+\ldots+r^n\overline{f}_i^{(1)}(\alpha_n)+r^{(n+1)}\overline{f}_i^{(2)}(\alpha_1)+r^{(n+2)}\overline{f}_i^{(2)}(\alpha_2)+\ldots+r^{2n}\overline{f}_i^{(2)}(\alpha_n)+\ldots+r^{(L-1)n+1}\overline{f}_i^{(L)}(\alpha_1)+r^{(L-1)n+2}\overline{f}_i^{(L)}(\alpha_2)+\ldots+r^{Ln}\overline{f}_i^{(L)}(\alpha_n)$

Output determination: If $E(y)$ has degree more than t then each party $P_j\in\mathcal{P}$ outputs $\perp$ and terminate. Else each party $P_j\in\mathcal{P}$ creates a *witness set* $\mathcal{W}^{(\mathsf{V})}$, initialized to $\emptyset$ and then does the following:

- Include party P_i to $\mathcal{W}^{(\mathsf{V})}$ if the relation $E(\alpha_i)\stackrel{?}{=}e_i$ is true.
- If $|\mathcal{W}^{(\mathsf{V})}|\geq 2t+1$ then P_j outputs $\mathcal{W}^{(\mathsf{V})}$, else P_j outputs $\perp$.

4.2 Five Round Statistical VSS for a Single Secret

To t-share s, D selects a random *secret-carrying* bivariate polynomial $F(x,y)$ of degree at most t such that $s=F(0,0)$. The ith row polynomial $f_i(x)$ of $F(x,y)$ is given to each P_i. We stress that *only* the row polynomials are distributed. The parties then verify the consistency of the distributed polynomials by publicly verifying the existence of a set VCORE of at least $2t+1$ parties, such that the row polynomials of the *honest* parties in VCORE lie on a unique bivariate polynomial, say $\overline{F}(x,y)$, of degree at most t. For this, n instances of Poly-Check are executed (one on the behalf of each party playing the role of the designated verifier V) and it is verified if there is common subset of at least $2t+1$ parties, present across all the generated witness sets. As there will be at least one instance of Poly-Check executed on the behalf of an honest verifier, clearly the common subset of $2t+1$ parties satisfies the properties of VCORE. To maintain the privacy of the row polynomials during the Poly-Check instances, n independent *blinding polynomials* are used by D, one for each instance. If a VCORE is found, then we say that D has "committed" the secret $\overline{s}=\overline{F}(0,0)$ to the parties in VCORE via their row polynomials and the next goal will be to ensure that each party P_j obtains its column polynomial $\overline{g}_j(y)$ of $\overline{F}(x,y)$; party P_j can then output its share $\overline{s}_j=\overline{g}_j(0)$ of $\overline{s}$ and hence $\overline{s}$ will be t-shared via $\overline{F}(x,0)$. If D is *honest* then $\overline{F}(x,y)=F(x,y)$ will hold (and hence $\overline{s}=s$), as VCORE will include all the honest parties.

To enable P_j obtain $\overline{g}_j(y)$, each $P_i\in$ VCORE can send the common point $\overline{f}_i(\alpha_j)$ on $\overline{g}_j(y)$ to P_j, where $\overline{f}_i(\alpha_j)$ denotes the jth value on the ith row polynomial received by P_i (if D is honest then $\overline{f}_i(\alpha_j)=f_i(\alpha_j)$ holds). The honest parties in VCORE will always send the correct values; however the corrupted parties may send incorrect values. Due to insufficient redundancy in the received $\overline{f}_i(\alpha_j)$ values, party P_j cannot error-correct them (for this we require |VCORE| to be of size at least $3t+1$). The way out is that P_j gives a *proof of possession* of the $\overline{f}_i(\alpha_j)$ values received from the parties P_i in VCORE. Namely the values on the row polynomials are initially distributed by D by executing instances of Distr. There will be n^2 such instances and instance Distr_{ij} is executed to distribute $f_i(\alpha_j)$ to P_i, considering P_i as an INT; the corresponding instances

$\mathsf{AuthVal}_{ij}$ are also executed and it is ensured that the AuthVal instances, involving any party from VCORE as an INT, is not aborted by D. Now when a party P_i in VCORE sends $\overline{f}_i(\alpha_j)$ to P_j, party P_j acts as an INT and publicly gives a proof of possession of $\overline{f}_i(\alpha_j)$ by executing an instance $\mathsf{RevealPoP}_{ji}$ of RevealPoP. The idea is to use the *transferability* property of ICPoP to identify the incorrectly transferred values. Namely if D is *honest* and an incorrect $\overline{f}_i(\alpha_j)$ is transferred to P_j, then the corresponding proof gets rejected during $\mathsf{RevealPoP}_{ji}$ and P_j discard such values.

Unfortunately, if D is *corrupted* then the above mechanism alone is not sufficient for P_j to robustly reconstruct $\overline{g}_j(y)$. Because a *corrupted* P_i in VCORE can then transfer an incorrect $\overline{f}_i(\alpha_j)$ to P_j and still the proof will get accepted; this is because if *both* D and INT are corrupted, then INT will know the full auxiliary and verification information involved in ICPoP. As a result, P_j will end up not reconstructing a degree t column polynomial from the received $\overline{f}_i(\alpha_j)$ values. To deal with this particular case, we ensure that the $\mathcal{M}$ sets used by D in the ICPoP instances have a similar "structure" as the corresponding $\mathcal{S}$ sets. Specifically, D selects two random *masking* bivariate polynomials $M^{(1)}(x,y)$ and $M^{(2)}(x,y)$ each of degree at most t. Let $m_i^{(1)}(x), m_i^{(2)}(x)$ denote the corresponding row polynomials. The instances Distr_{ij} are executed by setting $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathcal{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ (thus $\ell = 1$ and $\mathsf{pack} = 1$ in these instances). The corresponding $\mathsf{AuthVal}_{ij}$ instances are executed with $\overline{\mathcal{S}}_{ij} = \{\overline{f}_i(\alpha_j)\}$ and $\overline{\mathcal{M}}_{ij} = \{\overline{m}_i^{(1)}(\alpha_j), \overline{m}_i^{(2)}(\alpha_j)\}$, which denotes the $\mathcal{S}$ and $\mathcal{M}$ sets respectively received by P_i during Distr_{ij} (if D is honest then these will be the same as $\mathcal{S}_{ij}$ and $\mathcal{M}_{ij}$). The existence of VCORE will now imply that D has committed a secret-carrying polynomial, say $\overline{F}(x,y)$ and two masking bivariate polynomials, say $\overline{M}^{(1)}(x,y), \overline{M}^{(2)}(x,y)$ to the parties in VCORE, where all these polynomials have degree at most t. It follows that any linear combination of the column polynomials $\overline{F}(\alpha_j,y), \overline{M}^{(1)}(\alpha_j,y)$ and $\overline{M}^{(2)}(\alpha_j,y)$ will be a degree t univariate polynomial. And this property is used by P_j to identify the correctly transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ sets. Namely the values in the transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ sets should lie on degree t univariate polynomials and hence any random linear combination of these sets should also lie on a degree t polynomial. Based on this observation, party P_j selects a *common* random combiner, say e_j, for *all* the transferred $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ sets and publicly reveals a linear combination of these $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ sets via the $\mathsf{RevealPoP}_{ji}$ instances. It is then publicly verified if these linearly combined values lie on a degree t polynomial. If not then it implies that D is corrupted and it is discarded; see Fig. 5 for the formal details. For the ease of understanding, a pictorial representation of the information distributed in Sh-Single is given in Fig. 6.

The following theorem, which states the properties of Sh-Single is proved in Appendix C.

Theorem 4.2. Sh-Single *is a five round VSS protocol for a single secret, satisfying the requirements of VSS except with probability* $\frac{n^3 t}{|\mathbb{F}|-1}$. *The protocol has communication complexity* $\mathcal{PC}(\mathcal{O}(n^3))$ *and* $\mathcal{BC}(\mathcal{O}(n^3))$.

From Five Rounds to Four Rounds: In Sh-Single, the instances of RevealPoP which start getting executed during Round 4 can be instead instantiated during Round 3 itself. Namely irrespective of the formation of VCORE, each party P_j starts executing the instance $\mathsf{RevealPoP}_{ji}$ corresponding to *each* party $P_i \in \mathcal{P}$, based on the set of values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ which were transferred to P_j by P_i during Round 2. Next VCORE is computed and if P_i is found not to be present in VCORE, then the instance $\mathsf{RevealPoP}_{ji}$ can be halted; otherwise the remaining steps of the $\mathsf{RevealPoP}_{ji}$ instance are executed during Round 4. Based on this modification, Sh-Single now requires four rounds, while the rest of the properties remain the same.

Sharing $\ell \times (n-t)$ Secrets: To share $\ell \times (n-t)$ secrets, the underlying instances of Distr, AuthVal and RevealPoP are executed to deal with $\ell \times \mathsf{pack}$ values simultaneously, where $\mathsf{pack} = n-t$. The steps for consistency checking of the values transferred by the parties in VCORE are also generalized to deal with $\ell \times (n-t)$ values. With these modifications, we get a four round Sh for sharing $\ell(n-t)$ values; for details, see Appendix C. The properties of Sh are stated in Theorem 4.3, which is proved in Appendix C.

Theorem 4.3. Sh *is a four round VSS for* $\ell \times (n-t)$ *values, with an error probability of* $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\ell}{|\mathbb{F}|-1}\}$. *The protocol has communication complexity* $\mathcal{PC}(\mathcal{O}(n^3\ell))$ *and* $\mathcal{BC}(\mathcal{O}(n^3))$.

Figure 5: VSS for sharing a single secret

Sh-Single$(\mathsf{D}, \mathcal{P}, s)$

Round 1: Dealer D does the following:

- Select a random *secret-carrying bivariate polynomial* $F(x,y)$ of degree at most t with $F(0,0) = s$. Select two random *masking bivariate polynomials* $M^{(1)}(x,y)$ and $M^{(2)}(x,y)$, each of degree at most t. In addition select n random *blinding univariate polynomials* $B^{(P_1)}(y), \ldots, B^{(P_n)}(y)$, each of degree at most t, where $B^{(P_i)}$ is *associated* with party $P_i \in \mathcal{P}$. Corresponding to each $P_i \in \mathcal{P}$, compute row polynomials $f_i(x) \stackrel{\text{def}}{=} F(x, \alpha_i), m_i^{(1)}(x) \stackrel{\text{def}}{=} M^{(1)}(x, \alpha_i), m_i^{(2)}(x) \stackrel{\text{def}}{=} M^{(2)}(x, \alpha_i)$ and share-vector $(b_i^{(P_1)}, \ldots, b_i^{(P_n)})$ of blinding polynomials, where $b_i^{(P_j)} \stackrel{\text{def}}{=} B^{(P_j)}(\alpha_i)$ for $j \in [n]$. Let $\mathcal{S}_{ij} \stackrel{\text{def}}{=} \{f_i(\alpha_j)\}$ and $\mathcal{M}_{ij} \stackrel{\text{def}}{=} \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ for $i, j \in [n]$.
- To each $P_i \in \mathcal{P}$, send $(b_i^{(P_1)}, \ldots, b_i^{(P_n)})$. In addition, for $j \in [n]$, execute an instance Distr$(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \mathcal{S}_{ij} \cup \mathcal{M}_{ij})$ of Distr to give $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$ to P_i, considering P_i as an INT. Let Distr$_{ij}$ denote the corresponding instance of Distr.

Round 2: Each $P_i \in \mathcal{P}$ (including D) does the following: let $\overline{\mathcal{S}}_{ij} = \{\overline{f}_{ij}\}$ and $\overline{\mathcal{M}}_{ij} = \{\overline{m}_{ij}^{(1)}, \overline{m}_{ij}^{(2)}\}$ be the secret and masking set respectively received from D in Distr$_{ij}$. In addition, let $(\overline{b}_i^{(P_1)}, \ldots, \overline{b}_i^{(P_n)})$ denote the vector received [a]from D. Let $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ be the polynomials defined by the points $\{(\alpha_j, \overline{f}_{ij})\}_{j\in[n]}, \{(\alpha_j, \overline{m}_{ij}^{(1)})\}_{j\in[n]}$ and $\{(\alpha_j, \overline{m}_{ij}^{(2)})\}_{j\in[n]}$ respectively. If these polynomials are not of degree t then P_i broadcasts $(\texttt{Abort}, P_i)$, else it does the following:

- Transfer $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ to P_j by sending all the information received from D in the instance Distr$_{ij}$.
- As an INT, execute the steps of Round 1 of an instance AuthVal$(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij})$ of AuthVal, corresponding to the instance Distr$_{ij}$, for $j \in [n]$. Let this instance of AuthVal be denoted as AuthVal$_{ij}$.
- As a verifier V, execute the steps of Round 1 of an instance Poly-Check$(\mathsf{D}, P_i, \mathcal{P}, 3, \{M^{(1)}(x,y), M^{(2)}(x,y), F(x,y), B^{(P_i)}(y)\}, \{\overline{m}_j^{(1)}(x), \overline{m}_j^{(2)}(x), \overline{f}_j(x), \overline{b}_j^{(P_i)}\}_{j\in[n]})$ of Poly-Check; denote this instance as Poly-Check$^{(P_i)}$.

Round 3: Each $P_i \in \mathcal{P}$ (including D) does the following: If $(\texttt{Abort}, \star)$ message is received from the broadcast of more than t parties then discard D and abort Sh-Single. Else P_i does the following:

- Corresponding to each $j, k \in [n]$, participate as a verifier during Round 2 of AuthVal, in the instances AuthVal$_{jk}$
- Execute the steps of Round 2 of Poly-Check, corresponding to the instances Poly-Check$^{(P_1)}, \ldots,$ Poly-Check$^{(P_n)}$.
- **[Additional steps, If $P_i = \mathsf{D}$]** — In addition to the above steps, P_i executes the following steps if P_i is D:
  - As a D, execute the steps of Round 2 of AuthVal, corresponding to the instances AuthVal$_{jk}$ for each $j, k \in [n]$.
  - As a D, execute the steps of Round 2 of Poly-Check, corresponding to Poly-Check$^{(P_1)}, \ldots,$ Poly-Check$^{(P_n)}$.

Computation of VCORE — Every party $P_i \in \mathcal{P}$ (including D) executes the following steps:

- If in any of the instances Poly-Check$^{(P_1)}, \ldots,$ Poly-Check$^{(P_n)}$ the output is $\perp$, then discard D and abort Sh-Single.
- Let $\mathcal{W}^{(P_1)}, \ldots, \mathcal{W}^{(P_n)}$ denote the witness sets obtained in Poly-Check$^{(P_1)}, \ldots,$ Poly-Check$^{(P_n)}$ respectively. If $|\mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}| < 2t+1$, then discard D and abort Sh-Single. Else set VCORE $\stackrel{\text{def}}{=} \mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}$.
- If there exists any $P_j \in$ VCORE, such that D broadcasted Abort message in some instance AuthVal$_{jk}$ involving P_j as an INT, where $k \in [n]$, then remove P_j from VCORE. If finally $|\text{VCORE}| < 2t+1$ then discard D and abort Sh-Single.

Round 4: Each party $P_j \in \mathcal{P}$ executes the following steps:

- Corresponding to each $P_i \in$ VCORE, act as an INT and execute the steps of Round 1 of an instance RevealPoP$(\mathsf{D}, P_j, \mathcal{P}, 1, 1, \overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij})$ of RevealPoP, to reveal a random linear combination of the values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$, which were transferred from P_i to P_j during Round 2 of Sh-Single. In all these instances of RevealPoP, party P_j uses the same random combiner, say e_j. Let these instances of RevealPoP be denoted by RevealPoP$_{ji}$.

Round 5: Every party $P_k \in \mathcal{P}$ (including D) acts as a verifier and executes the steps of Round 2 of RevealPoP, corresponding to the instances RevealPoP$_{ji}$, where $j \in [n]$ and $P_i \in$ VCORE.

Consistency checking of the values transferred by the parties in VCORE: Each $P_k \in \mathcal{P}$ verifies the following for each $P_j \in \mathcal{P}$:

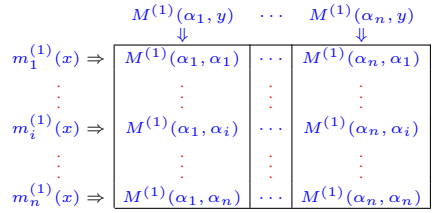
- Let sup_j denote the set of all $P_i \in$ VCORE, such that in the corresponding RevealPoP$_{ji}$ instances, the output is AcceptProof, along with a linear combination of values, say comb$_{ji}$.
- Discard D and abort Sh-Single if $\{(\alpha_i, \text{comb}_{ji})\}_{P_i \in \text{sup}_j}$ lie on a polynomial of degree more than t.

Share determination — Each $P_j \in \mathcal{P}$ interpolates a polynomial $\overline{g}_j(y)$ through $\{(\alpha_i, \overline{f}_{ij})\}_{P_i \in \text{sup}_j}$, where $\overline{\mathcal{S}}_{ij} = \{\overline{f}_{ij}\}$ denotes the secret set transferred to P_j from P_i during Round 2 of Sh-Single. Party P_j outputs $\overline{s}_j = \overline{g}_j(0)$ as its share and terminates.

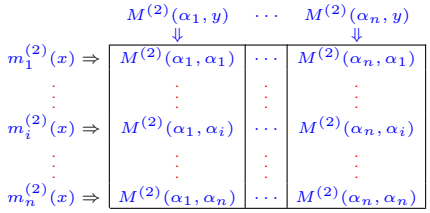
[a] If D is honest then $\overline{\mathcal{S}}_{ij} = \mathcal{S}_{ij}, \overline{\mathcal{M}}_{ij} = \mathcal{M}_{ij}$ and $(\overline{b}_i^{(P_1)}, \ldots, \overline{b}_i^{(P_n)}) = (b_i^{(P_1)}, \ldots, b_i^{(P_n)})$ holds.

Figure 6: Pictorial representation of the values distributed in Sh-Single protocol.

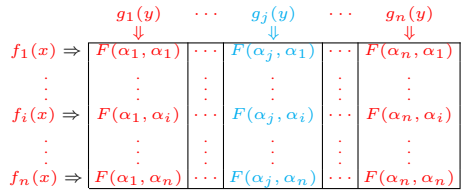
(a) $M^{(1)}(x, y)$ with ith row being possessed by P_i.



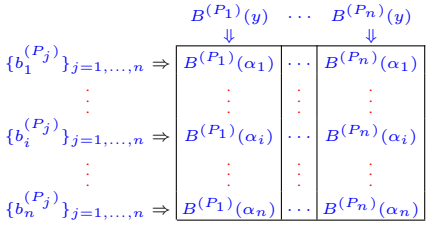
(b) $M^{(2)}(x, y)$ with ith row being possessed by P_i.



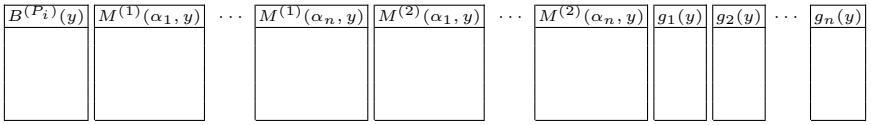
(c) $F(x, y)$ with the ith row being possessed by P_i.



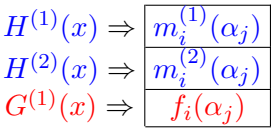
(d) Blinding polynomials with ith row being possessed by P_i.



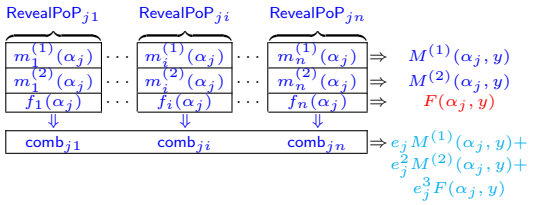
(e) Linear combination of the polynomials that are revealed during Poly-Check$^{(P_i)}$.



(f) $\mathsf{Distr}_{ij} = \mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, 1, 1, \mathcal{S}_{ij} \cup \mathcal{M}_{ij})$ where $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathcal{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$ for $i, j \in [n]$. Refer to the corresponding figure 2a which shows the distribution of values during Distr. We observe that for Distr_{ij}, $\ell = 1$ $\mathsf{pack} = 1$.



(g) $\mathsf{RevealPoP}_{ji}$ instances executed by Party P_j corresponding to the parties $P_i \in \mathsf{VCORE}$. The same random combiner e_j is used in all these instances. comb_{ji} denotes the linear combination of values output during $\mathsf{RevealPoP}_{ji}$. This is analogous to figure 2b with $\ell = 1, \mathsf{pack} = 1$.



$$\mathsf{comb}_{j1} = e_j m_1^{(1)}(\alpha_j) + e_j^2 m_1^{(2)}(\alpha_j) + e_j^3 f_1(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{ji} = e_j m_i^{(1)}(\alpha_j) + e_j^2 m_i^{(2)}(\alpha_j) + e_j^3 f_i(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{jn} = e_j m_n^{(1)}(\alpha_j) + e_j^2 m_n^{(2)}(\alpha_j) + e_j^3 f_n(\alpha_j)$$

$\{m_1^{(1)}(\alpha_j) \cdots m_n^{(1)}(\alpha_j)\}$ defines $M^{(1)}(\alpha_j, y)$ (refer fig 6a). Similarly $\{m_1^{(2)}(\alpha_j) \cdots m_n^{(2)}(\alpha_j)\}$ defines $M^{(2)}(\alpha_j, y)$ (refer fig 6b). $\{f_1(\alpha_j), f_2(\alpha_j) \cdots f_n(\alpha_j)\}$ defines $F(\alpha_j, y)$ (refer fig 6c). Therefore $e_j M^{(1)}(\alpha_j, y) + e_j^2 M^{(2)}(\alpha_j, y) + e_j^3 F(\alpha_j, y)$ is a t degree polynomial defined by the comb_{ji} values

4.3 Comparison of Our VSS with Existing Protocols

Our VSS protocol is *statistically-secure* and requires $(4, 3)$ rounds. It generates Shamir sharing of $\Theta(n\ell)$ secrets and requires a communication of $\mathcal{O}(n^3\ell)$ field elements over the point-to-point channels and broadcast of $\mathcal{O}(n^3)$ field elements. By instantiating the broadcast in our protocol by a communication-optimal reliable broadcast protocol over the point-to-point channels [32], the *total* communication complexity of our VSS protocol becomes $\mathcal{PC}(\mathcal{O}(n^3\ell + n^5))$ for Shamir-sharing $\Theta(n\ell)$ secrets. If ℓ is sufficiently large, say $\ell = \Omega(n^2)$, then the total communication complexity of our protocol will be $\mathcal{PC}(\mathcal{O}(n^3\ell))$ for Shamir-sharing $\Theta(n\ell)$ secrets; hence the overhead per secret-shared value is *quadratic*, namely $\mathcal{O}(n^2)$.

Our VSS protocol is to be compared with the previous best VSS protocols of [25, 28] with $t < n/3$, which are used in the hybrid MPC protocols of [15]; both these VSS protocols are *perfectly-secure*. The VSS protocol of [25] requires $(4, 3)$ rounds, while the protocol of [28] requires $(3, 1)$ rounds. To share ℓ secrets, the VSS protocol of [25] has communication complexity $\mathcal{PC}(\mathcal{O}(n^2\ell)) + \mathcal{BC}(\mathcal{O}(n^2\ell))$, while the VSS protocol of [28] has communication complexity $\mathcal{PC}(\mathcal{O}(n^3\ell)) + \mathcal{BC}(\mathcal{O}(n^3\ell))$. The VSS protocols of [25, 28] has broadcast communication complexity, which is *dependent* on the number of secrets to be shared, namely ℓ. On the contrary, the broadcast complexity of our VSS protocol is *independent* of ℓ. Instantiating broadcast by running a reliable broadcast protocol over the point-to-point channels is an expensive task in terms of communication complexity [30]. Hence it is desirable to have a VSS protocol with broadcast complexity being independent of the number of secret-shared values.

One can make the broadcast complexity of the VSS protocols of [25, 28] independent of ℓ by using *broadcast amplification* (aka multi-valued broadcast protocols) [22]. Informally, in such a protocol, broadcasting a "sufficiently large" message of size ℓ has communication complexity $\mathcal{PC}(\mathcal{O}(n\ell)) + \mathcal{BC}(\text{poly}(n))$. So for sufficiently large ℓ, the *total* communication complexity of such broadcast protocols (after emulating the *message-independent* broadcasts inside the protocol with a communication-optimal reliable broadcast over the point-to-point channels) asymptotically becomes $\mathcal{PC}(\mathcal{O}(n\ell))$. With $t < n/3$, the most efficient multi-valued broadcast protocol is due to [37]. The protocol requires $(5, 2)$ rounds and has communication complexity $\mathcal{PC}(\mathcal{O}(\ell n)) + \mathcal{BC}(n^2)$ bits for broadcasting an ℓ-bit message; emulating the message-independent broadcasts by a reliable broadcast protocol ensures that the *total* communication complexity of the multi-valued broadcast protocol of [37] becomes $\mathcal{PC}(\mathcal{O}(n\ell+n^4))$ bits. Deploying a multi-valued broadcast protocol to emulate the broadcasts required in the existing VSS protocols will further add an $\mathcal{O}(n)$ factor in the communication complexity over the point-to-point channels. More specifically, if we deploy the multi-valued broadcast protocol of [37] to emulate the broadcasts required in the VSS protocols of [25, 28], then we get the following complexity figures: the VSS protocol of [28] will require $(7, 2)$ rounds and communication complexity $\mathcal{PC}(\mathcal{O}(\ell n^4)) + \mathcal{BC}(n^2)$ for sharing ℓ secrets. The VSS protocol of [25] will lead to a $(16, 6)$-round VSS protocol with communication complexity $\mathcal{PC}(\mathcal{O}(\ell n^3)) + \mathcal{BC}(n^2)$. So for sufficiently large ℓ, the *total* communication communication complexity of the VSS protocol of [28] and [25] becomes $\mathcal{PC}(\mathcal{O}(\ell n^4))$ and $\mathcal{PC}(\mathcal{O}(\ell n^3))$ respectively for secret-sharing ℓ values, implying an overhead of $\mathcal{O}(n^4)$ and $\mathcal{O}(n^3)$ respectively per secret-shared value. On contrary, our VSS involves an $\mathcal{O}(n^2)$ overhead per secret-shared value. Moreover in terms of the number of rounds, our VSS is better than that of [25, 28]; however the VSS of [28] is better than ours in terms of the number of broadcast rounds.

5 Efficient Statistical MPC Protocol in the Partially Synchronous Setting

Using Sh, we design a statistical MPC protocol in the partially synchronous setting. The protocol is designed in the offline-online paradigm, where in the offline phase, the parties generate t-sharing of random and private multiplication triples of the form (a, b, c), where $c = ab$. Later in the online phase, these triples are used for the shared evaluation of the circuit using the standard Beaver multiplication triple based technique [2, 3, 5, 14]. For designing the offline phase protocol, we use the protocol Sh and deploy the efficient framework of [15]. The shared evaluation of the circuit is done in a completely asynchronous fashion in the online phase. We get

the following theorem; we refer to Appendix D for the complete details.

Theorem 5.1. *Let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function expressed as an arithmetic circuit over a finite field $\mathbb{F}$, consisting of c_M and c_R multiplication and random gates respectively. Assuming that the first four communication rounds are synchronous broadcast rounds after which the entire communication is asynchronous, there exists a statistical MPC protocol to securely compute f, provided $|\mathbb{F}| \geq 4n^4(c_M + c_R)(3t + 1)2^{\kappa}$ for a given error parameter κ. The protocol has communication complexity $\mathcal{PC}(\mathcal{O}(n^2(c_M + c_R) + n^4))$ and $\mathcal{BC}(\mathcal{O}(n^4))$.*

References

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

[2] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO*, *LNCS* 576 , pp 420–432. Springer Verlag, 1991.

[3] Z. Beerliová-Trubíniová and M. Hirt. Efficient Multi-party Computation with Dispute Control. In *TCC*, *LNCS* 3876, pp 305–328. Springer Verlag, 2006.

[4] Z. Beerliová-Trubíniová and M. Hirt. Simple and Efficient Perfectly-Secure Asynchronous MPC. In *ASIACRYPT*, *LNCS* 4833, pp 376–392. Springer Verlag, 2007.

[5] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In *TCC*, *LNCS* 4948, pp 213–230. Springer Verlag, 2008.

[6] Z. Beerliová-Trubíniová, M. Hirt, and J. B. Nielsen. On the Theoretical Gap between Synchronous and Asynchronous MPC Protocols. In *PODC*, pp 211–218. ACM, 2010.

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

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pp 1–10. ACM, 1988.

[9] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous Secure Computations with Optimal Resilience (Extended Abstract). In *PODC*, pp 183–192. ACM, 1994.

[10] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority. In *CRYPTO*, *LNCS* 7417, pp 663–680. Springer, 2012.

[11] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.

[12] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *STOC*, pp 11–19. ACM, 1988.

[13] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *FOCS*, pp 383–395. IEEE Computer Society, 1985.

[14] A. Choudhury, M. Hirt, and A. Patra. Asynchronous Multiparty Computation with Linear Communication Complexity. In *DISC*, *LNCS* 8205, pp 388–402. Springer, 2013.

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

[16] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *EUROCRYPT*, *LNCS* 1592, pp 311–326. Springer, 1999.

[17] R. Cramer, I. Damgård, and U. M. Maurer. General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In *EUROCRYPT*, *LNCS* 1807, pp 316–334. Springer Verlag, 2000.

[18] I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly Secure Multiparty Computation and the Computational Overhead of Cryptography. In *EUROCRYPT*, *LNCS* 6110, pp 445–465. Springer, 2010.

[19] I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith. Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In *CRYPTO*, *LNCS* 5157, pp 241–261. Springer, 2008.

[20] I. Damgård and J. B. Nielsen. Scalable and Unconditionally Secure Multiparty Computation. In *CRYPTO*, *LNCS* 4622, pp 572–590. Springer Verlag, 2007.

[21] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In *TCC*, *LNCS* 3876, pp 329–342. Springer, 2006.

[22] M. Fitzi and M. Hirt. Optimally Efficient Multi-valued Byzantine Agreement. In *PODC*, pp 163–168. ACM Press, 2006.

[23] M. Fitzi and J. B. Nielsen. On the Number of Synchronous Rounds Sufficient for Authenticated Byzantine Agreement. In *DISC*, *LNCS* 5805, pp 449–463. Springer, 2009.

[24] M. K. Franklin and M. Yung. Communication Complexity of Secure Computation (Extended Abstract). In *STOC*, pp 699–710. ACM, 1992.

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

[26] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pp 218–229. ACM, 1987.

[27] M. Hirt, U. M. Maurer, and B. Przydatek. Efficient Secure Multi-party Computation. In *ASIACRYPT*, *LNCS* 1976, pp 143–161. Springer, 2000.

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

[29] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. *SIAM J. Comput.*, 39(5):2090–2112, 2010.

[30] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

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

[32] J. A. Garay P. Berman and K. J. Perry. Bit Optimal Distributed Consensus. In *Computer Science Research*, pages 313–322, 1992. Preliminary version appeared in STOC 89.

[33] A. Patra, A. Choudhary, T. Rabin, and C. Pandu Rangan. The Round Complexity of Verifiable Secret Sharing Revisited. In *CRYPTO*, *LNCS* 5677, pp 487–504. Springer, 2009.

[34] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient Statistical Asynchronous Verifiable Secret Sharing and Multiparty Computation with Optimal Resilience. *IACR Cryptology ePrint Archive*, 2009: 492.

[35] A. Patra, A. Choudhury, and C. Pandu Rangan. Asynchronous Byzantine Agreement with Optimal Resilience. *Distributed Computing*, 27(2):111–146, 2014.

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

[37] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *OPODIS*, *LNCS* 7109, pp 34–49. Springer, 2011.

[38] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *STOC*, pp 73–85. ACM, 1989.

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

[40] A. C. Yao. Protocols for Secure Computations. In *FOCS*, pp 160–164. IEEE Computer Society, 1982.

A Properties of Polynomials

The following properties of bivariate polynomials are well known.

Lemma A.1 ([11, 1, 36]). *Let $f_1(x), \ldots, f_\ell(x), g_1(y), \ldots, g_\ell(y)$ be degree t univariate polynomials with $t+1 \leq \ell \leq n$, such that $f_i(\alpha_j) = g_j(\alpha_i)$ holds for every $\alpha_i, \alpha_j \in \{\alpha_1, \ldots, \alpha_\ell\}$. Then there exists a unique bivariate polynomial $\overline{F}(x,y)$ of degree t, such that $f_i(x)$ and $g_i(y)$ lie on $\overline{F}(x,y)$, for $i \in [\ell]$.*

Lemma A.2 ([11, 1, 36]). *Let $f_1(x), \ldots, f_\ell(x)$ be univariate polynomials of degree at most t where $t+1 \leq \ell \leq n$. Let $F(x,y)$ and $G(x,y)$ be two bivariate polynomials of degree at most t, such that $f_i(x)$ lies on both $F(x,y)$ and $G(x,y)$ for each $i \in [\ell]$. Then $F(x,y) = G(x,y)$.*

The following properties of univariate polynomials are standard.

Lemma A.3 ([36]). *Let $G^{(1)}(x), \ldots G^{(L)}(x)$ be degree d polynomials and let $A(x) \stackrel{def}{=} eG^{(1)}(x) + \cdots + e^L G^{(L)}(x)$, where e is a random value from $\mathbb{F} \setminus \{0\}$. Let a tuple $(\gamma, v_1, v_2, \ldots v_L)$ be such that $v_i \neq G^{(i)}(\gamma)$ for some $i \in [L]$. Then except with probability at most $\frac{L-2}{|\mathbb{F}|-1}$, the condition $A(\gamma) \neq ev_1 + \ldots e^L v_L$ holds.*

Lemma A.4 ([36]). *Let $h^{(0)}(y), \ldots h^{(L)}(y)$ be $L+1$ polynomials and r be a random value from $\mathbb{F} \setminus \{0\}$. Let $h_{com}(y) \stackrel{def}{=} h^{(0)}(y) + rh^{(1)}(y) + \ldots r^L h^{(L)}(y)$. If at least one of $h^{(0)}(y), \ldots h^{(L)}(y)$ has degree more than t, then except with probability at most $\frac{L}{|\mathbb{F}|}$, the polynomial $h_{com}(y)$ will have degree more than t.*

B Proof of the Properties of ICPoP

Lemma B.1 (ICPoP-**Correctness1**). *If* D *and* INT *are honest then each honest verifier $P_i \in \mathcal{P}$ outputs* `AcceptProof` *along with $(C(\beta_1), \ldots, C(\beta_{\mathsf{pack}}))$ at the end of* RevealPoP.

Proof. If D is honest, then for each honest verifier $P_i \in \mathcal{P}$, the relationship $G^{(k)}(\gamma_i) = v_{k,i}$ holds for each $k \in [\ell]$ and also $H^{(1)}(\gamma_i) = m_{1,i}$ and $H^{(2)}(\gamma_i) = m_{2,i}$ holds. Moreover if INT is honest then it correctly broadcasts the $C(x)$ polynomial during RevealPoP and each honest verifier P_i finds that the condition C1 is true. Hence each honest verifier broadcasts `Accept`. As there are more than t honest verifiers who broadcast `Accept` messages, each honest verifier see more than t `Accept` messages and hence outputs `AcceptProof`. □

Lemma B.2 (ICPoP-**Correctness2**). *If* D *is corrupt and* INT *is honest, and if* ICPoP *proceeds to* RevealPoP, *then all honest verifiers output* `AcceptProof`, *except with probability at most $\frac{n\ell}{|\mathbb{F}|-1}$.*

Proof. We claim that if INT is *honest* and ICPoP proceeds to RevealPoP, then an *honest* verifier P_i broadcasts `Accept`, except with probability at most $\frac{\ell}{|\mathbb{F}|-1}$. Assuming that the claim is true, from the union bound it follows that the probability any honest verifier fails to broadcast an `Accept` message is at most $\frac{n\ell}{|\mathbb{F}|-1}$, as the number of honest parties is upper bounded by n. This ensures that there will be more than t `Accept` messages broadcasted by honest verifiers implying that each honest verifier outputs `AcceptProof` at the end of RevealPoP.

We next proceed to prove our claim. For this we focus on a designated *honest* verifier P_i and consider the relationship that holds between the polynomials $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x), \overline{H}^{(1)}(x), \overline{H}^{(2)}(x)$ distributed by a *corrupted* D to INT and the tuple $(\overline{\gamma}_i, \overline{v}_{1,i}, \overline{v}_{2,i}, \ldots \overline{v}_{\ell,i}, \overline{m}_{1,i}, \overline{m}_{2,i})$ distributed by D to P_i. We have two cases:

- $\overline{v}_{k,i} = \overline{G}^{(k)}(\overline{\gamma}_i)$ for each $k \in [\ell]$ and $\overline{m}_{1,i} = \overline{H}^{(1)}(\overline{\gamma}_i), \overline{m}_{2,i} = \overline{H}^{(2)}(\overline{\gamma}_i)$: In this case, the claim is true without any error as P_i will find that condition C1 is true for the $C(x)$ polynomial during RevealPoP.

- At least one of the following holds — either $\overline{v}_{k,i} \neq \overline{G}^{(k)}(\overline{\gamma}_i)$ for some $k \in [\ell]$ or $\overline{m}_{1,i} \neq \overline{H}^{(1)}(\overline{\gamma}_i)$ or $\overline{m}_{2,i} \neq \overline{H}^{(2)}(\overline{\gamma}_i)$: In this case, $A(\overline{\gamma}_i) \neq d\overline{m}_{1,i} + d^2\overline{m}_{2,i} + d^3\overline{v}_{1,i} + d^4\overline{v}_{2,i} + \ldots d^{\ell+2}\overline{v}_{\ell,i}$ holds, except with probability at most $\frac{\ell}{|\mathbb{F}|-1}$ (follows from Lemma A.3 by substituting $L = \ell + 2$). So clearly the verifier P_i will find that condition C2 is true during RevealPoP

□

Lemma B.3 (ICPoP**-Correctness3).** *If* D *is honest,* INT *is corrupted,* ICPoP *proceeds to* RevealPoP *and if the honest verifiers output* `AcceptProof`*, then except with probability at most* $\frac{n\mathsf{d}}{|\mathbb{F}|-\mathsf{pack}}$*, the proof produced by* INT *corresponds to the values in* $\mathcal{S} \cup \mathcal{M}$*.*

Proof. If ICPoP proceeds to RevealPoP then it implies that D broadcasted `OK` message during AuthVal which implies that INT broadcasted the correct $A(x)$ polynomial during AuthVal. More specifically, the condition $A(\gamma_i) = dm_{1,i} + d^2m_{2,i} + d^3v_{1,i} + d^4v_{2,i} \ldots d^{\ell+2}v_{\ell,i}$ holds for every verifier $P_i \in \mathcal{P}$. This further implies that during RevealPoP, the condition C2 is never satisfied for any honest verifier P_i. To prove the lemma statement, we have to consider the case when a corrupted INT reveals a polynomial $C^\star(x) \neq eH^{(1)}(x) + e^2H^{(2)}(x) + e^3G^{(1)}(x) + e^4G^{(2)}(x) + \ldots + e^{\ell+2}G^{(\ell)}(x)$ during RevealPoP (if INT produces the correct $C(x)$ polynomial then the lemma statement is true without any error probability). We claim that the probability that an honest verifier $P_i \in \mathcal{P}$ broadcasts `Accept` message corresponding to $C^\star(x)$ is at most $\frac{\mathsf{d}}{|\mathbb{F}|-\mathsf{pack}}$. Assuming that the claim is true, it follows via the union bound that the probability that any honest verifier broadcasts `Accept` message corresponding to $C^\star(x)$ is at most $\frac{n\mathsf{d}}{|\mathbb{F}|-\mathsf{pack}}$, as the number of honest verifiers is upper bounded by n. This implies that there can be at most t `Accept` messages corresponding to $C^\star(x)$, broadcasted by t potentially corrupted verifiers, implying that each honest verifier outputs `RejectProof`.

We next prove our claim. For this we focus on a designated honest verifier P_i. As discussed above, the condition C2 never happens for P_i. So P_i broadcasts `Accept` message only if condition C1 holds for P_i. In order that C1 is satisfied for P_i, it should hold that $C^\star(\gamma_i) = C(\gamma_i)$. However since D is honest, the adversary will have no information about the secret evaluation point γ_i. So the only way a corrupted INT can ensure that $C^\star(\gamma_i) = C(\gamma_i)$ holds is by correctly guessing γ_i, which it can do with probability at most $\frac{\mathsf{d}}{|\mathbb{F}|-\mathsf{pack}}$. This is because two different polynomials of degree at most d can have at most d common roots and $\gamma_i \in \mathbb{F} \setminus \{\beta_1, \ldots, \beta_{\mathsf{pack}}\}$. □

Lemma B.4 (ICPoP**-Privacy).** *If* D *and* INT *are honest, then the information obtained by* Adv *during* ICPoP *is independent of the values in* $\mathcal{S}$*.*

Proof. Without loss of generality, let us assume that $P_1, P_2 \ldots P_t$ are under the control of Adv. We claim that adversary learns nothing about $G^{(1)}(x), \ldots, G^{(\ell)}(x)$ beyond t distinct values on these polynomials, different from $x = \beta_1, \ldots, \beta_{\mathsf{pack}}$. As each of these polynomials are of degree at most $\mathsf{d} = t + \mathsf{pack} - 1$, this implies that

Adv learns nothing about the value of these polynomials at $\beta_1, \ldots, \beta_{\mathsf{pack}}$, which are nothing but elements of $\mathcal{S}$. We next proceed to prove our claim.

During Distr, adversary obtains the tuple $(\gamma_i, v_{1,i}, v_{2,i}, \ldots v_{\ell,i}, m_{1,i}, m_{2,i})$ corresponding to each $P_i \in \{P_1, \ldots, P_t\}$ via which it obtains t distinct values of $G^{(1)}(x), \ldots, G^{(\ell)}(x), H^{(1)}(x), H^{(2)}(x)$. During AuthVal, adversary obtains $d, A(x)$. In addition, during RevealPoP, adversary obtains $e, C(x)$. However even after seeing $A(x)$ and $C(x)$, the privacy of $G^{(k)}(\beta_1), \ldots, G^{(k)}(\beta_{\mathsf{pack}})$ is preserved for each $k \in [\ell]$. This is because the polynomials $G^{(1)}(x), \ldots, G^{(\ell)}(x)$ are masked with $H^{(1)}(x)$ and $H^{(2)}(x)$ in the $A(x)$ and $C(x)$ polynomials and adversary will lack pack values of $H^{(1)}(x)$ and $H^{(2)}(x)$ to uniquely interpolate them. More specifically, from the view point of the adversary, for every choice $\overline{\mathcal{S}} = \{(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,\mathsf{pack})}), \ldots, (\overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,\mathsf{pack})})\}$ of the secret values, there exists corresponding secret-encoding polynomials $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x)$ of degree d, with $\overline{G}^{(k)}(\beta_1) = \overline{s}^{(k,1)}, \ldots, \overline{G}^{(k)}(\beta_{\mathsf{pack}}) = \overline{s}^{(k,\mathsf{pack})}$ for each $k \in [\ell]$, such that $\overline{G}^{(k)}(\gamma_i) = v_{k,i}$ holds corresponding to each $P_i \in \{P_1, \ldots, P_t\}$. Moreover corresponding to $\overline{G}^{(1)}(x), \ldots, \overline{G}^{(\ell)}(x)$ and the polynomials $A(x), C(x)$, there exists a corresponding masking set of values $\overline{\mathcal{M}} = \{(\overline{m}^{(1,1)}, \ldots, \overline{m}^{(1,\mathsf{pack})}), (\overline{m}^{(2,1)}, \ldots, \overline{m}^{(2,\mathsf{pack})})\}$ and corresponding masking polynomials $\overline{H}^{(1)}(x)$ and $\overline{H}^{(2)}(x)$ each of degree at most d, such that $A(x) = d\overline{H}^{(1)}(x) + d^2\overline{H}^{(2)}(x) + d^3\overline{G}^{(1)}(x) + d^4\overline{G}^{(2)}(x) + \ldots + d^{\ell+2}\overline{G}^{(\ell)}(x)$ and $C(x) = e\overline{H}^{(1)}(x) + e^2\overline{H}^{(2)}(x) + e^3\overline{G}^{(1)}(x) + e^4\overline{G}^{(2)}(x) + \ldots + e^{\ell+2}\overline{G}^{(\ell)}(x)$ holds, where $\overline{H}^{(1)}(\beta_1) = \overline{m}^{(1,1)}, \ldots, \overline{H}^{(1)}(\beta_{\mathsf{pack}}) = \overline{m}^{(1,\mathsf{pack})}$ and $\overline{H}^{(2)}(\beta_1) = \overline{m}^{(2,1)}, \ldots, \overline{H}^{(2)}(\beta_{\mathsf{pack}}) = \overline{m}^{(2,\mathsf{pack})}$ with $\overline{H}^{(1)}(\gamma_i) = m_{1,i}$, $\overline{H}^{(2)}(\gamma_i) = m_{2,i}$ holding for each $P_i \in \{P_1, \ldots, P_{t+1}\}$. □

Proof of Theorem 3.1 The properties of ICPoP follow from Lemma B.1-B.4. We next prove the communication complexity. During Distr, D sends $\ell + 2$ polynomials of degree d to INT and a tuple of $\ell + 3$ values to each individual verifier. During AuthVal a polynomial of degree d is broadcasted by INT and D broadcasts either an `OK` or `Abort` message. During RevealPoP, INT broadcasts a polynomial of degree d and each individual verifier broadcasts either an `Accept` or a `Reject` message. So overall the protocol has communication complexity $\mathcal{PC}(\mathcal{O}(n\ell))$ and $\mathcal{BC}(\mathcal{O}(n))$, as $\mathsf{d} = \mathcal{O}(n)$. This also proves the ICPoP-**Succinctness of the Proof** property, as the size of the proof is independent of ℓ.

C Properties of Sh-Single and Sh

We analyse the properties of Sh-Single.

Lemma C.1. *If* D *is honest then except with probability at most* $\frac{n^3t}{|\mathbb{F}|-1}$*, it is not discarded during* Sh-Single.

Proof. If D is honest then no honest P_i broadcasts $(\texttt{Abort}, \star)$ message as the received row polynomials will be of degree at most t. More specifically, $f_i(x) = \overline{f}_i(x) = F(x, \alpha_i), m_i^{(1)}(x) = \overline{m}_i^{(1)}(x) = M^{(1)}(x, \alpha_i)$ and $m_i^{(2)}(x) = \overline{m}_i^{(2)}(x) = M^{(2)}(x, \alpha_i)$ holds for P_i. So there can be at most t $(\texttt{Abort}, \star)$ messages corresponding to t potentially corrupted parties. Since D will distribute consistent row polynomials to all the parties, it follows from Lemma 4.1 and protocol steps of Poly-Check that all honest parties will be present in $\mathcal{W}^{(P_1)}, \ldots, \mathcal{W}^{(P_n)}$ and so clearly $|\mathsf{VCORE}| \geq 2t + 1$ will hold. Now consider a pair of parties P_i, P_j, with at least one of them being *corrupted*, such that in the RevealPoP$_{ji}$ instance the revealed proof does not correspond to $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$[8]. It follows via Lemma B.3 (by substituting $\mathsf{pack} = 1$ and $\mathsf{d} = t + \mathsf{pack} - 1 = t$) that except with probability at most $\frac{nt}{|\mathbb{F}|-1}$, the proof will be rejected. As there can be at most n^2 such pairs of (P_i, P_j), from the union bound it follows that except with probability at most $\frac{n^3t}{|\mathbb{F}|-1}$, the values which are finally considered for reconstructing the column polynomials for the parties will be correct and lie on polynomials of degree at most t. So except with probability at most $\frac{n^3t}{|\mathbb{F}|-1}$, the conditions which lead to an honest D being discarded never occur. □

[8]This may happen if a corrupted P_i transfers incorrect values to an honest P_j or if a corrupted P_j purposely tries to reveal a proof corresponding to an incorrect set of values.

Lemma C.2 (Correctness for an honest D). *If* D *is honest then except with probability at most* $\frac{n^3t}{|\mathbb{F}|-1}$*, the value* s *is* t*-shared at the end of* Sh-Single.

Proof. If D is honest then from Lemma C.1 it follows that except with probability at most $\frac{n^3t}{|\mathbb{F}|-1}$, any incorrect linear combination of values revealed in any of the RevealPoP instances is rejected. More specifically, if P_j is *honest* and $P_i \in \sup_j$, then the linear combination comb_{ji} revealed by P_j in the instance $\mathsf{RevealPoP}_{ji}$ will be correct and correspond to the values in $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$. This further implies that P_i transferred the correct $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$ to P_j. Thus the values used by an honest P_j to determine its column polynomial are correct (lying on $g_j(y) = F(\alpha_j, y)$). So $\overline{g}_j(y) = g_j(y)$ holds for each honest P_j, implying that s is t-shared via the polynomial $f_0(x) \stackrel{\text{def}}{=} F(x, 0)$, with P_j holding the share $f_0(j) = g_j(0)$. □

Lemma C.3. *Let* $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ *and* $\overline{m}_i^{(2)}(x)$ *be the row polynomials defined by the values in* $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ *received by party* $P_i \in \mathcal{P}$ *from* D *for* $j \in [n]$*. If* D *is corrupted and a* VCORE *is formed during* Sh-Single *then except with probability at most* $\frac{3n^2}{|\mathbb{F}|}$*, there exist bivariate polynomials, say* $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ *and* $\overline{M}^{(2)}(x, y)$*, each of degree at most* t*, such that for each honest* $P_i \in$ VCORE*, the polynomials* $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ *and* $\overline{m}_i^{(2)}(x)$ *lie on* $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ *and* $\overline{M}^{(2)}(x, y)$ *respectively.*

Proof. From the definition, $\mathsf{VCORE} = \mathcal{W}^{(P_1)} \cap \mathcal{W}^{(P_2)} \cap \ldots \cap \mathcal{W}^{(P_n)}$ and $|\mathsf{VCORE}| \geq 2t + 1$. This ensures that there are at least $t + 1$ common honest parties in VCORE, say HVCORE. Consider an honest party $P_j \in \mathcal{P}$, playing the role of the verifier V in the instance $\mathsf{Poly\text{-}Check}^{(P_j)}$. It follows from Lemma 4.1 (by substituting $L = 3$) that for the instance $\mathsf{Poly\text{-}Check}^{(P_j)}$, except with probability at most $\frac{3n}{|\mathbb{F}|}$, the row polynomials $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ of the parties $P_i \in$ HVCORE together lie on three unique bivariate polynomials, say $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ respectively of degree at most t. The same will be true with respect to every other instance $\mathsf{Poly\text{-}Check}^{(P_k)}$, corresponding to every other honest verifier $P_k \neq P_j$. Moreover, the set of three bivariate polynomials defined via each of these instances of Poly-Check will be the same, namely $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ respectively. This follows from Lemma A.2 (by substituting $\ell = |\mathsf{HVCORE}|$) and the fact that $|\mathsf{HVCORE}| \geq t + 1$. The lemma now follows from the union bound and the fact that there are $\Theta(n)$ honest parties, playing the role of V. □

Lemma C.4 (Correctness for a corrupted D). *If* D *is corrupted and not discarded during* Sh-Single*, then there exists some value, say* $\overline{s}$*, such that except with probability at most* $\frac{n^3}{|\mathbb{F}|-1}$*,* $\overline{s}$ *is* t*-shared at the end of* Sh-Single.

Proof. If a *corrupted* D is not discarded then it implies that a set VCORE with $|\mathsf{VCORE}| \geq 2t+1$ is constructed during Sh-Single. Let HVCORE be the set of honest parties in VCORE; clearly $|\mathsf{HVCORE}| \geq t + 1$. From Lemma C.3 it follows that except with probability at most $\frac{3n^2}{|\mathbb{F}|}$, the row polynomials $\overline{f}_i(x), \overline{m}_i^{(1)}(x)$ and $\overline{m}_i^{(2)}(x)$ of the parties in HVCORE lie on unique bivariate polynomials, say $\overline{F}(x, y), \overline{M}^{(1)}(x, y)$ and $\overline{M}^{(2)}(x, y)$ of degree at most t. We define $\overline{s} \stackrel{\text{def}}{=} \overline{F}(0, 0)$ and claim that $\overline{s}$ is t-shared via the polynomial $\overline{f}_0(x) \stackrel{\text{def}}{=} \overline{F}(x, 0)$, with each honest P_j holding the share $\overline{s}_j \stackrel{\text{def}}{=} \overline{F}(\alpha_j, 0)$. To prove our claim, we show that each *honest* party P_j outputs its degree t univariate polynomial $\overline{g}_j(y) \stackrel{\text{def}}{=} \overline{F}(\alpha_j, y)$ except with probability at most $\frac{n^2}{|\mathbb{F}|-1}$; this ensures that P_j obtains the correct share, as $\overline{s}_j = \overline{g}_j(0)$. For this, we further need to show that the $\overline{\mathcal{S}}_{ij}$ set transferred by each party $P_i \in \sup_j$ to P_j contains the value $\overline{g}_j(\alpha_i)$.

Consider an honest P_j. Notice that $\sup_j \subseteq$ VCORE. We first argue that every $P_i \in$ HVCORE is present in $\sup_j$, except with probability at most $\frac{n^2}{|\mathbb{F}|-1}$. This is because there are $\Theta(n)$ such parties P_i and in each corresponding $\mathsf{RevealPoP}_{ji}$ instance, the output is `AcceptProof`, which follows from Lemma B.2 (by substituting $\ell = 1$). Now consider the set of values $\overline{\mathcal{S}}_{ij} = \{\overline{f}_{ij}\}$ and $\overline{\mathcal{M}}_{ij} = \{\overline{m}_{ij}^{(1)}, \overline{m}_{ij}^{(2)}\}$ transferred by the parties

$P_i \in$ HVCORE to P_j. Since $\overline{f}_{ij} = \overline{f}_i(\alpha_j) = \overline{g}_j(\alpha_i)$ holds, it follows that the values $\{\overline{f}_{ij}\}_{P_i \in \mathsf{HVCORE}}$ define the degree t univariate polynomial $\overline{g}_j(y)$. Similarly the values $\{\overline{m}_{ij}^{(1)}\}_{P_i \in \mathsf{HVCORE}}$ and $\{\overline{m}_{ij}^{(2)}\}_{P_i \in \mathsf{HVCORE}}$ define degree t univariate polynomials $\overline{M}^{(1)}(y, \alpha_j)$ and $\overline{M}^{(2)}(y, \alpha_j)$ respectively. To complete the proof, we argue that except with probability at most $\frac{2}{|\mathbb{F}|}$, the values in the $\overline{\mathcal{S}}_{ij}$ and $\overline{\mathcal{M}}_{ij}$ set transferred by a *corrupted* party $P_i \in \sup_j$ lie on $\overline{g}_j(y), \overline{M}^{(1)}(y, \alpha_j)$ and $\overline{M}^{(2)}(y, \alpha_j)$ respectively. This is because the combiner e_j selected by the honest P_j in the $\mathsf{RevealPoP}_{ji}$ instances corresponding to the parties in $\sup_j$ is truly random and unknown to the adversary in advance, when the $\overline{\mathcal{S}}_{ij}$ and $\overline{\mathcal{M}}_{ij}$ sets are transferred to P_j. The rest follows from Lemma A.4 (by substituting $L = 2$) and the fact that the values $\{\mathsf{comb}_{ji}\}_{P_i \in \sup_j}$ lie on a polynomial of degree at most t (otherwise D would have been discarded), say $\mathsf{comb}_j(y)$, where $\mathsf{comb}_j(y) \stackrel{\text{def}}{=} e_j \overline{M}^{(1)}(y, \alpha_j) + e_j^2 \overline{M}^{(2)}(y, \alpha_j) + e_j^3 \overline{g}_j(y)$. As there can be n^2 pair of parties involving a corrupted party, it follows by the union bound that except with probability at most $\frac{2n^2}{|\mathbb{F}|}$, the corrupted parties in VCORE transfer the correct values to the honest parties.

As each honest P_j correctly obtains its column polynomial except with probability at most $\frac{n^2}{|\mathbb{F}|-1}$ and as there are $\Theta(n)$ such honest parties, it follows that except with probability at most $\frac{n^3}{|\mathbb{F}|-1}$, the value $\overline{s}$ is t-shared. □

Lemma C.5 (Privacy). *In protocol* Sh-Single*, the value s remains information theoretically secure.*

Proof. For privacy, we have to consider an honest D. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of Adv. We argue that throughout Sh-Single, the adversary learns nothing about $F(x, y)$, beyond the row polynomials $f_1(x), \ldots, f_t(x)$ and the column polynomials $g_1(y), \ldots, g_t(y)$. Through these polynomials, the adversary learns $t^2 + 2t$ distinct values of $F(x, y)$. As the degree of $F(x, y)$ is t, the adversary lacks one additional value on $F(x, y)$ to uniquely interpolate $F(x, y)$, implying information-theoretic security for s.

Through the instances Distr_{ij} where $i \in [t]$ and $j \in [n]$, the adversary Adv learns the row polynomials $f_1(x), \ldots, f_t(x), m_1^{(1)}(x), \ldots, m_t^{(1)}(x), m_1^{(2)}(x), \ldots, m_t^{(2)}(x)$ on the bivariate polynomials $F(x, y), M^{(1)}(x, y)$ and $M^{(2)}(x, y)$ respectively. From Lemma 4.1, during $\mathsf{Poly\text{-}Check}^{(P_1)}, \ldots, \mathsf{Poly\text{-}Check}^{(P_n)}$, no additional information about $F(x, y), M^{(1)}(x, y)$ and $M^{(2)}(x, y)$ is revealed to the adversary, because in each instance $\mathsf{Poly\text{-}Check}^{(P_i)}$, a random blinding univariate polynomial $B^{(P_i)}(y)$ is used. Now consider a pair of *honest* parties $P_i, P_j \in \mathcal{P}$. In the protocol, party P_i executes an instance $\mathsf{AuthVal}_{ij}$ involving $\mathcal{S}_{ij} = \{f_i(\alpha_j)\}$ and $\mathcal{M}_{ij} = \{m_i^{(1)}(\alpha_j), m_i^{(2)}(\alpha_j)\}$. Moreover, the set $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$ is privately transferred to P_j by P_i and later on during Round 4 and 5, an instance $\mathsf{RevealPoP}_{ji}$ is instantiated again involving $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$. We claim that during $\mathsf{AuthVal}_{ij}$ and $\mathsf{RevealPoP}_{ji}$, the privacy of $\mathcal{S}_{ij}$ is preserved. This follows from the privacy property of ICPoP (Lemma B.4) and the fact that the corresponding masking set $\mathcal{M}_{ij}$ used in these instances are private. Thus for every pair of honest parties P_i, P_j, no additional information about the $f_i(\alpha_j)$ values (which are the same as the $g_j(\alpha_i)$ values) are revealed during the instances $\mathsf{AuthVal}_{ij}$ and $\mathsf{RevealPoP}_{ji}$. The adversary is able to compute the column polynomials $g_1(y), \ldots, g_t(y)$ through the common values on these column polynomials which are transferred to $P_1, \ldots, P_t$ by the honest parties. Hence throughout the protocol, the adversary learns t row and column polynomials, proving the privacy. □

Proof of Theorem 4.2: The properties of VSS follow from Lemma C.2-C.5. In the protocol n^2 instances of ICPoP (with $\ell = 1, \mathsf{pack} = 1$) and n instances of Poly-Check (each with $L = 3$) are executed. The rest follows from the communication complexity of ICPoP (Theorem 3.1) and Poly-Check (Lemma 4.1).

C.1 Four Round Statistical VSS with a Quadratic Overhead

We now discuss the modifications to be made to Sh-Single to get a four round VSS protocol Sh, which allows D to t-share $\ell \times (n - t) = \Theta(n\ell)$ secrets with communication complexity $\mathcal{PC}(\mathcal{O}(n^3\ell))$ and $\mathcal{BC}(\mathcal{O}(n^3))$. For simplicity, we first discuss how to t-share $n - t = \Theta(n)$ secrets with communication complexity $\mathcal{PC}(\mathcal{O}(n^3))$ and $\mathcal{BC}(\mathcal{O}(n^3))$. The modifications to share $\ell \times (n - t)$ secrets follow in a straight forward fashion.

Sharing $n-t$ Secrets: The idea behind efficiently sharing $n-t$ secrets is to invoke the underlying instances of Distr, AuthVal and RevealPoP in Sh-Single with the maximum possible value of pack, which is $n-t$ (for the moment we will restrict to $\ell = 1$). The rest of the protocol steps remain the same, with a slight modification in the steps for consistency checking of the values transferred by the parties in VCORE. More specifically, let $\vec{S} = (s^{(1)}, \ldots, s^{(n-t)})$ be the set of values, which need to be t-shared. To do so D selects $n-t$ random degree t secret-carrying bivariate polynomials $F^{(1)}(x,y), \ldots, F^{(n-t)}(x,y)$, embedding the secrets $s^{(1)}, \ldots, s^{(n-t)}$ respectively in their constant terms. In addition, D picks $2(n-t)$ random masking bivariate polynomials $M^{(1,1)}(x,y), \ldots, M^{(1,n-t)}(x,y), M^{(2,1)}(x,y), \ldots, M^{(2,n-t)}(x,y)$ polynomials. The reason for picking so many masking polynomials will be clear in the sequel. Let $f_i^{(1)}(x), \ldots, f_i^{(n-t)}(x)$ and $g^{(1)}(y), \ldots, g^{(n-t)}(y)$ denote the ith row and column polynomials of $F^{(1)}(x,y), \ldots, F^{(n-t)}(x,y)$ respectively. Similarly, let $m_i^{(1,1)}(x), \ldots, m_i^{(1,n-t)}(x)$, $m_i^{(2,1)}(x), \ldots, m_i^{(2,n-t)}(x)$ denote the ith row polynomials of the masking bivariate polynomials. Corresponding to each party P_i, the dealer D sets $\mathcal{S}_{ij} = \{f_i^{(1)}(\alpha_j), \ldots, f_i^{(n-t)}(\alpha_j)\}$ and $\mathcal{M}_{ij} = \{(m_i^{(1,1)}(\alpha_j), \ldots, m_i^{(1,n-t)}(\alpha_j)), (m_i^{(2,1)}(\alpha_j), \ldots, m_i^{(2,n-t)}(\alpha_j))\}$. An instance Distr_{ij} is executed, considering P_i as an INT to give $\mathcal{S}_{ij} \cup \mathcal{M}_{ij}$ to P_i, for $j = 1, \ldots, n$. The instances of Distr are executed by setting $\ell = 1$ and $\mathsf{pack} = n-t$ (hence d will be $n-1$ in these instances).

Let $\overline{f}_i^{(1)}(x), \ldots, \overline{f}_i^{(n-t)}(x), \overline{m}_i^{(1,1)}(x), \ldots, \overline{m}_i^{(1,n-t)}(x), \overline{m}_i^{(2,1)}(x), \ldots, \overline{m}_i^{(2,n-t)}(x)$ denote the row polynomials received by P_i via the instances Distr_{ij}. The parties check for the existence of VCORE as in Sh-Single by executing n instances of Poly-Check, where P_i plays the role of the designated verifier in the ith instance. For each instance, one independent blinding polynomial will be used, which will be shared by D during the first round. If a VCORE is obtained, then it implies that the row polynomials of the honest parties P_i in VCORE lie on $n-t$ secret-carrying bivariate polynomials of degree t, say $\overline{F}^{(1)}(x,y), \ldots, \overline{F}^{(n-t)}(x,y)$ and $2(n-t)$ masking bivariate polynomials, say $\overline{M}^{(1,1)}(x,y), \ldots, \overline{M}^{(1,n-t)}(x,y), \overline{M}^{(2,1)}(x,y), \ldots, \overline{M}^{(2,n-t)}(x,y)$ respectively We define $(\overline{F}^{(1)}(0,0), \ldots, \overline{F}^{(n-t)}(0,0))$ to be the $n-t$ secrets "committed" by D (if D is honest then these will be the same as $\vec{S}$) and proceed to complete t-sharing of these values by ensuring that each P_j gets its degree t column polynomials $\overline{F}^{(1)}(\alpha_j, y), \ldots, \overline{F}^{(n-t)}(\alpha_j, y)$ and outputs their constant terms as its shares. This is done as follows.

Let $\overline{\mathcal{S}}_{ij} = \{\overline{f}_i^{(1)}(\alpha_j), \ldots, \overline{f}_i^{(n-t)}(\alpha_j)\}$ and $\overline{\mathcal{M}}_{ij} = \{(\overline{m}_i^{(1,1)}(\alpha_j), \ldots, \overline{m}_i^{(1,n-t)}(\alpha_j)), (\overline{m}_i^{(2,1)}(\alpha_j), \ldots, \overline{m}_i^{(2,n-t)}(\alpha_j))\}$ denote the sets received by P_i at the end of Distr_{ij}. By the properties of VCORE, each honest $P_i \in$ VCORE will be able to give a proof of possession of $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$, as the corresponding $\mathsf{AuthVal}_{ij}$ instance would not be aborted by D. Hence if P_i transfers these sets to P_j, then even P_j can give a proof of possession of these sets. So Each P_i (in VCORE) [9] sends the set $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ to P_j, who then publicly verifies these values by executing an instance $\mathsf{RevealPoP}_{ji}$ of RevealPoP and giving a proof of possession of these sets of values. Party P_j ensures that the *same* randomness e_j is used in all the $\mathsf{RevealPoP}_{ji}$ instances. Let sup_j denote the set of parties P_i from VCORE, such that in the corresponding $\mathsf{RevealPoP}_{ji}$ instance the output is `AcceptProof`, along with a set of $n-t$ linearly combined values, say $(\mathsf{comb}_{ji}^{(1)}, \ldots, \mathsf{comb}_{ji}^{(n-t)})$ (recall that now the instances of RevealPoP are executed with $\mathsf{pack} = n-t$ and so $n-t$ linearly combined values will be produced in these instances). If D is *honest* then with high probability, only the parties sending the correct $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ sets will be present in sup_j. However if D is *corrupted* then a corrupted P_i can send incorrect sets and still be present in sup_j. To check this, it is publicly verified if the sets of values $\{(\alpha_i, \mathsf{comb}_{ji}^{(1)})\}_{P_i \in \mathrm{sup}_j}, \ldots, \{(\alpha_i, \mathsf{comb}_{ji}^{(n-t)})\}_{P_i \in \mathrm{sup}_j}$ lie on $n-t$ univariate polynomials of degree at most t. If so then it ensures that with high probability, the parties in sup_j sent the correct sets to P_j. This is because the values in $\overline{\mathcal{S}}_{ij} \cup \overline{M}_{ij}$ corresponding to the *honest* parties in sup_j clearly define degree t column polynomials $\overline{F}^{(1)}(\alpha_j, y), \ldots, \overline{F}^{(n-t)}(\alpha_j, y), \overline{M}^{(1,1)}(\alpha_j, y), \ldots, \overline{M}^{(1,n-t)}(\alpha_j, y)$, $\overline{M}^{(2,1)}(\alpha_j, y), \ldots, \overline{M}^{(2,n-t)}(\alpha_j, y)$. Since P_j uses the same combiner e_j to produce the linear combination of the values in $\overline{\mathcal{S}}_{ij} \cup \overline{M}_{ij}$ in all the $\mathsf{RevealPoP}_{ji}$ instances, it follows that the linear combinations $\mathsf{comb}_{ji}^{(1)}, \ldots, \mathsf{comb}_{ji}^{(n-t)}$ of these $\overline{\mathcal{S}}_{ij} \cup \overline{M}_{ij}$ sets also lie on a degree t univariate polynomial; specifically the

[9] Even though each P_i sends the corresponding $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ to P_j, party P_j will focus only on the P_is in VCORE

set of values $\{(\alpha_i, \mathsf{comb}_{ji}^{(k)})\}$ corresponding to the *honest* parties P_i in sup_j will define a degree t univariate polynomial $e_j\overline{M}^{(1,k)}(\alpha_j, y) + e_j^2\overline{M}^{(2,k)}(\alpha_j, y) + e_j^3\overline{F}^{(k)}(\alpha_j, y)$ for $k = 1, \ldots, n-t$. Now if a *corrupted* P_i in sup_j sent an incorrect set to P_j, then with high probability, the corresponding $\mathsf{comb}_{ji}^{(k)}$ values will not lie on the degree t univariate polynomial $e_j\overline{M}^{(1,k)}(\alpha_j, y) + e_j^2\overline{M}^{(2,k)}(\alpha_j, y) + e_j^3\overline{F}^{(k)}(\alpha_j, y)$, in which case D will be discarded. For the ease of understanding, a pictorial representation of the values distributed during Sh to share $n-t$ secrets is shown in Fig. 7.

Sharing $\ell \times (n-t)$ Secrets Simultaneously: The principle behind sharing $\ell \times (n-t)$ secrets $\vec{S} = (s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ will be similar to that of sharing $n-t$ secrets as discussed above. The only difference will be that the $\mathcal{S}_{ij}$ sets in the underlying Distr_{ij}, $\mathsf{AuthVal}$ and $\mathsf{RevealPoP}_{ji}$ instances will be of size $\ell \times (n-t)$, instead of $n-t$; the $\mathcal{M}_{ij}$ sets will remain the same as above. More specifically, D will now select $\ell \times (n-t)$ secret-carrying random bivariate polynomials of degree t, say $F^{(l,k)}(x, y)$ for $l \in [\ell]$ and $k \in [n-t]$, each embedding a secret from $\vec{S}$ in its constant term; the number of masking polynomials remain $2(n-t)$. Now the $\{(\alpha_i, \mathsf{comb}_{ji}^{(k)})\}$ values corresponding to the *honest* parties P_i in sup_j will define a linear combination of $\ell+2$ column polynomials $M^{(1,k)}(\alpha_j, y), M^{(2,k)}(\alpha_j, y), F^{(1,k)}(\alpha_j, y), \ldots, F^{(\ell,k)}(\alpha_j, y)$ for $k \in [n-t]$. The rest of the protocol steps remain the same as above. To avoid repetition, we do not present the complete formal steps of Sh and the detailed proof of its properties. Instead we state the formal properties of Sh which follow in a straight forward fashion from the corresponding properties of $\mathsf{Sh\text{-}Single}$, taking into account that the underlying instances of ICPoP that are executed deal with $\ell \times (n-t)$ values.

Lemma C.6. *If D is honest then except with probability at most $\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}$, it is not discarded during Sh.*

Proof. Similar to Lemma C.1, except that now each instance of ICPoP satisfies the ICPoP-**Correctness3** property except with probability at most $\frac{n\mathsf{d}}{|\mathbb{F}|-\mathsf{pack}}$, where $\mathsf{pack} = n-t$ and $\mathsf{d} = t + \mathsf{pack} - 1 = n-1$. This ensures that if a corrupted $P_i \in \mathsf{VCORE}$ transfers incorrect values to an honest P_j, then it is caught in the corresponding $\mathsf{RevealPoP}_{ji}$ instance. And there are n^2 such instances, involving a corrupted P_i and an honest P_j. □

Lemma C.7 (Correctness for an honest D). *If D is honest then except with probability at most $\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}$, the $\ell \times (n-t)$ values $(s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ are t-shared at the end of Sh.*

Proof. Similar to Lemma C.2, except that now we rely on Lemma C.6. □

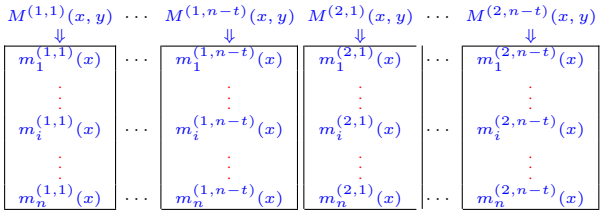
Lemma C.8. *Let $\overline{f}_i^{(1,1)}(x), \ldots, \overline{f}_i^{(1,n-t)}(x), \ldots, \overline{f}_i^{(\ell,1)}(x), \ldots, \overline{f}_i^{(\ell,n-t)}(x), \overline{m}_i^{(1,1)}(x), \ldots, \overline{m}_i^{(1,n-t)}(x)$ and $\overline{m}_i^{(2,1)}(x), \ldots, \overline{m}_i^{(2,n-t)}(x)$ be the row polynomials defined by the values in $\overline{\mathcal{S}}_{ij} \cup \overline{\mathcal{M}}_{ij}$ received by party $P_i \in \mathcal{P}$ from D for $j \in [n]$. If D is corrupted and a VCORE is formed during Sh then except with probability at most $\frac{n^2(\ell+2)(n-t)}{|\mathbb{F}|}$, there exist $(\ell+2)(n-t)$ bivariate polynomials, say $\overline{F}^{(1,1)}(x, y), \ldots, \overline{F}^{(1,n-t)}(x, y), \ldots, \overline{F}^{(\ell,1)}(x, y), \ldots, \overline{F}^{(\ell,n-t)}(x, y), \overline{M}^{(1,1)}(x, y), \ldots, \overline{M}^{(1,n-t)}(x, y), \overline{M}^{(2,1)}(x, y), \ldots, \overline{M}^{(2,n-t)}(x, y)$, each of degree at most t, such that for each honest $P_i \in \mathsf{VCORE}$, the polynomial $\overline{f}_i^{(l,k)}(x)$ lie on $\overline{F}^{(l,k)}(x, y)$ for $l \in [\ell], k \in [n-t]$, the polynomial $\overline{m}_i^{(1,k)}(x)$ lie on $\overline{M}^{(1,k)}(x, y)$ for $k \in [n-t]$ and the polynomial $\overline{m}_i^{(2,k)}(x)$ lie on $\overline{M}^{(2,k)}(x, y)$ for $k \in [n-t]$.*

Proof. Similar to Lemma C.3, except that now we rely on Lemma 4.1 with $L = (\ell+2)(n-t)$. □

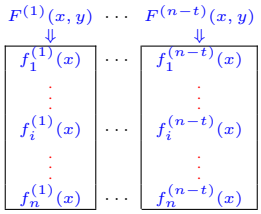
Lemma C.9 (Correctness for a corrupted D). *If D is corrupted and not discarded during $\mathsf{Sh\text{-}Single}$, then there exists $\ell \times (n-t)$ values, say $(\overline{s}^{(1,1)}, \ldots, \overline{s}^{(1,n-t)}, \ldots, \overline{s}^{(\ell,1)}, \ldots, \overline{s}^{(\ell,n-t)})$, such that then except with probability at most $\frac{n^3\ell}{|\mathbb{F}|-1}$, the values $\overline{s}^{(l,k)}$ are t-shared at the end of Sh for $l \in [\ell]$ and $k \in [n-t]$.*

Figure 7: Pictorial representation of the values distributed in Sh protocol that shares $n-t$ secrets.

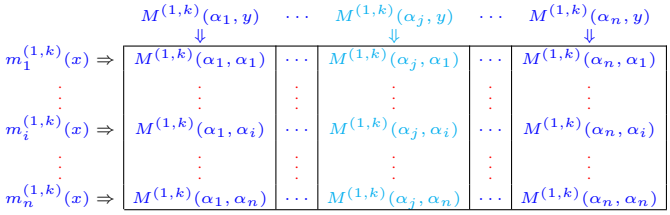
(a) Polynomials $M^{(1,1)}(x,y), \cdots M^{(1,n-t)}(x,y)$ and $M^{(2,1)}(x,y), \cdots M^{(2,n-t)}(x,y)$.



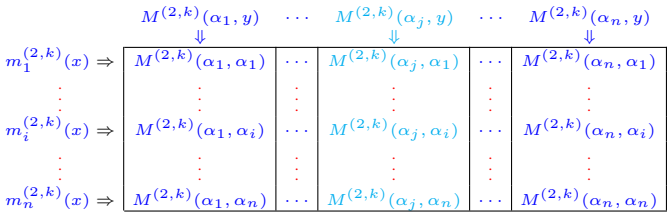
(b) Polynomial $F^{(k)}(x,y)$ where $k \in [n-t]$.



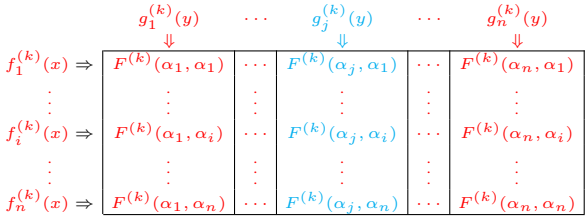
(c) Closer look at $M^{(1,k)}(x,y)$ with party P_i holding the ith row.



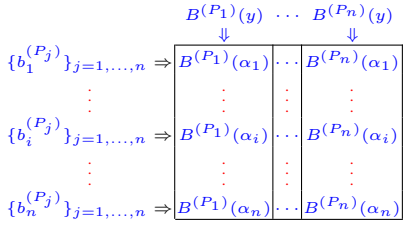
(d) Closer look at $M^{(2,k)}(x,y)\ with P_i$ holding the ith row.



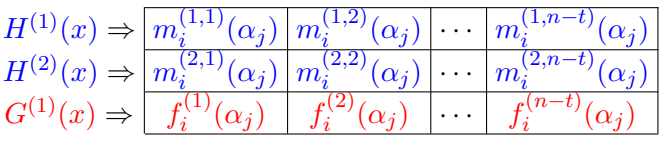
(e) Closer look at $F^{(k)}(x,y)$ with party P_i holding the ith row.



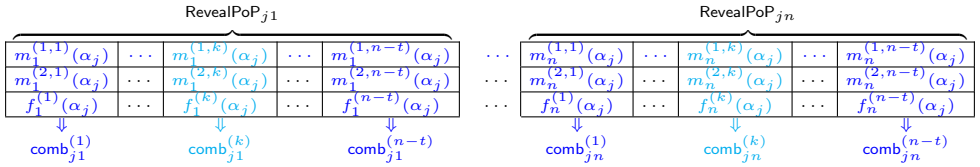
(f) Blinding polynomials with party P_i holding the ith row.



(g) $\mathsf{Distr}_{ij} = \mathsf{Distr}(\mathsf{D}, P_i, \mathcal{P}, 1, (n-t), \mathcal{S}_{ij} \cup \mathcal{M}_{ij})$ where $\mathcal{S}_{ij} = \{f_i^{(1)}(\alpha_j), \ldots, f_i^{(n-t)}(\alpha_j)\}$ and $\mathcal{M}_{ij} = \{(m_i^{(1,1)}(\alpha_j), \ldots, m_i^{(1,n-t)}(\alpha_j)), (m_i^{(2,1)}(\alpha_j), \ldots, m_i^{(2,n-t)}(\alpha_j))\}$ for $i, j \in [n]$.This is similar to the figure 6f with $\mathsf{pack} = (n-t)$.



(h) $\mathsf{RevealPoP}_{ji}$ instances executed by Party P_j corresponding to the parties $P_i \in \mathsf{VCORE}$. The same random combiner e_j is used in all these instances. $\mathsf{comb}_{ji}^{(k)}$ denotes the linear combination of values revealed in these instances for $k \in [n-t]$. This is analogous to figure 6g with $\ell = 1, \mathsf{pack} = n - t$.



$$\mathsf{comb}_{ji}^{(k)} = e_j m_1^{(1,k)}(\alpha_j) + e_j^2 m_1^{(2,k)}(\alpha_j) + e_j^3 f_1^{(k)}(\alpha_j)$$
$$\mathsf{comb}_{j2}^{(k)} = e_j m_2^{(1,k)}(\alpha_j) + e_j^2 m_2^{(2,k)}(\alpha_j) + e_j^3 f_2^{(k)}(\alpha_j)$$
$$\cdots$$
$$\mathsf{comb}_{jn}^{(k)} = e_j m_n^{(1,k)}(\alpha_j) + e_j^2 m_n^{(2,k)}(\alpha_j) + e_j^3 f_n^{(k)}(\alpha_j)$$

Note from figure 7c that $\{m_1^{(1,k)}(\alpha_j), m_2^{(1,k)}(\alpha_j) \cdots m_n^{(1,k)}(\alpha_j)\}$ define $M^{(1,k)}(\alpha_j, y)$. From figure 7d, $\{m_1^{(2,k)}(\alpha_j), \cdots m_n^{(2,k)}(\alpha_j)\}$ define $M^{(2,k)}(\alpha_j, y)$. Also from figure 7e, $\{f_1^{(k)}(\alpha_j), f_2^{(k)}(\alpha_j) \cdots f_n^{(k)}(\alpha_j)\}$ define $F^{(k)}(\alpha_j, y)$ where $k \in [n-t]$. Hence, the combination i.e $e_j M^{(1.k)}(\alpha_j, y) + e_j^2 M^{(2,k)}(\alpha_j, y) + e_j^3 F^{(k)}(\alpha_j, y)$ is a univariate t-degree polynomial defined by the $\mathsf{comb}_{ji}^{(k)}$ values

Proof. Similar to Lemma C.4, except that we now use Lemma C.8. Moreover, for every pair of *honest* parties (P_i, P_j), where $P_i \in \mathsf{VCORE}$, it is ensured that except with probability at most $\frac{n\ell}{|\mathbb{F}|-1}$, party P_i is present in sup_j; this follows from Lemma B.2. As there are $\Theta(n^2)$ such pairs, from the union bound it is ensured that except with probability at most $\frac{n^3\ell}{|\mathbb{F}|-1}$, every honest party from VCORE is present in the sup_j set of every honest P_j. Furthermore it is ensured that except with probability at most $\frac{(\ell+1)}{|\mathbb{F}|}$, no *corrupted* party $P_i \in \mathsf{VCORE}$ is present in sup_j set of an *honest* P_j; this follows from Lemma A.4 (by substituting $L = \ell + 1$). As there can be $\mathcal{O}(n^2)$ pairs of parties, from the union bound it follows that except with probability at most $\frac{n^2(\ell+1)}{|\mathbb{F}|}$, the values transferred by the corrupted parties in VCORE to the honest parties are correct. So overall the error probability is at most $\frac{n^3\ell}{|\mathbb{F}|-1}$. □

Lemma C.10 (Privacy). *In protocol* Sh*, the values* $(s^{(1,1)}, \ldots, s^{(1,n-t)}, \ldots, s^{(\ell,1)}, \ldots, s^{(\ell,n-t)})$ *remain information-theoretically secure.*

Proof. Similar to the proof of Lemma C.5. □

Proof of Theorem 4.3: The properties of VSS follow from Lemma C.7-C.10. In the protocol n^2 instances of ICPoP (with $\mathsf{pack} = n - t$) and n instances of $\mathsf{Poly\text{-}Check}$ (each with $L = (\ell + 2)(n - t)$) are executed. The rest follows from the communication complexity of ICPoP (Theorem 3.1) and $\mathsf{Poly\text{-}Check}$ (Lemma 4.1).

D Partially Synchronous Statistical MPC Protocol

D.1 Existing Asynchronous Primitives

Reconstruction of a t-shared Value by a Designated Party [11, 4, 36, 14]: Let v be a value, t-shared through a polynomial $p(\cdot)$. The well-know *online error correction* (OEC) algorithm allows some designated party $P_R \in \mathcal{P}$ to reconstruct $p(\cdot)$ and thus $p(0) = v$, in an asynchronous fashion. We denote the protocol as OEC, whose communication complexity $\mathcal{PC}(\mathcal{O}(n))$; every honest party eventually terminates the protocol.

Multiplication of Pairs of t-shared Values using Beaver's Technique [2]: It securely computes $[x \cdot y]_t$ from $[x]_t$ and $[y]_t$, at the expense of two *public reconstructions*, using a *pre-computed* t-shared random multiplication

triple (from the offline phase), say $([a]_t, [b]_t, [c]_t)$. For this, the parties first (locally) compute $[e]_t$ and $[d]_t$, where $[e]_t \stackrel{\text{def}}{=} [x]_t - [a]_t = [x-a]_t$ and $[d]_t \stackrel{\text{def}}{=} [y]_t - [b]_t = [y-b]_t$, followed by the public reconstruction of $e = (x-a)$ and $d = (y-b)$; to do the public reconstruction $2n$ instances of OEC are executed, two on the behalf of each party. Since $xy = ((x-a)+a)((y-b)+b) = de + eb + da + c$ holds, the parties can locally compute $[xy]_t = de + e[b]_t + d[a]_t + [c]_t$, once d and e are publicly known. The above computation leaks no information about x and y if a and b are random and unknown to Adv. We call the protocol as Beaver, whose communication complexity is $\mathcal{PC}(\mathcal{O}(n^2))$; the protocol eventually terminates for every honest party.

The Asynchronous Triple Transformation Protocol [15]: The heart of the efficient framework of [15] is the *asynchronous* triple transformation protocol TripTrans. The protocol takes as input a set of $(3t+1)$ independent t-shared triples, say $\{([x^{(i)}]_t, [y^{(i)}]_t, [z^{(i)}]_t\}_{i\in[3t+1]}$ and outputs a set of $(3t+1)$ "co-related" t-shared triples, say $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)})\}_{i\in[3t+1]}$, such that the following hold:

- There exist polynomials, say $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ of degree $\frac{3t}{2}$, $\frac{3t}{2}$ and $3t$ respectively, such that $X(\alpha_i) = \mathbf{x}^{(i)}, Y(\alpha_i) = \mathbf{y}^{(i)}$ and $Z(\alpha_i) = \mathbf{z}^{(i)}$ holds for $i \in [3t+1]$.
- $Z(\cdot) = X(\cdot)Y(\cdot)$ holds if and only if all the input triples are multiplication triples. This further implies that $Z(\cdot) = X(\cdot)Y(\cdot)$ is true if and only if all the $(3t+1)$ input triples are multiplication triples.
- If Adv knows $t' < \frac{3t}{2}$ input triples then Adv learns t' values on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, implying $\frac{3t}{2} + 1 - t'$ "degree of freedom" on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$. If $t' > \frac{3t}{2}$, then Adv completely learns $X(\cdot), Y(\cdot)$ and $Z(\cdot)$.

The protocol involves $\frac{3t}{2}$ instances of Beaver and has communication complexity $\mathcal{PC}(\mathcal{O}(n^3))$. The protocol can be executed in a completely asynchronous fashion and it is ensured that every honest party eventually terminates the protocol. We refer to [15] for the complete formal details of TripTrans.

D.2 The Framework of [15] for the Offline Phase

In [15] an efficient framework for the offline phase for generating t-shared random multiplication triples is presented. On a very high level, the framework consists of the following two modules:

Module I — Multiplication Triple Sharing: This module allows a designated dealer D to *verifiably* t-share multiplication triples. By verifiability, it means that the triples are guaranteed to be multiplication triples. Moreover, the triples remain private if D is honest. To achieve this task, the module takes any polynomial based VSS scheme and plug it with the triple transformation protocol TripTrans. In our context, we will use our VSS protocol Sh. The module is executed as follows.

D invokes our four round VSS protocol Sh to verifiably t-share $\mathfrak{l}(3t+1)$ values. So we require that the first four rounds are synchronous broadcast rounds, which ensures that at the end of the fourth round, $\mathfrak{l}(3t+1)$ values are shared by D. After this, the rest of the steps are executed in a completely asynchronous fashion[10]. The values shared by D can be viewed as $\mathfrak{l}$ batches of $3t+1$ triples. Consider a single batch $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i\in[3t+1]}$. The correctness property of Sh ensures that the triples are t-shared among $\mathcal{P}$ at the end of Sh. To check whether the triples are indeed multiplication triples, an instance of the triple transformation protocol TripTrans is invoked with this set of $(3t+1)$ t-shared triples as input. Let $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ denote the polynomials of degree at most $\frac{3t}{2}$, $\frac{3t}{2}$ and $3t$ respectively, which are guaranteed to exist at the end of the instance of TripTrans. We next use a probabilistic check to verify whether the relation $Z(\cdot) = X(\cdot)Y(\cdot)$ holds by public checking of $Z(\alpha) \stackrel{?}{=} X(\alpha)Y(\alpha)$ for a *random* $\alpha \in \mathbb{F}$; the random α can be generated by any standard technique[11] and

[10] We note that in [15] this module is designed to work in a completely asynchronous fashion, but with $t < n/4$. Since we are in the $t < n/3$ setting and want to use our VSS protocol Sh, we require the first four rounds to be synchronous broadcast rounds.

[11] For example, each party P_i can t-share a random $r^{(i)}$ and then we can set $[\alpha]_t \stackrel{\text{def}}{=} [r^{(1)}]_t + \ldots + [r^{(n)}]_t$. This is followed by publicly reconstructing α using OEC. We call this protocol as Rand().

we do not bother about the communication complexity of this procedure as it will be invoked only a constant number of times. It is trivial to see that the check passes for an honest D. For a corrupted D, if the input triples $\{([x^{(i)}]_t, [y^{(i)}]_t, [z^{(i)}]_t\}_{i\in[3t+1]}$ are not multiplication triples, then $Z(\alpha) \neq X(\alpha)Y(\alpha)$ holds (by the property of TripTrans). Therefore, the probability of a corrupt D passing the check in this scenario can be computed as the probability that $Z(\alpha) = X(\alpha)Y(\alpha)$ holds, even though $Z(\cdot) \neq X(\cdot)Y(\cdot)$. This probability is atmost $\frac{3t}{|\mathbb{F}|}$ for a random α since $Z(\cdot)$ has degree at most $3t$.

If D is honest, then through the above check, Adv learns one point on $X(\cdot), Y(\cdot)$ $Z(\cdot)$ i.e the value of the polynomials at α. However, this still leaves $\frac{3t}{2}$ degree of freedom in these polynomials. So if the verification passes, the parties output $\frac{3t}{2}$ shared triples $\{([a^{(i)}]_t, [b^{(i)}]_t, [c^{(i)}]_t)\}_{i\in[\frac{3t}{2}]}$ on the "behalf" of D, where $a^{(i)} = X(\beta_i), b^{(i)} = Y(\beta_i)$ and $c^{(i)} = Z(\beta_i)$ for $\frac{3t}{2}$ distinct β_i values, distinct from the random α. Thus the multiplication triples $\{([a^{(i)}]_t, [b^{(i)}]_t, [c^{(i)}]_t)\}_{i\in[\frac{3t}{2}]}$ are finally considered to be shared on the "behalf" of D.

The above idea is applied in parallel on all the $\mathfrak{l}$ batches of $3t+1$ t-shared triples and a single α is used for the probabilistic verification in all the batches. Through each batch $\frac{3t}{2}$ multiplication triples are considered to be shared by D and so overall the parties will get $(\mathfrak{l}\cdot\frac{3t}{2})$ t-shared multiplication triples at the end. If D is caught cheating in any of the batches, then it is discarded and some default $\mathfrak{l}\cdot\frac{3t}{2}$ multiplications triples are considered to be shared on the behalf of D. We call the resultant protocol TripleSh. In TripleSh, D needs to invoke Sh by setting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$. This ensures that D shares $\ell \times (n-t) = \mathfrak{l}(3t+1)$ triples, which when underwent through TripTrans and probabilistic check result in $(\mathfrak{l}\cdot\frac{3t}{2})$ multiplication triples being shared on the behalf of D.

The communication complexity of TripleSh will be $\mathcal{PC}(\mathcal{O}(n^3\mathfrak{l}))$ and $\mathcal{BC}(\mathcal{O}(n^3))$, which is computed as follows: the instance of Sh will have communication complexity $\mathcal{PC}(\mathcal{O}(n^3\ell))$ and $\mathcal{BC}(\mathcal{O}(n^3))$ (see Theorem 4.3). Substituting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$ and $n-t = 2t+1 = \Theta(n)$, this gives $\mathcal{PC}(\mathcal{O}(n^3\mathfrak{l}))$ and $\mathcal{BC}(\mathcal{O}(n^3))$. There will be $\mathfrak{l}$ instances of TripTrans, each having communication complexity $\mathcal{PC}(\mathcal{O}(n^3))$, thereby contributing $\mathcal{PC}(\mathcal{O}(n^3\mathfrak{l}))$ to the communication complexity.

The error probability of TripleSh is computed as follows. By setting $\ell = \frac{\mathfrak{l}(3t+1)}{n-t}$ in Theorem 4.3 we find that except with probability at most $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$, the values shared by D will be t-shared. Given that the values shared by D are t-shared, the probabilistic check ensures that except with probability at most $\mathfrak{l}\cdot\frac{3t}{2}$, the outputs values obtained on the behalf of D are indeed multiplication triples (there are $\mathfrak{l}$ batches and each batch can pass the probabilistic check with probability at most $\frac{3t}{2}$). So it follows that except with probability $\mathfrak{l}\cdot\frac{3t}{2} + \max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\} \approx \max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$, the parties output t-shared multiplication triples. The protocol will eventually terminate for each honest party: the instance of Sh will terminate, assuming that the first four communication rounds are synchronous broadcast rounds. Once Sh terminates, the instances of TripTrans which are executed asynchronously eventually terminate for each honest party. We refer to [15] for the formal details of TripleSh. For completeness, we state the properties of TripleSh in Lemma D.1, whose proof follows from the above discussion; for a detailed proof see [15].

Lemma D.1. *Given a partially synchronous communication setting where the first four rounds are synchronous broadcast rounds, protocol* TripleSh *achieves the following for every possible* Adv. **(1) Termination**: *Irrespective of* D*, every honest party eventually terminates the protocol.* **(2) Correctness**: *If* D *is honest then* $\mathfrak{l}\cdot\frac{3t}{2}$ *multiplication triples are* t*-shared. If* D *is corrupted then* $\mathfrak{l}\cdot\frac{3t}{2}$ *triples are* t*-shared; moreover except with probability at most* $\max\{\frac{n^3(n-1)}{|\mathbb{F}|-(n-t)}, \frac{n^3\mathfrak{l}(3t+1)}{|\mathbb{F}|(n-t)}\}$*, the triples will be multiplication triples.* **(3) Privacy**: *If* D *is honest, then the view of* Adv *in the protocol is distributed independently of the output multiplication triples.* **(4) Communication Complexity**: *The protocol has communication complexity* $\mathcal{PC}(\mathcal{O}(n^3\mathfrak{l}))$ *and* $\mathcal{BC}(\mathcal{O}(n^3))$*. Additionally one invocation to* Rand *is required.*

Module II : Multiplication Triple Extraction. The second module of the efficient framework of [15] is an asynchronous protocol TripExt. The input to the protocol is a set of $3t+1$ t-shared multiplication triples, where the ith triple is selected by the party P_i. It will be ensured that if P_i is *honest*, then the ith triple is random

and private. The protocol outputs a set of $\frac{t}{2} = \Theta(n)$ t-shared multiplications triples, each of which is random and unknown to Adv. The high level idea of TripExt is as follows: the input triples are first transformed using TripTrans to obtain a new set of t-shared $3t+1$ triples. Let $X(\cdot), Y(\cdot)$ and $Z(\cdot)$ be the underlying polynomials associated with the transformed triples. It follows from the correctness of TripTrans that $Z(\cdot) = X(\cdot)Y(\cdot)$ holds, since the input triples are guaranteed to be multiplication triples. Also, since Adv may know at most t input triples, by the property of TripTrans, it learns at most t points on $X(\cdot), Y(\cdot)$ and $Z(\cdot)$, leaving $\frac{3t}{2} - t = \frac{t}{2}$ degree of freedom on these polynomials. So the parties output $\{([X(\beta_i)]_t, [Y(\beta_i)]_t, [Z(\beta_i)]_t)\}_{i\in[\frac{t}{2}]}$, which can be computed as a linear function of the transformed triples. These triples are considered to be securely "extracted" from the set of input triples. The protocol eventually terminates for each honest party. As one instance of TripTrans is involved, TripExt has communication complexity $\mathcal{PC}(\mathcal{O}(n^3))$. For completeness the properties of TripExt are stated in Lemma D.2, which follows from the above discussion; for a detailed proof we refer to [15].

Lemma D.2. *Let $\{(x^{(i)}, y^{(i)}, z^{(i)})\}_{i\in[3t+1]}$ be a set of multiplication triples, where party $P_i \in \mathcal{P}$ has verifiably t-shared the triple $(x^{(i)}, y^{(i)}, z^{(i)})$. Then for every possible* Adv*, protocol* TripExt *achieves the following in a completely asynchronous setting.* **(1) Termination***: All honest parties eventually terminate the protocol.* **(2) Correctness***: Each of the $\frac{t}{2}$ output triples is a multiplication triple and t-shared.* **(3) Privacy***: The view of* Adv *in the protocol is distributed independently of the output multiplication triples.* **(4) Communication Complexity***: The protocol has communication complexity $\mathcal{PC}(\mathcal{O}(n^3))$.*

Module I + Module II $\Rightarrow$ Offline phase protocol in the partial synchronous setting. By combining TripleSh and TripExt, we get an offline phase protocol Offline in the partial synchronous setting as follows. The goal of Offline is to generate t-sharing of $c_M + c_R$ random and private multiplication triples.

- Each party P_i acts a D and ensures that $\frac{2(c_M+c_R)}{t}$ random multiplication triples are shared on its behalf. For this, it invokes an instance $\mathsf{TripleSh}_i$ of TripleSh by setting $\mathfrak{l} = \frac{4(c_M+c_R)}{3t^2}$; this ensures that at the end of $\mathsf{TripleSh}_i$, $\mathfrak{l} \cdot \frac{3t}{2} = \frac{2(c_M+c_R)}{t}$ t-shared multiplication triples are available on the behalf of P_i. This step is executed in a partially synchronous setting, where it is assumed that the first four communication rounds are synchronous broadcast rounds. This is to ensure that all the TripleSh instances are terminated. From Lemma D.1, by substituting the value of $\mathfrak{l}$, this step will have total communication complexity $\mathcal{PC}(\mathcal{O}(n^2(c_M + c_R)))$ and $\mathcal{BC}(\mathcal{O}(n^4))$. Additionally there will be one instance of Rand and its output can be used as a challenge across all the n instances of TripleSh for the verification of the shared triples. By substituting the value of $\mathfrak{l}$ and from the union bound (there are n instances of TripleSh) it follows that at the end of this step, except with probability at most $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$, the triples available on the behalf of all the parties are indeed multiplication triples.

- The parties then execute the protocol TripExt on the multiplication triples obtained at the end of the previous step and securely extract $c_M + c_R$ random and private t-shared multiplication triples. More specifically, the $\frac{2(c_M+c_R)}{t}$ shared triples available on the behalf of each party are considered as $\frac{2(c_M+c_R)}{t}$ batches of $3t+1$ triples, where the ith batch consists of the ith triple available on the behalf of all $3t+1$ parties. So each batch is of size $3t+1$. For every batch, the triples contributed by the honest parties will be random and private. So by applying an instance of TripExt, the parties can extract $\frac{t}{2}$ random and private t-shared multiplication triples. For each batch an instance of TripExt is executed and so from $\frac{2(c_M+c_R)}{t}$ batches, the parties will get total $c_M + c_R$ random and private t-shared multiplication triples. This step is executed in a completely asynchronous fashion and it will eventually terminate for each honest party, as the underlying instances of TripExt will eventually terminate. As there will be $\frac{2(c_M+c_R)}{t}$ instances of TripExt involved, from Lemma D.2, this step will have total communication complexity $\mathcal{PC}(\mathcal{O}(n^2(c_M + c_R)))$.

For completeness the properties of Offline are stated in Lemma D.3, which follows from the above discussion; for a detailed proof we refer to [15].

Lemma D.3. *Assuming that the first four rounds are synchronous broadcast rounds, protocol* Offline *achieves the following for every possible* Adv. **(1) Termination***: All honest parties eventually terminate the protocol.* **(2) Correctness***: The $c_M + c_R$ output triples are t-shared among the parties. Moreover, the output triples are multiplication triples, except with probability at most $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$.* **(3) Privacy***: The view of* Adv *is independent of the output multiplication triples.* **(4) Communication Complexity***: The protocol has communication complexity $\mathcal{PC}(\mathcal{O}(n^2(c_M + c_R)))$ and $\mathcal{BC}(\mathcal{O}(n^4))$. In addition, one invocation to* Rand *is required.*

D.3 Statistical MPC Protocol in the Partially Synchronous Setting

In our statistical MPC protocol MPC, the parties first execute the protocol Offline and generate t-sharing of $c_M + c_R$ random and private multiplication triples. For this we assume that the network is partially synchronous and the first four communication rounds are synchronous broadcast round. In parallel, each party P_i t-shares its input x_i for the computation by acting as a dealer D and invoking an instance Sh_i of Sh. These instances of Sh also utilise the first four synchronous broadcast rounds, which are utilized by Offline. Once Offline is over, the parties will have $c_M + c_R$ t-shared random and private multiplication triplets. In addition, the inputs of all the parties would be available in a t-shared fashion.

Next the parties start securely evaluating the circuit *asynchronously* on a gate by gate basis by maintaining the following *invariant* for each gate of the circuit: given t-sharing of the input(s) of a gate, the parties securely compute a t-sharing of the output of the gate. A gate is said to be evaluated if a t-sharing of the output of the gate is computed. This is achieved as follows for various gates: the linearity of t-sharing ensures that the linear gates can be evaluated locally. For a multiplication gate, the parties associate a multiplication triple from the set of preprocessed multiplication triples and then evaluate the gate by applying the Beaver's circuit randomization technique, namely by invoking an instance of Beaver. For every random gate in the circuit for generating a random value, the parties associate a multiplication triple from the set of preprocessed multiplication triples and the first component of the triple is considered as the outcome of the random gate. This explains the need for generating $c_M + c_R$ random t-shared multiplication triples in the offline phase (c_M triples corresponding to c_M multiplication gates and c_R triples corresponding to c_R random gates). Once all the gates are evaluated, the t-sharing of the output gate is publicly reconstructed. As this approach for circuit evaluation is standard and used in al most all the recent MPC protocols, we avoid giving the complete formal details of MPC.

If it is ensured that the triples from the offline phase are indeed t-shared and multiplication triples then protocol MPC correctly computes the function f. The probability that the offline phase protocol Offline fails to generate t-shared multiplication triples is at most $\frac{4n^4(c_M+c_R)(3t+1)}{3t^2(n-t)|\mathbb{F}|}$. So if we ensure that $|\mathbb{F}| \geq 4n^4(c_M + c_R)(3t + 1)2^\kappa$, then the function will be correctly computed except with an error probability of at most $2^{-\kappa}$. The protocol will achieve the privacy property, intuitively due to the following reason: the inputs of the honest parties remain private as they are t-shared. The intermediate gate outputs remain as private as possible, as they are also t-shared. This intuition can be easily formalized by giving a simulation based security proof using standard arguments (see for example [1]). The offline phase will have communication complexity $\mathcal{PC}(\mathcal{O}(n^2(c_M + c_R)))$ and $\mathcal{BC}(\mathcal{O}(n^4))$. In addition, sharing the inputs of the parties will cost $\mathcal{PC}(\mathcal{O}(n^4))$ and $\mathcal{BC}(\mathcal{O}(n^4))$. The circuit evaluation will have communication complexity $\mathcal{PC}(\mathcal{O}(c_M n^2))$, as there will be c_M instances of Beaver, while publicly reconstructing the circuit output will cost $\mathcal{PC}(\mathcal{O}(n^2))$.