Rumors MPC: GOD for Dynamic Committees, Low Communication via Constant-Round Chat

Bernardo David[1*], Arup Mondal[2†], and Rahul Satish[1*]

[1] IT University of Copenhagen
bernardo@bmdavid.com, rahs@itu.dk
[2] Ashoka University
arup.mondal_phd19@ashoka.edu.in

Abstract. Constructing MPC with ephemeral committees has gained a lot of attention since the seminal works on Fluid MPC and YOSO MPC (CRYPTO'21). However, most protocols in this setting focus on the extreme case of ephemeral committees who can only act for one round (*i.e.,* the maximally fluid case). The Layered MPC model (CRYPTO'23) recasts this notion as a protocol execution against an adaptive rushing adversary over a layered interaction graph, where each committee sits on a layer and can only communicate with the immediate next committee. Although protocols with abort allow for linear communication complexity (CRYPTO'23, CiC'24), Perfect Layered MPC with guaranteed output delivery (GOD) and its statistically secure counterpart (TCC'24) suffer from $O(n^9)$ and $O(\kappa n^{18})$ communication complexity for n parties per committee, respectively. In this work, we investigate communication complexity improvements gained in a relaxed Multi-Layered MPC model that allows for limited interaction among the parties in each committee, while still allowing only one round to communicate towards the immediate next committee. We construct Rumors MPC protocols, where the interaction among each committee's members is *constant-round*. Our protocols achieve GOD and optimal corruption threshold in the perfect (resp. statistical) security setting with committees acting for $\delta = 5$ (resp. $\delta = 13$) rounds and $O(n^6)$ (resp. $O(\kappa n^8)$) communication.

1 Introduction

Secure multiparty computation (MPC) allows n mutually distrustful parties to compute a function of their private inputs without revealing nothing else than the output even when t parties are corrupted. This notion can be achieved with computational security against a dishonest majority [GMW87] (*i.e.*, $n > t$), statistical security [RB89] when $n \geq 2t + 1$ and perfect security [CCD88, BGW88] when $n \geq 3t + 1$. These seminal results are in the standard setting where the same parties execute all protocol steps, a potential issue in large-scale long-running computations where parties may join/leave or where it is infeasible for all parties to execute all steps.

Many recent works have considered models where different sets of parties execute each round of an MPC protocol [GHK+21, CGG+21, AHKP22, RS22, DDG+23, BEP23, BDO23, KRY24, DKLZN24, BEP24]. Inspired by player replaceable large-scale blockchain protocols [CM19, DGKR18] where small committees are randomly selected to execute each round, this line of work was kick-started by a protocol for proactive secret sharing leveraging random anonymous committees [BGG+20]. While the YOSO model [GHK+21] considers MPC protocols where each round is executed by a fresh randomly selected committee of anonymous parties, the Fluid MPC model [CGG+21] does not require parties to be anonymous allows parties to join the execution for several rounds.

Besides works [GHK+21, GHM+21, CDK+22, CDGK22, BDO23, CKR+23, KRY24, CD24] in the YOSO setting where parties necessarily "only speak once", all work on the Fluid MPC so far [CGG+21, RS22, DDG+23, BEP23, DKLZN24, BEP24] has focused on the "maximally-fluid" case, where parties can only participate in the execution for exactly 1 round. In particular, we focus on the Layered MPC model [DDG+23], which recasts maximally-Fluid MPC as a standard MPC protocol execution with an interaction pattern represented by a directed layered graph of width n, where each committee of n parties is placed at a layer. Hence, each committee can only communicate towards the immediate next committee (via point-to-point and broadcast channels). Notice that this is more stringent than

*Funded by the Independent Research Fund Denmark (IRFD) grant number 0165-00079B.
†Work done while visiting the IT University of Copenhagen.

the YOSO model, where parties are allowed to send messages to other parties in arbitrary future committees.

In spite of the restrictive interaction pattern of the Layered MPC setting, achieving statistical [BEP23] and perfect [BEP24] security *with abort* in the Layered MPC setting incurs only linear communication complexity. However, the best Layered MPC protocols achieving perfect [DDG+23] and statistical [DKLZN24] security *with Guaranteed Output Delivery (GOD)* respectively require communication in the orders of $O(n^9)$ and $O(\kappa n^{18})$, where κ is the security parameter, significantly more than protocols in the standard setting. Considering the very high costs of establishing communication towards random anonymous parties [GHM+21, CDK+22, CDGK22, CKR+23, CD24], which is not estimated in these works, the communication complexity of YOSO MPC protocols [GHK+21, BDO23, CKR+23, KRY24] is presumably just as dire, although computationally secure constant-round protocols exist [KRY24].

Considering the lack of results in non-maximally Fluid setting and the significant communication complexity gap between Layered (maximally-Fluid) MPC and the standard setting, we investigate whether speaking more than once can alleviate this issue. We ask the following question, which we answer positively for the statistical and perfect security cases with Rumors MPC protocols:

Is it possible to achieve better communication in Layered MPC with GOD when allowing constant-round interaction among parties in the same committee?

1.1 Our Contributions

We introduce Multi-Layered MPC, recasting Fluid MPC [CGG+21] as MPC over a layered interaction graph with rushing adaptive adversaries à la Layered MPC [DDG+23]. In our model, each committee is present in δ consecutive layers of the interaction graph. At each layer, parties can access ideal secure authenticated point-to-point channels to each party in the next layer, *i.e.* either members of the same committee in the first $\delta - 1$ layers or members of the next committee in the last layer. Moreover, parties can access an ideal authenticated broadcast channel towards parties in the next layer. Hence, committee members can send messages to each other for $\delta - 1$ rounds and send messages to the immediate next committee in one final round. The model considers an adaptive rushing adversary that may corrupt t of n parties in each committee. Notice that Multi-Layered MPC corresponds to Layered MPC model for $\delta = 1$ and to Fluid MPC for general δ.

We construct Rumors MPC protocols, where parties in each committee are allowed $\delta - 1$ rounds of interaction for *constant* δ before handing over state to the next committee in round δ, when corrupted parties may send invalid state (as in a game of Rumors). We present the following results in this model:

- Perfectly secure Rumors MPC with GOD and optimal corruptions $n \geq 3t + 1$ where committees act for $\delta = 5$ rounds with $O(n^6)$ communication.
- Statistically secure Rumors MPC with GOD and optimal corruptions $n \geq 2t+1$ where committees act for $\delta = 13$ rounds with $O(\kappa n^8)$ communication.

Our protocols achieve significant improvements over the state-of-the-art Layered MPC protocols for the perfect [DDG+23] and statistically secure [DKLZN24] settings with GOD, which achieve communication of $O(n^9)$ and $O(\kappa n^{18})$, respectively. As discussed in more details in Section 1.3, we achieve this by using the extra rounds among members of the same committee to finish verifiable secret sharing (VSS) and multiplication protocols initiated by the *previous* committee. This allows us to more efficiently transfer state to the next committees while avoiding the main bottleneck of [DDG+23, DKLZN24], which extensively use future messaging protocols to conditionally transfer complaint resolution messages to multiple future committees. We remark that our approach requires less rounds and communication than trivially having a committee compute a multiplication and then re-share the output to the next committee.

We remark that our goal is not to completely sacrifice fluidity towards lower communication but rather to improve communication while keeping δ as small as we can. Our protocols already get significant communication savings even with δ smaller than what is needed to trivially compute a multiplication plus resharing. Although still shy of state-of-the-art GOD protocols in the standard setting, we approach the communication complexity of the original feasibility results of perfectly and statistically secure MPC plus an overhead incurred by resharing. This positions our protocols as a principled middle ground between rigidly static and fully dynamic MPC. Our results pave the way

for efficient MPC with dynamic committees in settings with delayed adaptive corruptions, where the adversary attempts to corrupt committee members as soon as they make themselves known but only gains control of such parties after a delay of a few rounds.

1.2 Related Work

YOSO MPC. Gentry et al. [GHK+21] introduced "You Only Speak Once" (YOSO) MPC, where each round of a protocol is executed by a new set of anonymous parties chosen at random to play *roles*. Parties send messages to any anonymous party playing a role in the future via an ideal *role assignment* mechanism. Since parties remain anonymous until they speak and only speak once, an adaptive adversary is limited to probabilistic corruptions. The original work achieved statistical security and, assuming a distributed key setup, computational security. Computational security without setup is achieved in [BDO23, KRY24]. Specific purpose YOSO protocols are known for threshold cryptography [EFR21, BHK+22] and random beacons [NRO22, LZMR+24] (with worst-case corruption). A limitation of YOSO MPC protocols is not achieving optimal corruption thresholds due to probabilistic corruptions.

Fluid MPC. Choudhuri et al. [CGG+21] introduced the Fluid MPC model. In this model, the set of parties involved in the protocol execution can be different from one round to the next one (but not necessarily disjoint nor anonymous as in YOSO). They presented a maximally-fluid (*i.e.* a new disjoint committee executes each round) statistically secure protocol with abort achieving optimal corruption bounds of t out of $n \geq 2t + 1$ parties [CGG+21]. Le Mans [RS22] introduces a computationally secure protocol in preprocessing model with security against a dishonest majority. Protocols achieving optimal corruption thresholds and linear communication complexity for maximally-fluid MPC with abort are shown in [BEP23] (statistical security) and [BEP24] (perfect security). Multi-layered MPC recasts Fluid MPC as MPC over a layered interaction graph with rushing adaptive adversaries.

Layered MPC. David et al. [DDG+23] introduced the notion of Layered MPC, capturing the worst case of maximally-Fluid MPC as an instance of standard MPC where the interaction pattern is defined by a layered graph of width n, allowing each party to send secret messages and broadcast messages only to parties in the next layer. Their protocol achieves perfect security with guaranteed output delivery (GOD) against rushing adaptive adversaries corrupting at most t out of $n \geq 3t + 1$ parties in each layer with $O(n^9)$ communication complexity. A statistically secure protocol with GOD for the case of $n \geq 2t+1$ with $O(n^{18})$ communication complexity is presented in [DKLZN24]. We remark the chasm in communication complexity in comparison to maximally-fluid MPC with abort [BEP23, BEP24].

Communication Among (Anonymous) Dynamic Committees. Different methods have been proposed to realize communication towards anonymous randomly chosen parties as required by YOSO, including realizing [GHM+21, CKR+23] the role assignment notion from [GHK+21] (inspired by [BGG+20]) and a new notion of Encryption to the Future (ETF) [CDK+22]. Publicly verifiable proactive secret sharing schemes in the YOSO model were proposed in [CDGK22, CD24]. A shared issue in these approaches is that sending messages to committees that have not yet been selected requires an interactive protocol with auxiliary committees [CDK+22].

Other Models. While we focus on the Fluid, Layered and YOSO MPC models, protocols in similar models have been proposed in [OY91, FHM98, GPS19, BJMS20, EHOY20, AHKP22, ELL22, DEP23]. The statistically secure [AKP23] (resp. perfectly secure [AKP20]) information theoretical MPC protocol with constant round complexity constructed as a feasibility result has communication complexity exponential in the number of parties (resp. conjectured to be higher than YOSO and Layered MPC protocols, as the authors do not estimate this complexity).

1.3 Technical Overview

Multi-Layered MPC model: We recast Fluid MPC [CGG+21] as MPC over a layered interaction graph with rushing adaptive adversaries, relaxing Layered MPC to allow for each committee $\mathcal{C}_i$ of n

parties to be assigned to δ multiple consecutive layers of the interaction graph. The rest of the model remains the same: 1. protocols are executed in $q = \delta \cdot d$ synchronous rounds mapped into layers $\mathcal{L}_i$ (where d is related to circuit depth in concrete protocols); 2. parties in $\mathcal{L}_i$ may send messages to parties in $\mathcal{L}_{i+1}$ via ideal secure point-to-point channels or via a broadcast channel; 3. functionalities take inputs from a party P_j^i in committee $\mathcal{C}_i$ and deliver outputs to a party $P_{j'}^{i'}$ in committee $\mathcal{C}_{i'}$ potentially for $j' \neq j$ and $i' \neq i$; 4. an adaptive rushing adversary may corrupt up to t parties in each committee. In practical terms, this means that parties in each committee $\mathcal{C}_i$ may interact with each other for up to $\delta - 1$ rounds but at the δ-th round (*i.e.* the last layer to which they are assigned in the interaction graph) must handover their state to committee $\mathcal{C}_{i+1}$ and may not communicate with $\mathcal{C}_j$ for $j > i + 1$.

Rumors-MPC with Perfect Security: Adapting the BGW [BGW88] approach directly to our setting would yield $\delta = 2 \cdot n_{\mathsf{VSS}} + 1$, where n_{VSS} is the round complexity of an underlying VSS scheme, without an improvement on communication complexity due to having each committee run two VSS instances in sequence followed by a reinforced re-sharing (which is the main communication bottleneck). Hence, we base our protocol on a black-box perfectly secure VSS scheme and follow the the CDM [CDM00] approach to obtain a multiplication protocol. Naively running a CDM multiplication within a committee $\mathcal{C}_i$ and then re-sharing the result towards the next committee $\mathcal{C}_j$ would imply that $\mathcal{C}_i$ must execute at least $\delta = 2 \cdot n_{\mathsf{VSS}} + 1$ rounds (as in BGW). As our main result in the statistical setting, we show that we can split up this protocol in a way that two successive committees execute a VSS instance each, and that the second VSS instance can be optimistically skipped in case no parties act maliciously. The main challenge lies exactly in improving the parameter δ while avoiding the explosion in communication complexity incurred by sending messages from a committee $\mathcal{C}_i$ to a future committee $\mathcal{C}_j$ for $j > i + 1$ using $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_{j-1}$ to securely send the message as in [DDG+23, DKLZN24].

We work with layered circuits and assume that a committee $\mathcal{C}_i$ holds shares of the inputs to the circuit layer it will start computing. Such shares are obtained from $\mathcal{C}_{i-1}$ via re-sharing. $\mathcal{C}_{i-1}$ either computed the previous circuit layer or freshly shared the circuit's inputs in case it is the input committee $\mathcal{C}_1$. Recall that in CDM multiplication, we run an instance of parallel VSS to obtain an intermediate result and check values, perform a check and, if there are complaints, invoke another instance of parallel VSS to resolve them. In order to avoid having both parallel VSS instances executed in sequence by $\mathcal{C}_i$, we design a protocol where $\mathcal{C}_i$ executes the first parallel VSS instance, while the second parallel VSS instance is executed by $\mathcal{C}_{i+1}$, in case there are complaints. The crux of our solution is efficiently re-sharing all the intermediate values needed by $\mathcal{C}_{i+1}$ in order to finalize the multiplication when there are complaints.

We re-share the intermediate results and check values from $\mathcal{C}_i$ to $\mathcal{C}_i + 1$ with the help of random shared secrets. Computing these random sharings sequentially before or after the first parallel VSS invocation in $\mathcal{C}_i$ would require n_{VSS} extra rounds. In order to avoid this, we pre-compute the necessary random sharings using instances of parallel VSS executed concurrently with the execution of the parallel VSS instance used for multiplication. Hence, we obtain the random sharings after n_{VSS} rounds, at the same time we obtain the shares of intermediate results and check values. In its last round, $\mathcal{C}_i$ uses the random sharings to re-share the intermediate results and check values towards $\mathcal{C}_{i+1}$ while avoiding the use of expensive secure future messaging methods of [DDG+23, DKLZN24].

Putting these ideas together, committee $\mathcal{C}_i$ starts the execution of a multiplication and re-shares the intermediate result and values necessary to verify this result and resolve complaints. Parties in $\mathcal{C}_{i+1}$ who do not complain can optimistically start computing the multiplications in next layer. If no complaints happen in the first round of $\mathcal{C}_{i+1}$'s execution, the multiplication was completed within $n_{\mathsf{VSS}} + 1$ rounds by $\mathcal{C}_i$, *i.e.* n_{VSS} rounds for the main multiplication plus 1 round for re-sharing. Otherwise, all parties $\mathcal{C}_{i+1}$ can resolve the complaint and complete the multiplication in $n_{\mathsf{VSS}} + 1$ rounds, where the last round is used to re-share the results towards $\mathcal{C}_{i+2}$, who starts computing the next layer. Hence, the worst-case round complexity for this protocol is $2(n_{\mathsf{VSS}} + 1)$ with $\delta = n_{\mathsf{VSS}} + 1$, while the communication complexity is $\mathcal{O}(n^2 \cdot \mathsf{Comm}_{\mathsf{VSS}})$.

Instantiating this protocol with the VSS from [GIKR01], we obtain $\delta = 5$ and communication complexity in $\mathcal{O}(n^6)$, a significant improvement on [DDG+23] and only a n multiplicative factor above BGW [BGW88]. Communication complexity and round complexity for individual sub-protocols are presented after each construction. We determine total round complexity assuming that sub-protocols

are executed as soon as possible by the committee that initiates its execution instead of sequentially executing each of them when they are called. See Table 1 for a comparison.

Scheme	Maximality (δ)	Committees	Communication
BGW [BGW88]	$\mathcal{O}(d \cdot n_{\mathsf{VSS}})$ ($\mathcal{O}(n_{\mathsf{VSS}})$ per mult.)	$1^\star$	$\mathcal{O}(n^2 \cdot \mathsf{Comm}_{\mathsf{VSS}} + n^5)$
Rumors MPC (this work)	$n_{\mathsf{VSS}} + 1$	2	$\mathcal{O}(n^4 \cdot \mathsf{Comm}_{\mathsf{VSS}})$
Layered-MPC [DDG+23]	1	11	$\mathcal{O}(n^9)$ [DDG+23]

Table 1. Worst-case communication complexity, number of committees and number of rounds δ executed by each committee for perfectly secure multiplication where n_{VSS} and $\mathsf{Comm}_{\mathsf{VSS}}$ are respectively the round and communication complexities of a perfectly secure VSS (*i.e.* $n_{\mathsf{VSS}} = 3, \mathsf{Comm}_{\mathsf{VSS}} = \mathcal{O}(n^3)$ for [AKP20] or $n_{\mathsf{VSS}} = 4, \mathsf{Comm}_{\mathsf{VSS}} = \mathcal{O}(n^2)$ for [GIKR01]). *In BGW, one multiplication takes $\mathcal{O}(n_{\mathsf{VSS}})$ rounds but one committee computes all gates, so it must execute $\mathcal{O}(d \cdot n_{\mathsf{VSS}})$ rounds for circuit multiplicative depth d.

Rumors MPC with Statistical Security: In the statistically secure setting, our approach is based on constructing a VSS using the information theoretical signature technique from [RB89] and using it for multiplication à la [GRR98]. Simply executing this multiplication protocol within one committee and re-sharing the result towards the next committee would result in a high number of rounds δ executed by each committee without any improvement in communication complexity (*i.e.* a trade-off between number of committees and δ). Hence, we aim at splitting the the multiplication protocol among two committees. However, in this case, we cannot even execute an entire instance of parallel VSS within a single committee without incurring an already high round complexity. Hence, we will construct a parallel linearly homomorphic VSS protocol where the sharing phase is started by a committee $\mathcal{C}_l$ and finished by the next committee $\mathcal{C}_{l+1}$, who can execute the reveal phase to open linear combinations of the shared values.

Our starting point is a parallel conditional future broadcast protocol that allows senders in $\mathcal{C}_l$ to initiate broadcasting a message towards receivers in $\mathcal{C}_{l+1}$ in such a way that the message is revealed if the honest receivers agree to do so. The idea in this protocol is to secret share each message m and compute n information theoretically secure Message Authentication Code (MAC) tags $M_{i,j}$ on each share m_i with different keys $K_{i,j}$ for $1 \leq j, k \leq n$, before sending to each party $P_i \in \mathcal{C}_{l+1}$ a share m_i, tags $M_{i,j}$ and keys $K_{i,j}$ for $1 \leq j \leq n$. When choosing to reveal m, all parties $P_i \in \mathcal{C}_{l+1}$ first broadcast their share m_i and tags $M_{i,j}$, and then broadcast their keys $K_{i,j}$, allowing all parties to check the validity of each share. Since we use a linearly homomorphic MAC, the same idea works for revealing linear combinations of multiple message. Moreover, we use the same idea to reveal a message to a single party $P_j \in \mathcal{C}_{l+1}$ by having all parties $P_i \in \mathcal{C}_{l+1}$ send $m_i, M_{i,j}$ and then $K_{i,j}$ directly to this P_j.

We then construct an information theoretical signature that a sender in $\mathcal{C}_l$ can use to sign a message to be checked by a receiver $R \in \mathcal{C}_{l+1}$. We adopt a construction similar to the information checking protocol of [RB89], where the sender computes κ (security parameter) MAC tags under independent keys on each message m for each party in $\mathcal{C}_{l+1}$, sending m and all tags to R and sets of κ keys to each party $P_i \in \mathcal{C}_{l+1}$. Moreover, the sender uses the conditional future broadcast to broadcast all messages and MAC keys towards $\mathcal{C}_{l+1}$. Parties in $\mathcal{C}_{l+1}$ then perform a cut-and-choose check, selecting half of the MAC tags to be checked at random. If the check fails, the message previously broadcast by the sender is revealed and used as the output. Otherwise, if the receiver later decides to reveal the message, the remaining MAC tags and keys that were not revealed during the cut-and-choose check are revealed and used to verify the message revealed by the receiver. Since all building blocks are linearly homomorphic, the protocol allows for revealing information theoretical signatures on linear combinations of different messages *from the same sender*.

As an intermediate building block, we obtain a distributed commitment protocol from a sender $S \in \mathcal{C}_l$ to a receivers in $\mathcal{C}_{l+1}$. This is a straightforward construction based on our information theoretical signatures: 1. Setup: S shares the message and initiates an information theoretical signature on each share towards each party $P_i \in \mathcal{C}_{l+1}$ with P_i acting as the receiver; Reveal: all parties $P_i \in \mathcal{C}_{l+1}$ reveal

their share via the information theoretical signature protocol, allowing all other parties to verify that the share was received from the sender and to gather $t+1$ shares to reconstruct the secret (in case the sender was honest). This protocol also allows for revealing linear combinations of distributed commitments to different messages *from the same sender*.

Finally, we construct a parallel VSS protocol allowing dealers in $\mathcal{C}_l$ to initiate a sharing phase that is finalized by parties in $\mathcal{C}_{l+1}$, who can later reconstruct the secret. However, the multiplication protocol [GRR98] we wish to instantiate requires parties to reconstruct linear combinations of secrets *from different dealers* and our building blocks so far only allow for computing linear combinations of messages from the same sender. We overcome this with a sharing phase where each dealer $D \in \mathcal{C}_l$ computes distributed commitments to the coefficients of polynomials defining Shamir shares of their secrets s and of random values r, allowing parties to compute distributed commitments to each Shamir share. Additionally, each D sends Shamir shares s_i and r_i of each of their secrets s and random values r to $P_i \in \mathcal{C}_{l+1}$. All $P_i \in \mathcal{C}_{l+1}$ act as senders to setup their own distributed commitment to s_i, r_i for all secrets shared by all $D \in \mathcal{C}_i$ towards all other parties in $\mathcal{C}_{l+1}$ (instead of committing towards the next committee). Now, parties in $\mathcal{C}_{l+1}$ can open random linear combinations of the distributed commitments from both the original dealers and from each $P_i \in \mathcal{C}_{l+1}$ in order to check they correspond to the same valid share. Now, since all shares s_i of a secret s given to $P_i \in \mathcal{C}_{l+1}$ by all $D \in \mathcal{C}_l$ are in distributed commitments setup by the same P_i as sender, linear combinations of shares from different senders can be revealed.

For the final multiplication protocol based on [GRR98], we again assume that parties in $\mathcal{C}_l$ have shares of the values to be multiplied, which they obtained by computing the last circuit layer or by sharing the inputs. Parties in $P_i \in \mathcal{C}_l$ use their last round to initiate the sharing phase in a parallel linearly homomorphic VSS allowing parties in $\mathcal{C}_{l+1}$ to obtain shares of each P_i's shares of each value to be multiplied as well as of the result of P_i locally multiplying this share. Now parties in $\mathcal{C}_{l+1}$ execute a test using linear combinations in order to verify that the claimed shares of the multiplication result obtained from $\mathcal{C}_l$ are valid. In case the check fails for the shares from a given $P_i \in \mathcal{C}_l$, they reconstruct P_i's shares in the open to continue the computation. Finally, after obtaining shares of the result, parties in $\mathcal{C}_{l+1}$ initiate the computation of the next layer, to be finished by $\mathcal{C}_{l+2}$. If $\mathcal{C}_{l+1}$ finishes computing the last circuit layer, it reconstructs the output in the last round, so the full computation requires $d+1$ committees.

This protocol requires each committee to execute $\delta = 13$ rounds with communication complexity $\mathcal{O}(\kappa \cdot n^8)$, significantly improving on [DDG+23]. Communication and round complexities for individual sub-protocols are presented after each construction. We determine total round complexity assuming that sub-protocols are executed as soon as possible by the committee that initiates its execution instead of sequentially executing each of them when called. See Table 2 for a comparison.

Scheme	Maximality (δ)	Committees	Communication
Rabin and Ben-or [RB89] with [CDD+99] optimization	$\mathcal{O}(d)$ ($\mathcal{O}(1)$ per mult.)	$1^\star$	$\mathcal{O}(\kappa \cdot n^5)$
Rumors MPC (this work)	13	$1^{\star\star}$	$\mathcal{O}(\kappa \cdot n^8)$
Layered-MPC [DKLZN24]	1	$\mathcal{O}(1)^{\star\star\star}$	$\mathcal{O}(\kappa \cdot n^{18})$

Table 2. Worst-case communication complexity, number of committees and number of rounds δ executed by each committee for statistically secure multiplication where k is the security parameter. *In [RB89, CDD+99], multiplication is constant-round but the single committee computing all gates executes $\mathcal{O}(d)$ rounds for circuit multiplicative depth d. **Full computation takes $d+1$ committees: multiplications start in last round of $\mathcal{C}_l$ and end in $\mathcal{C}_{l+1}$. ***In [DKLZN24], the number of committees is a presumably large constant not shown in the paper, the communication complexity was obtained from the authors.

2 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$. In threshold settings, we use n to denote the number of parties and t the corruption threshold such that $0 < t < n$. We use $[a, b]$ for $a, b \in \mathbb{Z}$, $a \leq b$, to denote $\{a, a+1, \ldots, b-1, b\}$. $[b]$ denotes the set $[1, b]$. We denote the concatenation of x and y by $(x\|y)$. Given a set $\mathcal{X}$, we denote by $x \leftarrow_{\$} \mathcal{X}$ the sampling of a value x from the uniform distribution on $\mathcal{X}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if it vanishes faster than any polynomial. We denote by $x = \mathsf{val}$ or $x \leftarrow \mathsf{val}$ the assignment of a value val to the variable x. We denote evaluating a PPT algorithm $\mathcal{S}$ that produces an output out from an input in with randomness $r \xleftarrow{\$} \{0,1\}^*$ as $\mathsf{out} \leftarrow \mathcal{S}(\mathsf{in}; r)$, omitting the randomness when it is obvious or not explicitly required.

2.1 Multi-Layered MPC

We recast Fluid MPC [CGG+21] as MPC over a layered interaction graph with rushing adaptive adversaries à la Layered MPC [DDG+23]. We consider that each committee can communicate over multiple layers, before handing over protocol execution to the next committee. This allows committee members to exchange (and broadcast) messages among themselves before for a fixed amount of rounds (*independent* from the circuit being computed in our protocols) before a single round of point-to-point (and broadcast) communication to parties in the next committee.

Definition 1. *Let n, t, δ, d be positive integers. An (n, t, δ, d)-layered protocol is a synchronous protocol Π over secure point-to-point channels and a broadcast channel with the following features:*

- ***Rounds and Synchrony:*** *The protocol is executed in $q = \delta \cdot d$ synchronous communication rounds mapped onto layers $\mathcal{L}_i$, for $1 \leq i < q$ of a layered interaction graph.*
- ***Interaction pattern:*** *The interaction consists of q rounds, where in round i, parties in $\mathcal{L}_i$ may send messages to parties in $\mathcal{L}_{i+1}$ over secure point-to-point channels. By default, we additionally allow each party in $\mathcal{L}_i$ to send a broadcast message to all parties in $\mathcal{L}_{i+1}$.*
- ***Committees:*** *There are d committees $\mathcal{C}_i$ for $i \in [d]$, each consisting of n parties. Each committee $\mathcal{C}_i$ is assigned to consecutive layers from $\mathcal{L}_{\delta(i-1)+1}$ to $\mathcal{L}_{i\delta}$ in the interaction graph. The j'th party in committee $\mathcal{C}_i$ is denoted by P_j^i. The first committee $\mathcal{C}_1$ and the last committee $\mathcal{C}_d$ are the committees consisting of the input and output clients respectively.*
- ***Functionalities:*** *We consider functionalities f that takes inputs from input clients and deliver outputs to output clients.*
- ***Adversaries:*** *We consider an active adaptive rushing adversary, $\mathcal{S}$, who may corrupt a set, $\mathcal{I}_i$, of upto t parties in each committee $\mathcal{C}_i$, for all $i \in [d]$.*

We say that a protocol Π is a Multi-Layered MPC protocol for $\mathcal{F}$ if it realizes $\mathcal{F}$ in the standard sense of secure MPC [Can00] with guaranteed output delivery.

We observe that this definition is equivalent to the definition of Layered MPC provided in [DDG+23], if we set $\delta = 1$. This is also equivalent to Maximally Fluid MPC [CGG+21] as each committee would then be allowed to speak only for a single round before a handover. Similarly, by setting $\delta = r_\Pi$ where r_Π is the number of rounds required to execute some MPC protocol Π, we have MPC in the standard model where one committee stays for the entire protocol execution.

Adaptive Security and Composition Following the same approach as in the original Layered MPC model [DDG+23] and [AL17], we define and prove security of our protocols in the real/ideal paradigm [Can00] considering static adversaries and stand-alone executions. Similarly to these previous works, in order to obtain the desired adaptive security guarantees, we use the general results from [CDD+04] implying that our protocols achieve adaptive security when allowing for inefficient simulators (*i.e.* of running time super polynomial in running time of the adversary). Moreover, in order to show that our protocols achieve security under arbitrary composition (*i.e.* in the sense of the Universal Composability framework [Can01]), we also invoke the results of [KLR06].

2.2 Building Blocks

Secret Sharing. A (t, n)-secure threshold secret sharing scheme takes a secret s as input and computes n shares with the secrecy guarantee that any set of at most t shares reveal nothing about s while any set of $t+1$ correct shares can be used to efficiently reconstruct the secret. A robust secret sharing scheme additionally guarantees correct reconstruction from n shares, even when an arbitrary set of t out of the n shares have been tampered. We denote a robust secret sharing scheme as a pair of algorithms $(\mathsf{Sh}, \mathsf{Rec})$ The sharing algorithm Sh takes as input a secret $s \in S$ and randomness ρ chosen uniformly from some domain R and computes $(s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s, \rho)$, where $s_i \in U$ will be called the i-th share of the secret s. If ρ is clear from the context we just write $(s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s)$. The reconstruction algorithm Rec takes as input the set of all shares and computes $\hat{s} \leftarrow \mathsf{Rec}(s_1, \ldots, s_n)$. When explicitly using Shamir secret sharing, we denote by $\langle s, f(x) \rangle_l$ the shares $f(i)$ of a secret $s = f(0)$ defined by polynomial $f(x)$ held by $P_i^l \in \mathcal{C}_l$ for $1 \leq i \leq n$. Moreover, we denote by $[\![s, f(x), \{f_i(x)\}_{i\in[n]}]\!]_l$ the "reinforced sharing" or "double sharing" of secret s, *i.e.* Shamir shares $f(i)$ of a secret $s = f(0)$ as defined by polynomial $f(x)$ held by $P_i^l \in \mathcal{C}_l$ and Shamir shares $f_i(j)$ of $f_i(0) = f(i)$ as defined by polynomial $f_i(x)$ held by $P_j^l \in \mathcal{C}_l$ for $1 \leq i, j \leq n$. We present a formal definition in Appendix A.1.

Message Authentication Codes. We use a standard construction of information theoretical Message Authentication Codes (MACs) that is linearly homomorphic for correlated keys. For a message $m \in \mathbb{F}$ and a key $K = (A, B) \in \mathbb{F}^2$ we denote computing a MAC tag M by $M = \mathsf{MAC}_K(m) = A \cdot m + B$. Notice that, for keys $K_1 = (A, B_1), \ldots, K_n = (A, B_n)$ with the same A and independent $B_1, \ldots, B_n$, this MAC is linearly homomorphic. We formally define MACs and present this construction in Appendix A.2.

Other Building Blocks ($\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{RandomShare}}, \mathcal{F}_{\mathsf{ITSigClassical}}, \mathcal{F}_{\mathsf{DistCommitClassical}}, \mathcal{F}_{\mathsf{Beacon}}$). In our constructions, we also employ standard ideal functionalities and protocols for Verifiable Secret Sharing (Appendix A.3), Random Sharings (Appendix A.4), Information Theoretical Signatures (Appendix A.5), Distributed Commitments (Appendix A.6) and Random Beacons (Appendix A.7).

3 Perfectly Secure Rumors MPC Protocol

In this section, we start by building a protocol for VSS resharing and reinforced resharing. These protocols are then used to build a protocol for multiplication with a helper party and multiplication without a helper party. Finally, we use those primitives to build a protocol for layered MPC for any layered circuit $\mathbf{C}$.

3.1 Reshare Protocol

We define the a resharing functionality $\mathcal{F}_{\mathsf{Reshare}}$ in Figure 1 and realize with Protocol Π_{Reshare} in Figure 2, which uses a random sharing functionality to reshare a existing VSS sharing. The advantage of this approach is twofold: 1. it reduces the number of rounds used in the committee towards which the resharing happens; 2. the random sharing execution is independent from the secret being reshared, so it can begin as soon as the resharing committee executes their first round.

As the resharing protocol calls t instances of $\mathcal{F}_{\mathsf{RandomShare}}$, the communication complexity of $\Pi_{\mathsf{RandomShare}}$ is $\mathcal{O}(n^2 \cdot \mathsf{comm}_{\mathsf{VSS}})$. We provide a full proof for the above protocol in Appendix B.1.

Lemma 2. *If $n, t \in \mathbb{N}$ and $(\mathsf{Sh}, \mathsf{Rec})$ is a robust (t, n) secret-sharing scheme (in Definition 14), then Π_{Reshare} (in Figure 2) realizes the functionality $\mathcal{F}_{\mathsf{Reshare}}$ (in Figure 1) in the $\mathcal{F}_{\mathsf{RandomShare}}$-hybrid model, with perfect security for $t < n/3$.*

3.2 Reinforced Resharing Protocol

David et al. [DDG+23] define the notion of reinforced resharing as an enhanced form of Shamir secret sharing. We recall and formalized the notion of reinforced resharing from [DDG+23]. We

Public Parameters: Sending Committee $\mathcal{C}_l$, Receiving Committee $\mathcal{C}_{l+1}$.
Secret Inputs: $\langle s_i, f_i(x)\rangle_l$
Notation: $\langle s, f'(x)\rangle_{l+1} \leftarrow \langle s, f(x)\rangle_l$.
Additional Input to Functionality: Corruption Sets $\mathcal{I}_l, \mathcal{I}_{l+1}$

1. Upon receiving $(\mathsf{Reshare}, sid, f(i))$ from all honest parties $P_i^l \in \mathcal{C}_l \backslash \mathcal{I}_l$, it reconstructs the secret $s = f(0)$.
2. $\mathcal{F}_{\mathsf{Reshare}}$ samples a random degree-t polynomial $f'(.)$ such that $f'(0) = s$.
3. For every corrupt party $P_j^{l+1} \in \mathcal{I}_{l+1}$, $\mathcal{F}_{\mathsf{Reshare}}$ sends the shares $f'(j)$ to the (ideal) adversary $\mathcal{S}$.
4. It outputs $(\mathsf{OutReshare}, sid, f'(i))$ to P_i^{l+1} to distribute $\langle s, f'(x)\rangle_{l+1}$ in $\mathcal{C}_{l+1}$.

Figure 1: Functionality $\mathcal{F}_{\mathsf{Reshare}}$

Public Parameters: Sending Committee $\mathcal{C}_l$, Receiving Committee $\mathcal{C}_{l+1}$.
Secret Inputs: $\langle s, f(x)\rangle_l$.
Output: $\langle s, f'(x)\rangle_{l+1}$, where $f'(x) = s + \sum_{k=1}^{t} \alpha_k x^k$.

Committee $\mathcal{C}_l$:
- Each P_i^l sends $(\mathsf{RandShare}, sid)$ to t parallel instances of $\mathcal{F}_{\mathsf{RandomShare}}$ to generate t random sharings $\langle \alpha_k, f_k(x)\rangle_l \leftarrow \$$ for $k \in [t]$.
- Let $s_j = f(j)$ be the share of s held by P_j^l. For each $k \in [t]$, let $\alpha_{k,j} = f_k(j)$ be the share of α_k held by P_j^k. Then, P_j^k sends $s'_{i,j} = s_j + \sum_{k=1}^{t} i^k \alpha_{k,j}$ to P_i^{l+1} for each $i \in [n]$.

Committee $\mathcal{C}_{l+1}$: For $i \in [n]$, P_i^{l+1} stores $s'_i = \mathsf{Rec}(s'_{i,1}, \ldots, s'_{i,n})$ as its share of s.

Figure 2: Protocol Π_{Reshare} in the $\mathcal{F}_{\mathsf{RandomShare}}$-hybrid model

formalized the functionality for reinforced resharing in Figure 3. The functionality requires that the input parties hold a valid Shamir secret sharing, and that at most t input parties are corrupt. The protocol $\Pi_{\mathsf{DoubleReshare}}$ (Figure 3) makes calls to n instances of $\mathcal{F}_{\mathsf{Reshare}}$, hence the communication complexity is $\mathcal{O}(n^3 \cdot \mathsf{comm}_{\mathsf{VSS}})$. We state this protocol's security in Lemma 3, and give a full proof in Appendix B.2.

Public Parameters: Sending Committee $\mathcal{C}_l$, Receiving Committee $\mathcal{C}_{l+1}$.
Secret Inputs: $\langle s, f(x)\rangle_l$.
Notation: $[\![s, f'(x), \{f'_i(x)\}_{i\in[n]}]\!]_{l+1} \leftarrow \langle s, f(x)\rangle_l$.

1. Upon receiving $(\mathsf{DoubleReshare}, sid, f(i))$ from all honest parties $P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_l$, it reconstructs the secret $s = f(0)$.
2. $\mathcal{F}_{\mathsf{DoubleReshare}}$ samples a random degree-t polynomial $f'(.)$ such that $f'(0) = s$.
3. For all $i \in [n]$, samples another random degree-t polynomial $f'_{s_i}(.)$ such that $f'_{s_i}(0) = s_i = f'(i)$.
4. For every corrupt party $P_j^{l+1} \in \mathcal{I}_{l+1}$, $\mathcal{F}_{\mathsf{DoubleReshare}}$ sends the shares $(f'(j), \{f'_{s_i}(j)\}_{P_i^j \in \mathcal{C}_l \setminus \mathcal{I}_l})$ to the (ideal) adversary $\mathcal{S}$.
5. It outputs $(\mathsf{OutDoubleReshare}, sid, (f'(j), \{f'_{s_i}(j)\}_{P_i^j \in \mathcal{C}_l}))$ to P_j^{l+1} to distribute $[\![s, f'(x), \{f'_i(x)\}_{i\in[n]}]\!]_{l+1}$ in $\mathcal{C}_{l+1}$.

Figure 3: Functionality $\mathcal{F}_{\mathsf{DoubleReshare}}$

Lemma 3. *Let $n, t \in \mathbb{N}$, and $(\mathsf{Sh}, \mathsf{Rec})$ be a robust (t, n) secret-sharing scheme (in Definition 14) and $\mathcal{F}_{\mathsf{Reshare}}$ be a resharing functionality, then $\Pi_{\mathsf{DoubleReshare}}$ (in Figure 4) realizes the functionality $\mathcal{F}_{\mathsf{DoubleReshare}}$ (in Figure 3) in the $\mathcal{F}_{\mathsf{Reshare}}$-hybrid model, with perfect security for $t < n/3$.*

3.3 Multiplication with and without Helper Party

We now present an multi-layered multiplication protocol based on the multiplication protocol from CDM [CDM00]. As CDM protocol, we first define a functionality $\mathcal{F}_{\mathsf{MultHelper}}$ for multiplication with a helper party in the multi-layered setting in Figure 5 and realize it with Protocol $\Pi_{\mathsf{MultHelper}}$ as shown in Figure 7. Later, we use $\mathcal{F}_{\mathsf{MultHelper}}$ to realize the multiplication functionality from Figure 6 via Protocol Π_{MultCDM} shown in Figure 8.

Public Parameters: Sending Committee $\mathcal{C}_l$, Receiving Committee $\mathcal{C}_{l+1}$.
Secret Inputs: $\langle s, f(x)\rangle_l$
Output: $[\![s, f'(x), \{f'_i(x)\}_{i\in[n]}]\!]_{l+1}$.

Committee $\mathcal{C}_l$:

1. Each P_i^l sends $(\mathsf{RandShare}, sid)$ to t parallel instances of $\mathcal{F}_{\mathsf{Reshare}}$ to generate t random sharings $\langle \alpha_k, g_k(x)\rangle_l \leftarrow \$$ for $k \in [t]$.
2. For $i \in [n]$, define
$$\langle s_i, g'_i(x)\rangle_l = \langle s, f(x)\rangle_l + \sum_{k=1}^{t} i^k \langle \alpha_k, g_k(x)\rangle_l$$
where we define $f'(x) := s + \sum_{k=1}^{t} \alpha_k x^k$ and $s_i = f'(i) = g'_i(0)$.
3. For $i \in [n]$, each P_j^l sends $(\mathsf{Reshare}, sid, g'_i(j))$ to $\mathcal{F}_{\mathsf{Reshare}}$ to obtain $\langle s_i, f'_i(x)\rangle_{l+1}$ where $f'_i(x) = f'(i) + \alpha_{i,1}x + \cdots + \alpha_{i,t}x^t$. In addition, P_j^l reveals it's share $s_{i,j}$ corresponding to $\langle s_i, g'_i(x)\rangle_l$ to P_i^{l+1}.

Committee $\mathcal{C}_{l+1}$: For each $i \in [n]$, P_i^{l+1} locally computes $s_i = \mathsf{Rec}(s_{i,1}, \cdots, s_{i,n})$ and parties in $\mathcal{C}_{l+1}$ store $[\![s, f'(x), \{f'_i(x)\}_{i\in[n]}]\!]_{l+1}$

Figure 4: Protocol $\Pi_{\mathsf{DoubleReshare}}$ in the $\mathcal{F}_{\mathsf{Reshare}}$-hybrid model.

Public Parameters: Helper Party $P_h^l \in \mathcal{C}_l$, Output committee $\mathcal{C}_{l+1}$
Secret Inputs: $\langle \alpha, f(x)\rangle_l, \langle \beta, g(x)\rangle_l$, P_h^l holds α, β.

- Upon receiving $(\mathsf{MultHelp}, sid, f(i), g(i))$ from all honest parties $P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_l$, and $(\mathsf{MultHelpShare}, sid, \gamma)$ from P_h^l, $\mathcal{F}_{\mathsf{MultHelper}}$ interpolates f, g. It sends to the (ideal) adversary, the values $f(i), g(i)$ for all $P_i^l \in \mathcal{I}_l$.
- For a corrupt helper, $P_h^l \in \mathcal{I}_l$, send f, g to the (ideal) adversary and receive $h'(.)$ from the (ideal) adversary. If $h'(0) = f(0) \cdot g(0)$, output $(\mathsf{OutMultHelper}, sid, h'(i))$ to $P_i^{l+1} \in \mathcal{C}_{l+1}$, else it outputs $(\mathsf{OutMultHelper}, sid, \perp)$ to all parties $P_i^{l+1} \in \mathcal{C}_{l+1}$.
- For an honest helper $P_h^l \in \mathcal{C}_l \setminus \mathcal{I}_l$, sample a random polynomial $h(.)$ such that $h(0) = \alpha \cdot \beta$. It sends to the (ideal) adversary the values $h(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$ and outputs $(\mathsf{OutMultHelper}, sid, h(i))$ to $P_i^{l+1} \in \mathcal{C}_{l+1}$.

Figure 5: Functionality $\mathcal{F}_{\mathsf{MultHelper}}$

Public Parameters: Input committee $\mathcal{C}_l$, Output committee $\mathcal{C}_{l+1}$
Secret Inputs: $[\![\alpha, f(x), (f_i(x))_{i\in[n]}]\!]_l$, $[\![\beta, g(x), (g_i(x))_{i\in[n]}]\!]_l$

1. Upon receiving $(\mathsf{MultCDM}, sid, f(i), g(i), \{f_j(i), g_j(i)\}_{j\in[n]})$ from all honest parties $P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_{l+1}$, interpolate polynomials $f, g, \{f_j, g_j\}_{j\in[n]}$. It sends to the (ideal) adversary the values $f(i), g(i), \{f_j(i), g_j(i)\}_{P_j^l \in \mathcal{C}_l \setminus \mathcal{I}_l}$ and polynomials f_i, g_i for all $P_i^l \in \mathcal{I}_l$.
2. The functionality samples a random polynomial $h(x)$ of degree at most t in x such that $h(0) = f(0)\cdot g(0)$. It sends to the (ideal) adversary $h(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$. It also receives from the (ideal) adversary polynomials $h_i(x)$ for $P_i^{l+1} \in \mathcal{I}_{l+1}$. In addition it also samples for each $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, a random degree-$t$ polynomial $h_i(x)$ such that $h_i(0) = h(i)$. It then outputs $\langle \gamma, h(x)\rangle_{l+1}$, $\{\langle h(i), h_i(x)\rangle_{l+1}\}_{i\in[n]}$ to committee $\mathcal{C}_{l+1}$.

Figure 6: Functionality $\mathcal{F}_{\mathsf{MultCDM}}$

The communication complexity of $\Pi_{\mathsf{MultHelper}}$ comes from calling $\mathcal{F}_{\mathsf{DoubleReshare}}$ on the multiplied shares, which costs $\mathcal{O}(n^3 \cdot \mathsf{comm}_{\mathsf{VSS}})$. Notice that even though $\mathcal{F}_{\mathsf{DoubleReshare}}$ is called in the end of Step 2 for committee $\mathcal{C}_l$, the random sharing done by the protocol realizing $\mathcal{F}_{\mathsf{DoubleReshare}}$ can be started already at Step 1 concurrently with the protocol realizing $\mathcal{F}_{\mathsf{VSS}}$. Hence, both $\mathcal{C}_l$ and $\mathcal{C}_{l+1}$ only need to execute $\delta = n_{\mathsf{VSS}} + 1$ rounds. We formally state security for protocol $\Pi_{\mathsf{MultHelper}}$ in Lemma 4, which we prove in Appendix B.3.

Lemma 4. *For $n, t \in \mathbb{N}$, the protocol $\Pi_{\mathsf{MultHelper}}$ (in Figure 7) realizes the functionality $\mathcal{F}_{\mathsf{MultHelper}}$ (in Figure 5) in the $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{DoubleReshare}})$-hybrid model, with perfect security for $t < n/3$.*

In Protocol Π_{MultCDM} (Figure 8), as each P_i for $i \in [n]$ calls $\mathcal{F}_{\mathsf{MultHelper}}$ as a helper party, the cost per multiplication is $\mathcal{O}(n^4 \cdot \mathsf{comm}_{\mathsf{VSS}})$. We formally state the security of this protocol in Lemma 5, which we prove in Appendix B.4.

Public Inputs: Helper Party P_h^l, Input Committee $\mathcal{C}_l$, Output Committee $\mathcal{C}_{l+1}$
Secret Inputs: The helper P_h^l holds α, β. In addition, every party P_i^l holds shares $f(i), g(i)$ as their shares of α, β respectively denoted as $\langle \alpha, f(x) \rangle_l, \langle \beta, g(x) \rangle_l$.

Committee $\mathcal{C}_l$:

1. Party P_h^l samples $\alpha_k \leftarrow_\$ \mathbb{F}, \beta_k \leftarrow_\$ \mathbb{F}$ for each $k \in [t]$. Define $f_\alpha(x) = \alpha + \sum_{k=1}^{t} \alpha_k x^k$, $g_\beta(x) = \beta + \sum_{k=1}^{t} \beta_k x^k$. Let $h_{\gamma_0}(x) = f_\alpha(x) \cdot g_\beta(x) = \sum_{k=0}^{2t} \gamma_k x^k$.
   P_h^l sends $\{(\mathsf{Share}, sid, f_{\alpha_k})\}_{k\in[t]}$, $\{(\mathsf{Share}, sid, f_{\beta_k})\}_{k\in[t]}$, and $\{(\mathsf{Share}, sid, f_{\gamma_k})\}_{k\in[0,2t]}$ to $4t$ instances of $\mathcal{F}_{\mathsf{VSS}}$ as dealer to distribute shares $\langle \alpha_k, f_{\alpha_k}(x) \rangle_l, \langle \beta_k, g_{\beta_k}(x) \rangle_l$ for each $k \in [t]$ and $\langle \gamma_k, h_{\gamma_k}(x) \rangle_l$ for each $k \in [0, 2t]$.
2. For each $i \in [n]$, reveal the following combination of shares to Party P_i^{l+1}:

$$
\begin{aligned}
(\hat{f}_\alpha(i))_{P_i^{l+1}} &\Longleftarrow \langle \alpha, f(x) \rangle_l + i\langle \alpha_1, f_{\alpha_1}(x) \rangle_l + \cdots + i^t \langle \alpha_t, f_{\alpha_t}(x) \rangle_l \\
(\hat{g}_\beta(i))_{P_i^{l+1}} &\Longleftarrow \langle \beta, g(x) \rangle_l + i\langle \beta_1, g_{\beta_1}(x) \rangle_l + \cdots + i^t \langle \beta_t, g_{\beta_t}(x) \rangle_l \\
(\hat{h}_\gamma(i))_{P_i^{l+1}} &\Longleftarrow \langle \gamma_0, h_{\gamma_0}(x) \rangle_l + i\langle \gamma_1, h_{\gamma_1}(x) \rangle_l + \cdots + i^{2t} \langle \gamma_{2t}, h_{\gamma_{2t}}(x) \rangle_l
\end{aligned}
$$

   In addition, each party $P_i^l \in \mathcal{C}_l$ sends $(\mathsf{DoubleShare}, sid, h_{\gamma_0}(i))$ to $\mathcal{F}_{\mathsf{DoubleReshare}}$ to obtain $[\![\gamma_0, h(x), \{h_i(x)\}_{i\in[n]}]\!]_{l+1}$

Committee $\mathcal{C}_{l+1}$:

1. Each party P_i^{l+1} in Committee $\mathcal{C}_{l+1}$ checks if $\hat{f}(i) \cdot \hat{g}(i) = \hat{h}(i)$. Otherwise broadcasts a complaint.
2. For each $i \in [n]$, if P_i^{l+1} has raised a complaint, make public the following values:

$$
\begin{aligned}
(\hat{f}_\alpha(i))_{P_i^{l+1}} &\Longleftarrow \langle \alpha, f(x) \rangle_l + i\langle \alpha_1, f_{\alpha_1}(x) \rangle_l + \cdots + i^t \langle \alpha_t, f_{\alpha_t}(x) \rangle_l \\
(\hat{g}_\beta(i))_{P_i^{l+1}} &\Longleftarrow \langle \beta, g(x) \rangle_l + i\langle \beta_1, g_{\beta_1}(x) \rangle_l + \cdots + i^t \langle \beta_t, g_{\beta_t}(x) \rangle_l \\
(\hat{h}_\gamma(i))_{P_i^{l+1}} &\Longleftarrow \langle \gamma_0, h_{\gamma_0}(x) \rangle_l + i\langle \gamma_1, h_{\gamma_1}(x) \rangle_l + \cdots + i^{2t} \langle \gamma_{2t}, h_{\gamma_{2t}}(x) \rangle_l
\end{aligned}
$$

   Note that $\hat{f}_\alpha(i), \hat{g}_\beta(i), \hat{h}_\gamma(i)$ are to be revealed only for $i \in [n]$ with a registered complaint. Once the complaints are made, this can be achieved by having each P_i^l secret share their share of $\hat{f}_\alpha(i), \hat{g}_\beta(i), \hat{h}_\gamma(i)$ to $\mathcal{C}_{l+1}$, such that parties in $\mathcal{C}_{l+1}$ selectively reveal these shares only for $i \in [n]$ with a complaint.
3. For each $i \in [n]$ with a complaint, check if $\hat{f}(i) \cdot \hat{g}(i) = \hat{h}(i)$. If the equality check succeeds for all complaints, then the parties store $\langle \gamma_0, h(x) \rangle_{l+1}$ as shares of $\alpha\beta$.

Figure 7: Protocol $\Pi_{\mathsf{MultHelper}}$ in the $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{DoubleReshare}})$-hybrid model.

Lemma 5. *For $n, t \in \mathbb{N}$, the protocol Π_{MultCDM} (in Figure 8) realizes the functionality $\mathcal{F}_{\mathsf{MultCDM}}$ (in Figure 6) in the $(\mathcal{F}_{\mathsf{DoubleReshare}}, \mathcal{F}_{\mathsf{MultHelper}})$-hybrid model, with perfect security for $t < n/3$.*

Secret Inputs: $[\![\alpha, f(x), (f_i(x))_{i\in[n]}]\!]_l, [\![\beta, g(x), (g_i(x))_{i\in[n]}]\!]_l$
Public Inputs: A field $\mathbb{F}$ and n distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. Input Committee $\mathcal{C}_l$.

Committee $\mathcal{C}_l$:

1. For each $i \in [n]$, P_i^l sends $(\mathsf{MultHelpShare}, sid, f(i) \cdot g(i))$ and every P_j^l sends $(\mathsf{MultHelp}, sid, f_i(j), g_i(j))$ to $\mathcal{F}_{\mathsf{MultHelper}}$.

Committee $\mathcal{C}_{l+1}$:

1. Each P_k^{l+1} receives the share $h_i(k)$ for $\langle f(i) \cdot g(i), h_i(x) \rangle_{l+1}$ from $\mathcal{F}_{\mathsf{MultHelper}}$ where P_i^l was the helper which was not disqualified. It uses these shares, and locally computes by Lagrange interpolation a sharing for $\langle \alpha\beta, h(x) \rangle$.
2. Each P_i^{l+1} sends $(\mathsf{DoubleShare}, sid, h(i))$ to $\mathcal{F}_{\mathsf{DoubleReshare}}$ to distribute $[\![\alpha\beta, h'(x), ((h'_i)(x))_{i\in[n]}]\!]_{l+2}$.

Figure 8: Protocol Π_{MultCDM} in the $(\mathcal{F}_{\mathsf{DoubleReshare}}, \mathcal{F}_{\mathsf{MultHelper}})$-hybrid model

3.4 Protocol for Perfectly Secure MPC

We now sketch a protocol in Figure 9 for securely computing any given function f by securely evaluating a layered arithmetic circuit C computing the function. Suppose each party $P_0^i, i \in [n]$ in

the input committee has $z_i \in F$ as input, and each party in the output layer (specified later) wants to compute $f(z_1, \cdots, z_n)$. The secure computation of f proceeds in three phases: an input sharing phase, a circuit evaluation phase and an output reconstruction phase.

Theorem 6. *Let f be an n-party functionality computed by a layered arithmetic circuit C over a finite field $\mathbb{F}$ with multiplicative depth d. Then the protocol $\Pi_{\mathsf{IT\text{-}MPC}}$ (in Figure 9) is a $(n, t, n_{\mathsf{VSS}}+1, 2d+2)$-layered MPC protocol in the $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{Reshare}}, \mathcal{F}_{\mathsf{MultCDM}})$-hybrid model computing the functionality f with perfect security for $n = 3t+1$ with communication complexity $\mathcal{O}(n^4 \cdot \mathsf{Comm}_{\mathsf{VSS}})$, where n_{VSS} and $\mathsf{Comm}_{\mathsf{VSS}}$ are respectively the round complexity and communication complexity of the protocol realizing $\mathcal{F}_{\mathsf{VSS}}$.*

Proof (Sketch). We argue the security of the protocol in $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{Reshare}}, \mathcal{F}_{\mathsf{MultCDM}})$-hybrid model and appeal to sequential composition to prove security of the entire protocol. This is possible since the protocol during the input sharing stage essentially uses execution of (parallel) $\mathcal{F}_{\mathsf{VSS}}$ by parties in $\mathcal{C}_1$ and then parallel invocations of $\mathcal{F}_{\mathsf{Reshare}}$ by parties in $\mathcal{C}_1$ to $\mathcal{C}_2$, and then during the circuit evaluation stage, for the addition gate or multiplication-by-constant the only parallel invocations of $\mathcal{F}_{\mathsf{Reshare}}$ from $\mathcal{C}_{2i}$ to $\mathcal{C}_{2i+2}$ for each $1 \leq i \leq D$ and for the multiplication the only parallel invocations of $\mathcal{F}_{\mathsf{MultCDM}}$ from $\mathcal{C}_{2i}$ to $\mathcal{C}_{2i+2}$ for each $1 \leq i \leq D$. Thus, the protocol amounts to sequential invocations of secure protocols implementing parallel VSS share, VSS reshare, addition and multiplication functionalities.

Consider the protocol in $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{Reshare}}, \mathcal{F}_{\mathsf{MultCDM}})$-hybrid model obtained from the protocol in Figure 9 by replacing executions of VSS sharing, VSS resharing and multiplication without helper party with $\mathcal{F}_{\mathsf{VSS}}$, $\mathcal{F}_{\mathsf{Reshare}}$ and $\mathcal{F}_{\mathsf{MultCDM}}$, respectively. First, all inputs are secret shared to $\mathcal{C}_1$. During the circuit evaluation stage, the protocol ensures that the value on the output wire of every gate is freshly and randomly secret shared before being consumed by gates in the next layer of the circuit which is computed by the next committee. Thus, for each $1 \leq i \leq D$, parties in the committee $\mathcal{C}_{2i}$ hold fresh VSS shares of the values on the input wires of gates. Proceeding in this manner, it can be seen that the parties in committee $\mathcal{C}_{2D+2}$ hold the value on the output wire of the circuit. Then the parties execute a reconstruction protocol to recover the value on the output wire from these shares.

Secret Inputs: Each input client P_i^1 in $\mathcal{C}_1$ has an input $x_i \in \mathbb{F}$.
Public Inputs: An arithmetic circuit $\mathbf{C}$ over a field $\mathbb{F}$.

Input Sharing Stage:

1. For each $i \in [n]$, P_i^1 sends $(\mathsf{Share}, sid, f_{x_i})$ to $\mathcal{F}_{\mathsf{VSS}}$ to distribute $\langle x_i, f_{x_i} \rangle_1$.
2. For each $i, j \in [n]$, P_i^1 sends $(\mathsf{DoubleShare}, sid, f_{x_j}(i))$ to $\mathcal{F}_{\mathsf{DoubleReshare}}$ to distribute $\left[\!\left[x_j, f'_{x_j}(x), \{f'_{x_{j,k}}(x)\}_{k \in [n]} \right]\!\right]_2$

Circuit Evaluation Stage:

Invariant: For $1 \leq i \leq D$, where D is the depth of the layered circuit, parties in committee $\mathcal{C}_{2i}$ hold refreshed shares of the values on the input wires of any gate in layer i of circuit C. For any gate G_j, they proceed as follows:

1. G_j is an addition gate or a multiplication-by-constant gate:
   (a) Parties in $\mathcal{C}_{2i}$ locally compute the output sharing $[\![o_j, g_j(x)]\!]_i$ by adding their shares of the corresponding inputs.
   (b) Execute a resharing of these by calling $\mathcal{F}_{\mathsf{DoubleReshare}}$ on the sharings in $\mathcal{C}_{2i}$ and again in $\mathcal{C}_{2i+1}$ such that parties in $\mathcal{C}_{2i+2}$ receive shares $[\![o_j, g'_j(x)]\!]_{2i+2}$.
2. G_j is a multiplication gate: Parties in $\mathcal{C}_{2i}$ call $\mathcal{F}_{\mathsf{MultCDM}}$ to distribute output shares $[\![o_j, g_j(x)]\!]_{2i+2}$.

Output Reconstruction Phase:

1. Output Clients in committee $\mathcal{C}_{2D+2}$ receive refreshed shares of the output wires. They execute a reconstruction protocol to recover the value on the output wire from these shares.

Figure 9: Protocol $\Pi_{\mathsf{IT\text{-}MPC}}$ in the $(\mathcal{F}_{\mathsf{VSS}}, \mathcal{F}_{\mathsf{Reshare}}, \mathcal{F}_{\mathsf{MultCDM}})$-hybrid model.

3.5 Improved Number of Committees via Optimistic Execution

In Theorem 6 we state that $\Pi_{\mathsf{IT\text{-}MPC}}$ is a $(n, t, n_{\mathsf{VSS}}+1, 2d+2)$-layered MPC protocol, meaning that it requires $2d+2$ committees each executing $n_{\mathsf{VSS}}+1$ in order to compute a circuit of multiplicative depth

d, where n_{VSS} is the round complexity of the black-box VSS protocol used as building block. This is because Protocol $\Pi_{\mathsf{IT\text{-}MPC}}$ requires two committees $\mathcal{C}_l, \mathcal{C}_{l+1}$ to each execute $n_{\mathsf{VSS}}+1$ rounds in order to compute a multiplication in the worst case. However, this is due to the complaint resolution done by $\mathcal{C}_{l+1}$ in $\Pi_{\mathsf{MultHelper}}$ in case there are complaints caused by invalid shares of the multiplication result received from $\mathcal{C}_l$. A simple optimization can be done in $\Pi_{\mathsf{MultHelper}}$ in order to use only 1 committee executing $\delta = n_{\mathsf{VSS}} + 1$ rounds per multiplication when there are no complaints. When there are no complaints throughout an execution, this optimization reduces the total number of committees used in $\Pi_{\mathsf{IT\text{-}MPC}}$ to $d + 2$.

Basically, the optimization involves having each party in $\mathcal{C}_{l+1}$ optimistically start re-sharing the multiplication result (step 2 of Π_{MultCDM}) when their check passes and they do not broadcast a complaint in step 1 of $\Pi_{\mathsf{MultHelper}}$. If any party broadcasts a complaint, all parties ignore the optimistic execution attempt and revert to the complain resolution steps of $\Pi_{\mathsf{MultHelper}}$. Namely, at the first round of $\Pi_{\mathsf{MultHelper}}$ executed by $\mathcal{C}_{l+1}$, parties for which the check succeeds (*i.e.* parties who do not complain), optimistically move on to step 2 of Π_{MultCDM} and start re-sharing the result of the multiplication. In the next round, if a complaint was broadcast by a party whose check in $\Pi_{\mathsf{MultHelper}}$ failed, all parties execute the remainder of $\Pi_{\mathsf{MultHelper}}$ in order to resolve the complaint (*i.e.* parties who started an optimistic execution of step 2 of Π_{MultCDM} revert back to step 2 of $\mathcal{C}_{l+1}$ in $\Pi_{\mathsf{MultHelper}}$ and parties who broadcast complaints ignore the messages received in the previous round except for complaints). Otherwise, if no complaints were broadcast, all parties continue the optimistic execution of Π_{MultCDM}.

4 Statistically Secure Rumors MPC Protocol

We now present a statistically secure Rumors MPC protocol, which is a Multi-Layered of the protocol shown in the seminal work of Rabin Ben-Or [RB89] that can be executed by a sequence of committees each executing a constant number of rounds. We first start with a conditional future broadcast protocol that can be used to conditionally transfer messages from a committee $\mathcal{C}_l$ to the next committee $\mathcal{C}_{l+1}$. Using this protocol, we construct a Multi-Layered version of the Information Theoretical Signature, Distributed Commitment and VSS protocols of [RB89]. Finally, we leverage the VSS protocol and conditional future broadcast to construct a multiplication protocol based on [GRR98] that allows $\mathcal{C}_l$ holding shares of the inputs to start a secure multiplication in its last round, such that the multiplication is finished by $\mathcal{C}_{l+1}$ in its first 12 rounds.

4.1 Conditional Future Broadcast

In Figure 10, we define a corruption aware functionality $\mathcal{F}_{\mathsf{FB}}$ for parallel conditional broadcast in the multi-layered setting. In Figure 11, we describe Protocol Π_{FB}, which realizes functionality $\mathcal{F}_{\mathsf{FB}}$. This is a robust secret sharing protocol based on information theoretic MACs, with a publicly verifiable conditional reveal protocol. Essentially, the sender $P_S \in \mathcal{C}_l$ creates robust secret shares of its message that are transferred to parties in $\mathcal{C}_{l+1}$. If these parties decide to reconstruct the message, they run an interactive protocol to check the MACs for each
share and reconstruct the message using valid shares. The protocol allows for both reconstructing a message in public and revealing it to a single party. The communication complexity of Protocol Π_{FB} is $\mathcal{O}(n' \cdot n^2)$ field elements per sender in the sharing phase executed by committee $\mathcal{C}_l$ and $\mathcal{O}(n' \cdot n^2)$ field elements in the reveal phase executed by committee $\mathcal{C}_{l+1}$. The round complexity is 1 round for the sharing phase and 2 rounds for the reveal phase. We formally claim its security in Lemma 7, and prove it in Appendix D.1.

Lemma 7. *Protocol Π^n_{FB} (in Figure 11) realises $\mathcal{F}_{\mathsf{FB}}$ (in Figure 10) with $(\mathsf{negl}(\kappa))$-statistical security where input sender $P_{S,i} \in \mathcal{C}_l$ acting as the Sender for the i'th parallel instance Π^i_{FB}.*

4.2 Information Theoretic Signature

In Figure 12, we describe the functionality $\mathcal{F}_{\mathsf{ITSig}}$, which captures an Information Theoretical Signature (IT-Sig) in the Multi-Layered MPC model. This functionality allows senders $P_S \in \mathcal{C}_l$ to create IT-Sigs on messages towards intermediate receivers $P_{\mathsf{int}} \in \mathcal{C}_{l+1}$ that can be checked by $\mathcal{C}_{l+1}$ or by a single party $P_i^{l+1} \in \mathcal{C}_{l+1}$.

Public Parameters: Sender $P_{S,1}, \ldots, P_{S,n} \in \mathcal{C}_l$, message domain M.
Secret Inputs: $P_{S,i}$ holds messages $(m_{i,1}, \ldots, m_{i,n'}) \in M$ to be broadcast to $\mathcal{C}_{l+1}$ or sent to a party $P_j^{l'} \in \mathcal{C}_{l'}$ for $l' \in \{l+1, l+2\}$. In addition, there is a public value $(L_1, \ldots, L_n, \mathcal{P}_\mathcal{R})$ agreed upon by honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ where:
- $L_i : M^{n'} \mapsto M$ for $i \in [n]$ is a linear operator.
- $\mathcal{P}_\mathcal{R} \in \mathcal{C}_{l+1} \cup \bigcup_{i=1}^{n} \{P_i^{l+1}\} \cup \bigcup_{i=1}^{n} \{P_i^{l+2}\}$ is the recipient of the message.

Additional Inputs to the functionality: Corruption sets $\mathcal{I}_{l'}, \mathcal{I}_{l+1}, \mathcal{I}_{l+2}$

Share: Upon receiving $(\mathsf{Share}, sid, (m_{i,1}, \ldots, m_{i,n'}))$ from $P_{S,i}$, store $(m_{i,1}, \ldots, m_{i,n'})$ if $P_{S,i} \notin \mathcal{I}_l$.

Broadcast: If each honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, sends $(\mathsf{Broadcast}, sid, L_1, \ldots, L_n, \mathcal{P}_\mathcal{R})$, do the following:
- If $\mathcal{P}_\mathcal{R} = \mathcal{C}_{l+1}$, send $L_j(m_{j,1}, \ldots, m_{j,n'})$ to the (ideal) adversary $\mathcal{S}$ for all j such that $P_{S,j} \notin \mathcal{I}_l$.
- If $\mathcal{P}_\mathcal{R} = P_i^{l'} \in \mathcal{I}_{l'}$, and $P_{S,j} \notin \mathcal{I}_l$, send $L_j(m_{j,1}, \ldots, m_{j,n'})$ for $j \in [n]$ to $\mathcal{S}$.
- If $\mathcal{P}_\mathcal{R} = \mathcal{C}_{l+1}$ or $\mathcal{P}_\mathcal{R} = P_i^{l'} \notin \mathcal{I}_{l'}$, and $P_{S,j} \notin \mathcal{I}_l$, send $L_j(m_{j,1}, \ldots, m_{j,n'})$ for $j \in [n]$ to all parties in $\mathcal{P}_\mathcal{R}$.
- If $\mathcal{P}_\mathcal{R} = \mathcal{C}_{l+1}$ or $\mathcal{P}_\mathcal{R} = P_i^{l'} \notin \mathcal{I}_{l'}$, and $P_{S,j} \in \mathcal{I}_l$, receive from the (ideal) adversary $l_{\mathcal{S},j}$, and send it to all parties in $\mathcal{P}_\mathcal{R}$.

Figure 10: Conditional Future Broadcast functionality $\mathcal{F}_{\mathsf{FB}}$

Secret Inputs: $P_S \in \mathcal{C}_l$ holds messages $(m_1, \ldots, m_{n'}) \in M$
Public Inputs: Share Holders $\{P_1^{l+1}, \cdots, P_n^{l+1}\}$ in Committee $\mathcal{C}_{l+1}$.

Sharing Phase $\mathsf{FB.Share}$:
- **Committee** $\mathcal{C}_l$: Sender P_S computes $(m_{i,1}, \ldots, m_{i,n}) = \mathsf{Sh}(m_i)$ for all $i \in [n']$. It generates keys $K_{i,j,k}$ and computes $M_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m_{i,j})$ for $i \in [n']$,$j,k \in [n]$. For $i \in [n']$, $j \in [n]$, send $(m_{i,j}, \{M_{i,j,1}, \cdots, M_{i,j,n}\}, \{K_{i,1,j}, \cdots, K_{i,n,j}\})$ to P_j^{l+1}.

Broadcast Phase $\mathsf{FB.Broadcast}$:
When $\mathcal{P}_\mathcal{R} = \mathcal{C}_{l+1}$ do the following:
- **Committee** $\mathcal{C}_{l+1}$: Each P_j^{l+1} does the following:
  1. Broadcast $(L(m_{1,j}, \ldots, m_{n',j}), \{M_{i,j,1}, \cdots, M_{i,j,n}\})$ for all $i \in [n']$.
  2. Broadcast $(\{K_{i,1,j}, \cdots, K_{i,n,j}\}_{i \in [n']})$. Accept $L(m_{1,j}, \ldots, m_{n',j})$ iff

$$L(M_{1,j,k}, \ldots, M_{n',j,k}) = \mathsf{MAC}_{L(K_{1,j,k}, \ldots, K_{n',j,k})}(L(m_{1,j}, \ldots, m_{n',j}))$$

  for at least $t+1$ of the keys. Then interpolate over the accepted shares to define a polynomial, and output the constant term. Otherwise output $\perp$.

When $\mathcal{P}_\mathcal{R} = P_j^{l'}$ do the following:
- **Committee** $\mathcal{C}_{l+1}$: Each P_j^{l+1} sends to $P_j^{l'}$, $(L(m_{1,j}, \ldots, m_{n',j}), \{M_{i,j,1}, \cdots, M_{i,j,n}\})$ for all $1 \leq i \leq n'$ and $(\{K_{i,1,j}, \cdots, K_{i,n,j}\})$ $1 \leq i \leq n'$.
- **Committee** $\mathcal{C}_{l'}$: $P_j^{l'}$ accepts $(L(m_{1,j}, \ldots, m_{n',j})$ iff $L(M_{1,j,k}, \ldots, M_{n',j,k}) = \mathsf{MAC}_{L(K_{1,j,k}, \ldots, K_{n',j,k})}(L(m_{1,j}, \ldots, m_{n',j}))$ for at least $t+1$ of the keys. Then it interpolates over the accepted shares to define a polynomial, and output the constant term. Otherwise it outputs $\perp$.

Figure 11: Protocol Π_{FB}

We build on the YOSO [GHK+21] version of the IT-Sig protocol of [RB89] to realize functionality. We describe Protocol Π_{ITSig} in Figure 13. In this protocol, the sender $P_S \in \mathcal{C}_l$ creates κ MAC tags on each message with fresh keys for every party in $\mathcal{C}_{l+1}$ and sends them the corresponding keys, while sending the messages and MAC tags to the intermediate receiver $P_{\mathsf{int}} \in \mathcal{C}_{l+1}$. Additionally, P_S conditionally broadcasts all messages and keys towards $\mathcal{C}_{l+1}$. Next, parties in $\mathcal{C}_{l+1}$ check half of the MAC tags at random and the (honest) $\mathcal{C}_{l+1}$ triggers a reveal for the message previously broadcast by P_S if a check fails for that message. Otherwise, in the later reveal phase, P_{int} broadcasts the message it claims to have received to $\mathcal{C}_{l+1}$, who checks the validity of the message using the MAC keys that were not used in the previous check. Notice that parties can also reveal and check linear combinations of IT-Sigs on messages *from the same sender* given that the MAC keys are correlated (*i.e.* $K_i = (A, B_i)$ for a fixed A and fresh B_i).

Protocol Π_{ITSig} has communication complexity of $\mathcal{O}(n' \cdot \kappa \cdot n^3)$ field elements for the Setup phase and $\mathcal{O}(n' \cdot \kappa \cdot n)$ field elements for the Reveal phase. The round complexity in the Setup phase is of 1 round for senders in $\mathcal{C}_l$ and 5 rounds for parties in $\mathcal{C}_{l+1}$, while the Reveal phase requires 2 rounds in $\mathcal{C}_{l+1}$. We formally state the security of Protocol Π_{ITSig} in Lemma 8, which is proven in Appendix D.2.

Public Parameters: Sender $P_S \in \mathcal{C}_l$, an intermediate receiver $P_{\mathsf{int}} \in \mathcal{C}_{l+1}$.
Secret Inputs: P_S holds messages $(m_1, \ldots, m_{n'}) \in M$ to be broadcast to $\mathcal{C}_{l+1}$ or sent to a party $P_i^{l+2} \in \mathcal{C}_{l+2}$. In addition, there is a public value $(L, \mathcal{P}_\mathcal{R})$ agreed upon by honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ where:
- $L : M^{n'} \mapsto M$ is a linear operator.
- $\mathcal{P}_\mathcal{R} \in \mathcal{C}_{l+1} \cup \bigcup_{i=1}^{n}\{P_i^{l+1}\} \cup \bigcup_{i=1}^{n}\{P_i^{l+2}\}$ is the recipient of the message.

Setup: Upon receiving $(\mathsf{SetupSig}, sid, (m_1, \ldots, m_{n'}))$ from P_S, store $(m_1^*, \ldots, m_{n'}^*) = (m_1, \ldots, m_{n'})$ if $P_S \notin \mathcal{I}_l$. Additionally, send $(m_1^*, \ldots, m_{n'}^*)$ to $\mathcal{S}$ when $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$. Otherwise, receive $(m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$ from the (ideal) adversary $\mathcal{S}$ and set $(m_1^*, \ldots, m_{n'}^*) = (m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$.
Reveal: If P_{int} sends $(\mathsf{RevealSig}, sid, L, \mathcal{P}_\mathcal{R})$, then:
- If P_{int} is honest, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_\mathcal{R}$. If $\mathcal{P}_\mathcal{R} = P_i^{l+1} \in \mathcal{I}_{l+1}$, then send $L(m_1^*, \ldots, m_{n'}^*)$ to the (ideal) adversary $\mathcal{S}$.
- If P_{int} is corrupt:
  - P_S is honest: Receive boolean $\mathsf{reveal} \in \{0, 1\}$ from the (ideal) adversary. If $\mathsf{reveal} = 1$, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_\mathcal{R}$, else send $\perp$.
  - P_S is corrupt: Receive $l_\mathcal{S}^*$ from the (ideal) adversary $\mathcal{S}$ and send $l_\mathcal{S}^*$ to all parties in $\mathcal{P}_\mathcal{R}$.

Figure 12: IT-Sig Functionality $\mathcal{F}_{\mathsf{ITSig}}$

Secret Inputs: Sender $P_S \in \mathcal{C}_l$ holds input $(m_1, \ldots, m_{n'}) \in M$.
Public Inputs: Verifiers $P_1^{l+1}, \cdots, P_n^{l+1} \in \mathcal{C}_{l+1}$, Intermediate Receiver $P_{\mathsf{int}} \in \mathcal{C}_{l+1}$.

Setup Phase $\mathsf{ITSig.Setup}$:
- **Committee $\mathcal{C}_l$:** P_S generates keys $K_{i,j,k}, i \in [n'], j \in [n], k \in [\kappa]$, computes $M_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m_i)$ and sends $\{K_{i,j,k}\}_{i\in[n'],k\in[k]}$ to P_j^{l+1} and $(m_1, \ldots, m_{n'}), \{M_{1,j,k}, \ldots, M_{n',j,k}\}_{j\in[n],k\in[\kappa])}$ to P_{int}. It also sends $(\mathsf{Setup}, sid, (m_1, \ldots, m_{n'}))$ and $(\mathsf{Setup}, sid, (K_{1,j,k}, \ldots, K_{n',j,k}))$ to $\mathcal{F}_{\mathsf{FB}}$ for $j \in [n], k \in [\kappa]$.
- **Committee $\mathcal{C}_{l+1}$:**
  1. Each P_j^{l+1} chooses half the indices at random, say INX_j, and broadcasts $K_{i,j,k}$ for $i \in [n'], k \in INX_j$.
  2. Parties in $\mathcal{C}_{l+1}$ sends $(\mathsf{Broadcast}, sid, 1, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{FB}}$, for all $i \in [n'], j \in [n]$, and for all $k \in INX_j$ to reveal $\bar{K}_{i,j,k}$.
  3. If there exists indices j and k for which $\mathsf{MAC}_{\bar{K}_{i,j,k}}(m_i) \neq M_{i,j,k}$, then the receiver P_{int} complains and all parties P_i^{l+1} send $(\mathsf{Broadcast}, sid, L, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{FB}}$ to reveal $L(\bar{m}_1, \ldots, \bar{m}_{n'})$ and outputs $L(\bar{m}_1, \ldots, \bar{m}_{n'})$.

Reveal Phase $\mathsf{ITSig.Reveal}$: If P_{int} forced a reveal in $\mathsf{ITSig.Setup}$, then output the same message $L(\bar{m}_1, \ldots, \bar{m}_{n'})$. Else:
- When $\mathcal{P}_\mathcal{R} = \mathcal{C}_{l+1}$ do the following:
  - **Committee $\mathcal{C}_{l+1}$:**
    1. P_{int} broadcasts $(L(m_1, \ldots, m_{n'}), \{L(M_{1,j,k}, \ldots, M_{n',j,k})\}_{j\in[n],k\notin INX_j)}$.
    2. Set the number of votes for $L(m_1, \ldots, m_n)$ to be number of j's for which $L(\bar{K}_{1,j,k}, \ldots, \bar{K}_{n',j,k}) \neq L(K_{1,j,k}, \ldots, K_{n',j,k})$, for some $k \in INX_j$. For all other j's, P_j^{l+1} broadcasts $L(K_{1,j,k}, \ldots, K_{n',j,k})$ for $k \notin INX_j$. If $\mathsf{MAC}_{L(K_{1,j,k},\ldots,K_{n',j,k})}(L(m_1, \ldots, m_{n'})) = L(M_{1,j,k}, \ldots, M_{n',j,k})$ for any one of the recovered values, then increment the vote by 1. If there are $t+1$ votes, output $L(m_1, \ldots, m_{n'})$.
- When $\mathcal{P}_\mathcal{R} = P_r^{l'}$ do the following:
  - P_{int} sends $(L(m_1, \ldots, m_{n'}), \{L(M_{1,j,k}, \ldots, M_{n',j,k})\}_{j\in[n],k\notin INX_j)}$ to $P_r^{l'}$.
  - Set the number of votes for $L(m_1, \ldots, m_n)$ to be number of j's for which $L(\bar{K}_{1,j,k}, \ldots, \bar{K}_{n',j,k}) \neq L(K_{1,j,k}, \ldots, K_{n',j,k})$, for some $k \in INX_j$. For all other j's, P_j^{l+1} sends $L(K_{1,j,k}, \ldots, K_{n',j,k})$ for $k \notin INX_j$ to $P_r^{l'}$. If $\mathsf{MAC}_{L(K_{1,j,k},\ldots,K_{n',j,k})}(L(m_1, \ldots, m_{n'})) = L(M_{1,j,k}, \ldots, M_{n',j,k})$ for any one of the recovered values, then increment the vote by 1. If there are $t+1$ votes, output $L(m_1, \ldots, m_{n'})$.

Figure 13: Protocol Π_{ITSig} in the $\mathcal{F}_{\mathsf{FB}}$-hybrid model.

Lemma 8. *Protocol Π_{ITSig} (in Figure 13) realizes the functionality $\mathcal{F}_{\mathsf{ITSig}}$ (in Figure 12) with $(\mathsf{negl}(\kappa))$-statistical security in the $\mathcal{F}_{\mathsf{FB}}$-hybrid model.*

4.3 Distributed Commitment

In Figure 14, we describe the functionality $\mathcal{F}_{\mathsf{DistCommit}}$, which captures a Distributed Commitment in the Multi-Layered MPC model. This functionality allows a committer $P_S \in \mathcal{C}_l$ to create a distributed commitment that can be opened by parties in $\mathcal{C}_{l+1}$ (if P_S was honest).

In Figure 15, we describe Protocol $\Pi_{\mathsf{DistCommit}}$, which realizes $\mathcal{F}_{\mathsf{DistCommit}}$. Notice that this protocol can be obtained from $\mathcal{F}_{\mathsf{ITSig}}$ in a straightforward manner. Basically, the sender shares their message and sets up IT-Sigs on each share towards $\mathcal{C}_{l+1}$. When the distributed commitment is opened, parties in $\mathcal{C}_{l+1}$ can check the IT-Sigs to determine which shares are valid and reconstruct the original message (or output $\perp$ if reconstruction fails for lack of valid shares). Notice that parties can also reveal linear combinations of distributed commitments *from the same sender*, since the underlying IT-Sigs allow for that.

Public Parameters: Committer $P_S \in \mathcal{C}_l$.
Secret Inputs: P_S holds messages $(m_1, \ldots, m_{n'}) \in M$ to be broadcast to $\mathcal{C}_{l+1}$ or sent to a party $P_i^{l+2} \in \mathcal{C}_{l+2}$. In addition, there is a public value $(L, \mathcal{P}_\mathcal{R})$ agreed upon by honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ where:
- $L : M^{n'} \mapsto M$ is a linear operator.
- $\mathcal{P}_\mathcal{R} \in \mathcal{C}_{l+1} \cup \bigcup_{i=1}^{n}\{P_i^{l+1}\} \cup \bigcup_{i=1}^{n}\{P_i^{l+2}\}$ is the recipient of the message.

Setup: Upon receiving $(\mathsf{SetupDC}, sid, (m_1, \ldots, m_{n'}))$ from P_S, store $(m_1^*, \ldots, m_{n'}^*) = (m_1, \ldots, m_{n'})$ if $P_S \notin \mathcal{I}_l$. Otherwise, receive $(m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$ from the (ideal) adversary $\mathcal{S}$ and set $(m_1^*, \ldots, m_{n'}^*) = (m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$.
Reveal: If all honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ send $(\mathsf{RevealDC}, sid, L, \mathcal{P}_\mathcal{R})$, then:
- If $P_S \notin \mathcal{I}_0$, then send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_\mathcal{R}$. If $\mathcal{P}_\mathcal{R} = P_i^{l'} \in \mathcal{I}_{l'}$ for $l' = l+1$ or $l+2$, then send $L(m_1^*, \ldots, m_{n'}^*)$ to the (ideal) adversary $\mathcal{S}$.
- If $P_S \in \mathcal{I}_0$, then receive from (ideal) adversary $\mathcal{S}$ a boolean value $\mathsf{open} \in \{0, 1\}$. If $\mathsf{open} = 1$, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_\mathcal{R}$, else deliver $\perp$.

Figure 14: Distributed Commitment Functionality $\mathcal{F}_{\mathsf{DistCommit}}$

Protocol $\Pi_{\mathsf{DistCommit}}$ has communication complexity of $\mathcal{O}(n' \cdot \kappa \cdot n^4)$ field elements in the Setup phase and $\mathcal{O}(n' \cdot \kappa \cdot n^2)$ field elements in the reveal phase. The round complexity for the Setup phase is 1 round in $\mathcal{C}_l$ and 5 rounds in $\mathcal{C}_{l+1}$, while the reveal phase requires 2 rounds.

Secret Inputs: Committer $P_S \in \mathcal{C}_l$ holds input $(m_1, \ldots, m_{n'}) \in M$.
Public Inputs: Share holders and verifiers, $P_1^{l+1}, \cdots, P_n^{l+1} \in \mathcal{C}_{l+1}$

Setup Phase DistCommit.Setup**:**
- **Committee** $\mathcal{C}_l$**:** For all $1 \leq i \leq n'$, Committer C computes $\mathsf{Sh}(m_i)$, say $(m_{i,1}, \cdots, m_{i,n})$ for $n \leq 2t+1$. It then sends $(\mathsf{SetupSig}, sid, (m_{1,j}, \ldots, m_{n',j}))$ to $\mathcal{F}_{\mathsf{ITSig}}$ with P_j^{l+1} as the intermediate receiver.

Reveal Phase DistCommit.Reveal**:**
- **Committee** $\mathcal{C}_{l+1}$**:** For all $i = 1, \cdots, n$, send $(\mathsf{RevealSig}, sid, L, \mathcal{P}_\mathcal{R})$ to $\mathcal{F}_{\mathsf{ITSig}}$ to reveal $l_j = L(m_{1,j}, \ldots, m_{n',j})$. Interpolate through valid l_j to obtain a degree-t polynomial. Output it's constant term if it succeeds, else output $\perp$.

Figure 15: Protocol $\Pi_{\mathsf{DistCommit}}$ in the $\mathcal{F}_{\mathsf{ITSig}}$-hybrid model.

Lemma 9. *Protocol $\Pi_{\mathsf{DistCommit}}$ (in Figure 15) realizes the functionality $\mathcal{F}_{\mathsf{DistCommit}}$ (in Figure 14) with $(\mathsf{negl}(\kappa))$-statistical security in the $\mathcal{F}_{\mathsf{ITSig}}$-hybrid model.*

The proof is presented in Appendix D.3.

4.4 Verifiable Secret Sharing

In Figure 16, we describe the functionality that captures a statistically secure parallel linearly homomorphic VSS in the Multi-Layered MPC model. This functionality allows for dealers in $\mathcal{C}_l$ to share secrets that can be reconstructed by parties in $\mathcal{C}_{l+1}$. Moreover, it allows for revealing linear combinations of secrets shared by different dealers, which was not the case in our previous primitives.

In Figure 17, we describe Protocol Π_{StatVSS}, which realizes $\mathcal{F}_{\mathsf{StatVSS}}$. This protocol works by having each dealer in $\mathcal{C}_l$ set up distributed commitments to coefficients of the polynomials defining shares of their messages and shares of a random value towards parties in $\mathcal{C}_{l+1}$, while directly sending a share to each of these parties. Next, parties in $\mathcal{C}_{l+1}$ also set up distributed commitments to the

Public Parameters: Dealers $D_1, \ldots, D_n \in \mathcal{C}_l$, Shareholders $P_1^{l+1}, \cdots, P_n^{l+1} \in \mathcal{C}_{l+1}$.
Secret Inputs: Each D_i for $1 \leq i \leq n$ holds n' degree-t polynomial $f_{i,j}(x)$ such that $f_{i,j}(0) = m_{i,j}$. In addition, there is a public value $(L, L_1, \ldots, L_n, \mathcal{P}_\mathcal{R})$ agreed upon by honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ where:
- $L : M^n \mapsto M$ is a linear operator.
- $L_i : M^{n'} \mapsto M$ is a linear operator for all $1 \leq i \leq n$.
- $\mathcal{P}_\mathcal{R} \in \mathcal{C}_{l+1} \cup \bigcup_{i=1}^n \{P_i^{l+1}\} \cup \bigcup_{i=1}^n \{P_i^{l+2}\}$ is the recipient of the message.

Share: Upon receiving $(\mathsf{VSSShare}, sid, f_{i,1}, \ldots, f_{i,n'})$, from $D_i \notin \mathcal{I}_l$, store the polynomial $f^*_{i,j} = f_{i,j}$ for $1 \leq j \leq n'$, else receive a degree-t polynomial $f_{\mathcal{S},i,j}$ from the (ideal) adversary and set $f^*_{i,j} = f_{\mathcal{S},i,j}$ for $1 \leq j \leq n'$ and all valid $f_{\mathcal{S},i,j}$. For any invalid polynomials $f_{\mathcal{S},i,j}$ sent by $\mathcal{S}$, set $f^*_{i,j} = 0$. Additionally, if dealer $D_i \notin \mathcal{I}_l$, then send $f_{i,j}(k)$ to $\mathcal{S}$ for all $P_k^{l+1} \in \mathcal{I}_{l+1}$.
Reveal: If all $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ send $(\mathsf{VSSReconstruct}, sid, L, L_1, \cdots, L_n, \mathcal{P}_\mathcal{R})$, then send $l^* = L(L_1(f_{1,1}(0), \ldots, f_{1,n'}(0)), \ldots, L_n(f_{n,1}(0), \ldots, f_{n,n'}(0)))$ to all parties in $\mathcal{P}_\mathcal{R}$. If $\mathcal{P}_\mathcal{R} = P_i^{l'} \in \mathcal{I}_{l'}$ for $l' = l+1$ or $l+2$, then send l^* to the (ideal) adversary $\mathcal{S}$.

Figure 16: Verifiable Secret Sharing Functionality $\mathcal{F}_{\mathsf{StatVSS}}$

shares they received from each dealer. Finally, $\mathcal{C}_{l+1}$ opens random linear combinations of all dealers' distributed commitments to shares of their secrets and shares of a random value (which can be computed as a linear combination of the distributed commitments to coefficients), and open the same linear combinations of the distributed commitments from the parties in $\mathcal{C}_{l+1}$ itself. This check allows for establishing that sufficiently many parties in $\mathcal{C}_{l+1}$ have correctly committed to valid shares from all dealers. Since we new have distributed commitments to the shares of secrets of all dealers with the same senders (*i.e.* the parties in $\mathcal{C}_{l+1}$ that each committed to the shares received from all dealers), we can reconstruct linear combinations of these secrets.

Protocol Π_{StatVSS} has communication complexity of $\mathcal{O}(n' \cdot \kappa \cdot n^5)$ field elements in the setup phase and $\mathcal{O}(n' \cdot \kappa \cdot n^3)$ field elements in the reveal phase. The round complexity is of 1 round for $\mathcal{C}_l$ and 7 rounds for $\mathcal{C}_{l+1}$ in the sharing phase, while the reveal phase requires 2 rounds.

Secret Inputs: Dealer D holds a degree-t polynomial f such that $f(0) = s$.
Output: $\langle s, f(x) \rangle_{l+1}$

Sharing Phase VSS.Share**:**
- **Committee $\mathcal{C}_l$:** The Dealer $D \in \mathcal{C}_l$ does the following:
  1. It chooses a random polynomial $r(x)$ of degree t. Let the co-efficients of $f(x)$ and $r(x)$ be f_j, r_j for $0 \leq j \leq t$ respectively. It sends to P_i^{l+1}, the values $s_i = f(i), \rho_i = r(i)$ for all $1 \leq i \leq n$.
  2. It additionally sends $(\mathsf{SetupDC}, sid, f_i)$ and $(\mathsf{SetupDC}, sid, r_i)$ to $\mathcal{F}_{\mathsf{DistCommit}}$ as a committer for all $1 \leq i \leq n$. As they are homomorphic, it results in $DC_D(s_i)$ and $DC_D(\rho_i)$, say through linear operators L_{s_i}, L_{r_i} respectively.
- **Committee $\mathcal{C}_{l+1}$:** : Each $P_i^{l+1} \in \mathcal{C}_{l+1}$ does the following:
  1. Send $(\mathsf{SetupDC}, sid, s_i, \rho_i)$ to $\mathcal{F}_{\mathsf{DistCommitClassical}}$ with $\mathcal{C}_{l+1}$ as the shareholders to share $DC_{P_i^{l+1}}(s_i)$ and $DC_{P_i^{l+1}}(\rho_i)$
  2. After receiving e from $\mathcal{F}_{\mathsf{Beacon}}$, it broadcasts $s_i + e\rho_i$. Additionally, send $(\mathsf{RevealDC}, sid, L_{s_i} + eL_{r_i}, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{DistCommit}}$ to open $DC_D(s_j + e\rho_j)$ and $(\mathsf{RevealDC}, sid, L \leftarrow (1, e), \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{DistCommitClassical}}$ to open $DC_{P_j^{l+1}}(s_j + e\rho_j)$ for all $j \in [n]$ and check consistency.
  3. For any j where the opened values disagree, or the reveal outputs $\perp$, send $(\mathsf{RevealDC}, sid, L_{s_i}, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{DistCommit}}$ to open $DC_D(s_j)$. If it returns $\perp$, disqualify the dealer. Else output $DC_{P_j^{l+1}}(s_j)$.

Reveal Phase VSS.Reveal**:**
- Each P_i^{l+1} sends $(\mathsf{RevealDC}, sid, L_{s_i}, \mathcal{P}_\mathcal{R})$ to $\mathcal{F}_{\mathsf{DistCommitClassical}}$ to open $DC_{P_i^{l+1}}(s_i)$ for all s_i not opened in the sharing phase. Interpolate through $s_i \neq \perp$, to obtain a degree-t polynomial. Output it's constant term.

Figure 17: Protocol Π_{StatVSS} in the $(\mathcal{F}_{\mathsf{DistCommit}}, \mathcal{F}_{\mathsf{DistCommitClassical}}, \mathcal{F}_{\mathsf{Beacon}})$-hybrid model.

Extending Π_{StatVSS} to deal with multiple inputs from the Dealer. We described the protocol (in Figure 17) in the case where the dealer P_D only has one input s to avoid a notational blow-up. However, it is easy to generalize the construction to the case where the dealer P_D has multiple inputs $s_1, \ldots, s_{n'}$. The protocol is simply executed n' times in parallel, but using the *same* instances of

$DC_{P_D}(\cdot)$ and $DC_{P_i^{l+1}}(.)$ for all $i \in [n]$, which allow for an arbitrary n' number of inputs. This ensures homomorphism across the sender's inputs. Furthermore, in each parallel execution with input s_j for $j \in [n']$ the set $\mathcal{I}^{(j)}$ might be different: we consider $\mathcal{I} = \cup_{j\in[n']}\mathcal{I}^{(j)}$ to be the union of all such sets. Again, if $|\mathcal{I}| > t$ then P_D is disqualified.

Extending Π_{StatVSS} to deal with multiple Dealers. We described the protocol (in Figure 17) in the case where there was only one dealer P_D to avoid a notational blow-up. Consider parallel executions with dealer $P_{D,j}$ with input $P_{D,j}$ for $j \in [n]$. However, when invoked in parallel by different dealers the protocol still provides linearity among values dealt by *different* dealers. The homomorphic properties of distributed commitment DC are incorporated in the VSS. Specifically, in the case when two dealers executes VSS while sharing secret with *the same set of shareholders*, these homomorphic properties remains intact. Consider a scenario, for two secrets s_1 and s_2 distributed by two dealers $P_{D,1}$ and $P_{D,2}$, we have $VSS_{P_{D,1}}(s_1) + VSS_{P_{D,2}}(s_2) = VSS(s_1 + s_2)$. Now, notice that the $VSS(s_1 + s_2)$ part is not associated to a specific dealer as it combines sharings of $P_{D,1}$ and $P_{D,2}$. The reason the homomorphism holds across dealers is two folds: first, it leverages the homomorphic properties of distributed commitment DC_{P_i}, which apply only when the same committer is involved; second, the same set of parties P_i act as shareholders in both VSS.

Lemma 10. *Protocol Π_{StatVSS} (in Figure 17) realizes the functionality $\mathcal{F}_{\mathsf{StatVSS}}$ (in Figure 16) with $(\mathsf{negl}(\kappa))$-statistical security in the $(\mathcal{F}_{\mathsf{DistCommit}}, \mathcal{F}_{\mathsf{DistCommitClassical}}, \mathcal{F}_{\mathsf{Beacon}})$-hybrid model.*

The proof is presented in Appendix D.4.

Re-sharing in the Statistically secure case: We realize re-sharing protocol as in the perfectly secure case by calling our statistically secure VSS functionality $\mathcal{F}_{\mathsf{StatVSS}}$ in $\mathcal{C}_{l-1}$ to create enough random shares in $\mathcal{C}_l$ so that parties in $\mathcal{C}_l$ can re-share secrets towards $\mathcal{C}_{l+1}$. See Appendix C.1 for more details.

4.5 Augmented VSS

In Figure 18, we describe Functionality $\mathcal{F}_{\mathsf{AugVSS}}$ for augmented VSS in the Multi-Layered MPC model. This functionality will allow parties in committee $\mathcal{C}_l$ to provide shares-of-shares of secrets to parties in $\mathcal{C}_{l+1}$, which will be useful when porting the multiplication protocol of [GRR98] to the Multi-Layered MPC model.

Public Parameters: Dealers $D_1, \ldots, D_n \in \mathcal{C}_l$, Shareholders $P_1^{l+1}, \cdots, P_n^{l+1} \in \mathcal{C}_{l+1}$.
Secret Inputs: Each D_i for $1 \le i \le n$ holds n' degree-t polynomial $f_{i,j}(x)$ such that $f_{i,j}(0) = m_{i,j}$. In addition, there is a public value $(L, L_1, \ldots, L_n, \mathcal{P}_{\mathcal{R}})$ agreed upon by honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$ where:
- $L : M^n \mapsto M$ is a linear operator.
- $L_i : M^{n'} \mapsto M$ is a linear operator for all $1 \le i \le n$.
- $\mathcal{P}_{\mathcal{R}} \in \mathcal{C}_{l+1} \cup \bigcup_{i=1}^{n}\{P_i^{l+1}\} \cup \bigcup_{i=1}^{n}\{P_i^{l+2}\}$ is the recipient of the message.

Share:
- **Committee $\mathcal{C}_l$:** Upon receiving $(\mathsf{AugVSSShare}, sid, f_{i,1}, \ldots, f_{i,n'})$, from $D_i \notin \mathcal{I}_l$, store the polynomial $f^*_{i,j} = f_{i,j}$ for $1 \le j \le n'$, else receive a degree-t polynomial $f_{\mathcal{S},i,j}$ from the (ideal) adversary and set $f^*_{i,j} = f_{\mathcal{S},i,j}$ for $1 \le j \le n'$ and all valid $f_{\mathcal{S},i,j}$. For any invalid polynomials $f_{\mathcal{S},i,j}$ sent by $\mathcal{S}$, set $f^*_{i,j} = 0$. Let $f^*_{i,j}(x) = a_{i,j,t}x^t + \ldots + a_{i,j,1}x + a_{i,j,0}$ be valid polynomials. Additionally, for each $1 \le i \le n, 1 \le j \le n'$, sample $t+1$ random degree-t polynomials $f_{a_{i,j,k}}$ such that $f_{a_{i,j,k}}(0) = a_{i,j,k}$ for all $1 \le i \le n, 1 \le j \le n', 0 \le k \le t$. If the dealer $P_D \notin \mathcal{I}_l$, then send $\{f_{i,j}(k), f_{a_{i,j,0}}(k), \ldots, f_{a_{i,j,t}}(k)\}$ to $\mathcal{S}$ for all $P_k^{l+1} \in \mathcal{I}_{l+1}$
- **Committee $\mathcal{C}_{l+1}$:** Send $\{f^*_{i,j}(k), f_{a_{i,j,0}}(k), \ldots, f_{a_{i,j,t}}(k)\}$ to party P_k^{l+1}.

Reveal: If all honest $P_i^{l+1} \in \mathcal{C}_{l+1}$ send $(\mathsf{AugVSSReconstruct}, sid, L, L_1, \ldots, L_n, \mathcal{P}_{\mathcal{R}})$, then send $l^* = L(L_1(f_{1,1}(0), \ldots, f_{1,n'}(0)), \ldots, L_n(f_{n,1}(0), \ldots, f_{n,n'}(0)))$ to all parties in $\mathcal{P}_{\mathcal{R}}$. If $\mathcal{P}_{\mathcal{R}} = P_i^{l'} \in \mathcal{I}_{l'}$ for $l' = l+1$ or $l+2$, then send l^* to the (ideal) adversary $\mathcal{S}$.

Figure 18: Augmented VSS Functionality $\mathcal{F}_{\mathsf{AugVSS}}$

Secret Inputs: Dealers $D_i \in \mathcal{C}_l$ for $1 \leq i \leq n$ holds polynomials $f_{i,j}(x)$ for $1 \leq j \leq n'$.
Output: $[\![m_{i,j}, f_{i,j}(x), \{f_{i,j,k}(x)\}_{k\in[n]}]\!]_{l+1,D_i}$

Sharing Phase AugVSS.Share**:**

- **Committee** $\mathcal{C}_l$**:** Let $f_{i,j}(x) = a_{i,j,t}x^t + \ldots + a_{i,j,1}x + a_{i,j,0}$. Each Dealer $D_i \in \mathcal{C}_l$ samples $t+1$ random degree-t polynomials $f_{a_{i,j,k}}$ such that $f_{a_{i,j,k}}(0) = a_{i,j,k}$ for all $0 \leq k \leq t$. It sends $(\mathsf{VSSShare}, sid, f_{a_{i,j,t}}, \ldots, f_{a_{i,j,0}})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ for $0 \leq j \leq n'$ such that parties in $\mathcal{C}_{l+1}$ hold $\{\langle a_{i,j,k}, f_{a_{i,j,k}}(x)\rangle_{l+1}\}_{1\leq i\leq n, 1\leq j\leq n', 0\leq k\leq t}$.
- **Committee** $\mathcal{C}_{l+1}$**:** Let $L_k = (k^t, k^{t-1}, \ldots, k, 1)$. Hence, each party P_k^{l+1} locally computes $\sum_{k=0}^{t} k^i \langle a_{i,j,t}, f_{a_{i,j,t}}(x)\rangle_{l+1}$. Each party $P_i^{l+1} \in \mathcal{C}_{l+1}$ sends $(\mathsf{VSSReconstruct}, sid, L_k, P_k^{l+1})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ to reveal $f_{i,j}(k)$, which is the share of P_k^{l+1} for $m_{i,j}$. Each P_k^{l+1} stores as it's sharing $(f_{i,j}(k), f_{i,j,0}(k), \ldots, f_{i,j,t}(k))$ for $1 \leq i \leq n, 1 \leq j \leq n'$.

Reveal Phase AugVSS.Reveal**:**

- Each P_k^{l+1} sends $(\mathsf{VSSReconstruct}, sid, L, L_1, \ldots, L_n, \mathcal{P}_{\mathcal{R}})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ to reveal $L(L_1(f_{1,1}(0), \ldots, f_{1,n'}(0)), \ldots, L_n(f_{n,1}(0), \ldots, f_{n,n'}(0)))$.

Figure 19: Protocol Π_{AugVSS} in the $\mathcal{F}_{\mathsf{StatVSS}}$-hybrid model.

In Figure 19, we describe Protocol Π_{AugVSS}, which realizes Functionality $\mathcal{F}_{\mathsf{AugVSS}}$. This protocol is a direct application of Functionality $\mathcal{F}_{\mathsf{VSS}}$. The communication complexity of Protocol Π_{AugVSS} is of $\mathcal{O}(n' \cdot \kappa \cdot n^7)$ field elements for the sharing phase and of $\mathcal{O}(n' \cdot \kappa \cdot n^5)$ field elements for the reveal phase. The round complexity for the sharing phase is of 1 round for committee $\mathcal{C}_l$ and 8 rounds for $\mathcal{C}_{l+1}$, while the reveal phase requires 2 rounds.

Lemma 11. *Protocol* Π_{AugVSS} *(in Figure 19) realizes the functionality* $\mathcal{F}_{\mathsf{AugVSS}}$ *(in Figure 18) with* $(\mathsf{negl}(\kappa))$*-statistical security in the* $\mathcal{F}_{\mathsf{StatVSS}}$*-hybrid model.*

The proof follows in a straightforward manner from the use of $\mathcal{F}_{\mathsf{VSS}}$.

Secret Inputs: Party $P \in \mathcal{C}_l$ holds values a, b, c. Parties $P_1^{l+1}, \cdots, P_n^{l+1}$ in committee $\mathcal{C}_{l+1}$ hold sharings $\langle a, f_a(x)\rangle_{l+1}, \langle b, f_b(x)\rangle_{l+1}, \langle c, f_c(x)\rangle_{l+1}$. Additionally, assume that parties in $\mathcal{C}_l$ have access to a random sharing $\langle b', f_{b'}(x)\rangle_l$.
Public Inputs: Shareholders $P_1^{l+1}, \cdots, P_n^{l+1} \in \mathcal{C}_{l+1}$.

Committee $\mathcal{C}_l$**:** Each party P_i^l sends $(\mathsf{VSSReconstruct}, sid, 1, P)$ to $\mathcal{F}_{\mathsf{VSS}}$ to reveal b' to P and sends $(\mathsf{Reshare}, sid, f_{b'}(i))$ to $\mathcal{F}_{\mathsf{VSSReshare}}$ such that parties in $\mathcal{C}_{l+1}$ hold $\langle b', f'_{b'}(x)\rangle_{l+1}$. P sends $(\mathsf{VSSShare}, sid, f_{ab'})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ such that parties in $\mathcal{C}_{l+1}$ hold $\langle ab', f''_{ab'}(x)\rangle_{l+1}$.
Committee $\mathcal{C}_{l+1}$**:** Each party $P_i^{l+1} \in \mathcal{C}_{l+1}$ does the following after receiving e from $\mathcal{F}_{\mathsf{Beacon}}$:

1. Send $(\mathsf{VSSReconstruct}, sid, 1, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ to reveal $r = e \cdot b + b'$.
2. Send $(\mathsf{VSSReconstruct}, sid, 1, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ to reveal $d = r \cdot a - e \cdot c - a \cdot b'$. Accept if $d = 0$, else reject.

Figure 20: Protocol Π_{PLM} in the $(\mathcal{F}_{\mathsf{VSSReshare}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}_{\mathsf{StatVSS}})$-hybrid model.

4.6 Proof of Local Multiplication

As part of our multiplication protocol employing techniques from [GRR98], we will use Protocol Π_{PLM} as presented in Figure 20. This protocol allows for checking whether shares of the result of a multiplication are valid. We present Protocol Π_{PLM} as a separate protocol for the sake of clarity but do not prove its security directly. Since it is used as a sub-protocol in our multiplication protocol, we argue about it as part of the proof of security for multiplication. As Protocol Π_{PLM} is derived from our statistically secure VSS, its communication complexity is similar. The round complexity is of 2 rounds for $\mathcal{C}_l$ and 4 rounds for $\mathcal{C}_{l+1}$.

4.7 Multiplication

In Figure 21, we define Functionality $\mathcal{F}_{\mathsf{Mult}}$ for secure multiplication in the Multi-Layered MPC model. This functionality allows for parties in committee $\mathcal{C}_l$ holding augmented shares of the inputs (*i.e.* both

shares and shares-of-shares of the same value) to provide augmented shares of the multiplication output to $\mathcal{C}_{l+1}$.

Public Parameters: Input committee $\mathcal{C}_l$, Output committee $\mathcal{C}_{l+1}$
Secret Inputs: $[\![a, f_a(x), (f_{a,i}(x))_{i\in[0,t]}]\!]_l$, $[\![b, f_b(x), (f_{b,i}(x))_{i\in[0,t]}]\!]_l$

1. Upon receiving $(\mathsf{Mult}, sid, f_a(i), g_b(i), \{f_{a,j}(i), g_{b,j}(i)\}_{j\in[t]})$ from all honest parties $P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_l$, interpolate polynomials $f_a, g_b, \{f_{a,j}, g_{b,j}\}_{j\in[t]}$. It sends to the (ideal) adversary the values $f_a(i), g_b(i), \{f_{a,j}(i), g_{b,j}(i)\}_{j\in[0,t], P_i^l\in\mathcal{I}_l}$.
2. The functionality samples a random polynomial $h_\gamma(x) = \sum_{k=0}^{t} h_{\gamma,k}x^k$ of degree at most t such that $h_\gamma(0) = \gamma = f_a(0)\cdot g_b(0)$. In addition it also samples for each $j \in [t]$, a random degree-t polynomial $h_{\gamma,j}(x)$ such that $h_{\gamma,j}(0) = h_{\gamma,j}$. It then distributes $\langle \gamma, h_\gamma(x)\rangle_{l+1}$, $\{\langle h_\gamma(i), h_{\gamma,i}(x)\rangle_{l+1}\}_{i\in[t]}$ to committee $\mathcal{C}_{l+1}$. For all $P_j^{l+1} \in \mathcal{I}_{l+1}$, it additionally sends $h_\gamma(j), \{h_{\gamma,i}(j)\}_{i\in[0,t]}$ to $\mathcal{S}$.

Figure 21: Functionality $\mathcal{F}_{\mathsf{Mult}}$

Secret Inputs: Party $P_i^l \in \mathcal{C}_l$ holds an augmented VSS of a and augmented VSS of b, namely $[\![a, f_a(x), (f_{a,i}(x))_{0\le i\le t}]\!]_l, [\![b, f_b(x), (f_{b,i}(x))_{0\le i\le t}]\!]_l$.
Output to $\mathcal{C}_{l+1}$: $[\![\gamma = ab, f_\gamma(x), (f_{\gamma,i}(x))_{0\le i\le t}]\!]_{l+1}$

Committee $\mathcal{C}_l$: Each P_i^l does the following:
1. It sends $(\mathsf{AugVSSShare}, sid, \gamma_i)$ to $\mathcal{F}_{\mathsf{AugVSS}}$ where $\gamma_i = f_a(i)\cdot f_b(i)$, which results in $[\![\gamma, f_\gamma(x), (f_{\gamma,i}(x))_{0\le i\le t}]\!]_j$ in $\mathcal{C}_{l+1}$. It then executes Π_{PLM} on $f_a(i), f_b(i), \gamma_i$. Additionally send $(\mathsf{Reshare}, sid, f_{a,j}(i))$ and $(\mathsf{Reshare}, sid, f_{b,j}(i))$ for all $0 \le j \le t$ to $\mathcal{F}_{\mathsf{VSSReshare}}$ such that parties in $\mathcal{C}_{l+1}$ hold $\langle s_{a,i} = f_a(i), f'_{s_{a,i}}(x)\rangle_{l+1}$ and $\langle s_{b,i} = f_b(i), f'_{s_{b,i}}(x)\rangle_{l+1}$.

Committee $\mathcal{C}_{l+1}$: For any i for which the Π_{PLM} or AugVSS fails, send $(\mathsf{VSSReconstruct}, sid, 1, \mathcal{C}_{l+1})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ to reveal a_i, b_i and set $\gamma_i = a_i \cdot b_i$. Using lagrange interpolation, compute an augmented sharing $\mathsf{VSS}_{P_i^{l+1}}(c = a\cdot b)$ using $\mathsf{VSS}_{P_i^l}(\gamma_i)$ and $\mathsf{VSS}_{P_i^{l+1}}(c_i)$ using $\mathsf{VSS}_{P_i^l}(\gamma_{i,j})$.

Figure 22: Protocol Π_{Mult} in the $(\mathcal{F}_{\mathsf{VSSReshare}}, \mathcal{F}_{\mathsf{AugVSS}})$-hybrid model.

In Figure 22, we present Protocol Π_{Mult}, which realizes $\mathcal{F}_{\mathsf{Mult}}$. This protocols follows the approach of [GRR98], where parties in $\mathcal{C}_l$ first locally multiply their shares of the inputs and then invoke Protocol Π_{PLM} to verify if these multiplications have been done correctly. Parties in $\mathcal{C}_{l+1}$ learn the result of these checks and, if a check fails for a multiplication done by a given party, that party's shares are reconstructed so that the multiplication can be correctly done in public. Since we construct this protocol in the Multi-Layered model, our construction crucially relies on our parallel linearly homomorphic VSS protocol that allows us to (re-)share secrets held by parties in $\mathcal{C}_l$ so that parties in $\mathcal{C}_{l+1}$ learn (linear combinations of) the shares.

The communication complexity of Protocol Π_{Mult} is of $\mathcal{O}(\kappa\cdot n^8)$ field elements for $\mathcal{C}_l$ and of $\mathcal{O}(\kappa\cdot n^5)$ field elements for $\mathcal{C}_{l+1}$. The round complexity is of 1 round for $\mathcal{C}_l$ and 12 rounds for $\mathcal{C}_{l+1}$. Notice that using this protocol for computing a circuit with d multiplicative layers requires only $d+1$ committees, since the first committee shares the inputs and initiates Protocol Π_{Mult}, which is invoked d times until the last committee reconstructs the outputs.

Theorem 12. *Protocol* Π_{Mult} *securely realizes* $\mathcal{F}_{\mathsf{Mult}}$ *with* $(\mathsf{negl}(\kappa))$*-statistical security in the* $(\mathcal{F}_{\mathsf{VSSReshare}}, \mathcal{F}_{\mathsf{StatVSS}}, \mathcal{F}_{\mathsf{AugVSS}})$*-hybrid model.*

The proof is presented in Appendix D.5.

4.8 Protocol for Statistically Secure MPC

Since we obtain secure additions trivially by the linear homomorphism of our VSS $\mathcal{F}_{\mathsf{StatVSS}}$ and we have secure multiplication $\mathcal{F}_{\mathsf{Mult}}$, it follows directly that we obtain a statistically secure MPC protocol. Our MPC protocol's communication complexity is of the order of out multiplication protocol's communication complexity, *i.e.* $\mathcal{O}(\kappa \cdot n^8)$. For a circuit of multiplicative depth d, our protocol runs with $d+1$ committees, each executing $\delta = 13$ rounds.

Theorem 13. *Let f be an n-party functionality computed by a layered arithmetic circuit C over a finite field $\mathbb{F}$ of multiplicative depth d. There exists a $(n, t, 13, d+1)$-layered MPC protocol in the $(\mathcal{F}_{\mathsf{StatVSS}}, \mathcal{F}_{\mathsf{Mult}})$-hybrid model realizing the functionality f with statistical security for $n = 2t+1$ and communication complexity $\mathcal{O}(\kappa \cdot n^8)$, where κ is the security parameter.*

Notice that Theorem 13 follows from a similar construction as in the pefect security case. Since we have a statistically secure linearly homomorphic VSS $\mathcal{F}_{\mathsf{StatVSS}}$, additions are trivially computed locally. Multiplications are handled by our statistically secure multiplication functionality $\mathcal{F}_{\mathsf{Mult}}$. Since executing $\mathcal{F}_{\mathsf{Mult}}$ takes longer than a local addition, results of addition gates can be refreshed by multiplying them by 1 in parallel with the circuit's multiplication gates. Alternatively, more efficient re-sharing protocols can be used, leveraging shares of random values obtained via our statistically secure linearly homomorphic VSS $\mathcal{F}_{\mathsf{StatVSS}}$.

References

AHKP22. A. Acharya, C. Hazay, V. Kolesnikov, and M. Prabhakaran. SCALES - MPC with small clients and larger ephemeral servers. In *TCC 2022, Part II*, *LNCS* 13748, pages 502–531. Springer, Cham, November 2022.

AKP20. B. Applebaum, E. Kachlon, and A. Patra. The round complexity of perfect MPC with active security and optimal resiliency. In *61st FOCS*, pages 1277–1284. IEEE Computer Society Press, November 2020.

AKP23. B. Applebaum, E. Kachlon, and A. Patra. The round complexity of statistical MPC with optimal resiliency. In *55th ACM STOC*, pages 1527–1536. ACM Press, June 2023.

AL17. G. Asharov and Y. Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, January 2017.

BDO23. L. Braun, I. Damgård, and C. Orlandi. Secure multiparty computation from threshold encryption based on class groups. In *CRYPTO 2023, Part I*, *LNCS* 14081, pages 613–645. Springer, Cham, August 2023.

BEP23. A. Bienstock, D. Escudero, and A. Polychroniadou. On linear communication complexity for (maximally) fluid MPC. In *CRYPTO 2023, Part I*, *LNCS* 14081, pages 263–294. Springer, Cham, August 2023.

BEP24. A. Bienstock, D. Escudero, and A. Polychroniadou. Perfectly secure fluid MPC with abort and linear communication complexity. *IACR Cryptol. ePrint Arch.*, page 2071, 2024.

BGG+20. F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In *TCC 2020, Part I*, *LNCS* 12550, pages 260–290. Springer, Cham, November 2020.

BGW88. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

BHK+22. F. Benhamouda, S. Halevi, H. Krawczyk, A. Miao, and T. Rabin. Threshold cryptography as a service (in the multiserver and YOSO models). In *ACM CCS 2022*, pages 323–336. ACM Press, November 2022.

BJMS20. S. Badrinarayanan, A. Jain, N. Manohar, and A. Sahai. Secure MPC: Laziness leads to GOD. In *ASIACRYPT 2020, Part III*, *LNCS* 12493, pages 120–150. Springer, Cham, December 2020.

Can00. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CCD88. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

CD24. I. Cascudo and B. David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In *EUROCRYPT 2024, Part V*, *LNCS* 14655, pages 216–248. Springer, Cham, May 2024.

CDD+99. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT'99*, *LNCS* 1592, pages 311–326. Springer, Berlin, Heidelberg, May 1999.

CDD+04. R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, June 2004.

CDGK22. I. Cascudo, B. David, L. Garms, and A. Konring. YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In *ASIACRYPT 2022, Part I*, *LNCS* 13791, pages 651–680. Springer, Cham, December 2022.

CDK+22. M. Campanelli, B. David, H. Khoshakhlagh, A. Konring, and J. B. Nielsen. Encryption to the future - A paradigm for sending secret messages to future (anonymous) committees. In *ASIACRYPT 2022, Part III*, *LNCS* 13793, pages 151–180. Springer, Cham, December 2022.

CDM00. R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT 2000*, *LNCS* 1807, pages 316–334. Springer, Berlin, Heidelberg, May 2000.

CGG+21. A. R. Choudhuri, A. Goel, M. Green, A. Jain, and G. Kaptchuk. Fluid MPC: Secure multiparty computation with dynamic participants. In *CRYPTO 2021, Part II*, *LNCS* 12826, pages 94–123, Virtual Event, August 2021. Springer, Cham.

CKR+23. R. Canetti, S. Kolby, D. Ravi, E. Soria-Vazquez, and S. Yakoubov. Taming adaptivity in YOSO protocols: The modular way. In *TCC 2023, Part II*, *LNCS* 14370, pages 33–62. Springer, Cham, November / December 2023.

CM19. J. Chen and S. Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

DDG+23. B. David, G. Deligios, A. Goel, Y. Ishai, A. Konring, E. Kushilevitz, C.-D. Liu-Zhang, and V. Narayanan. Perfect MPC over layered graphs. In *CRYPTO 2023, Part I*, *LNCS* 14081, pages 360–392. Springer, Cham, August 2023.

DEP23. I. Damgård, D. Escudero, and A. Polychroniadou. Phoenix: Secure computation in an unstable network with dropouts and comebacks. In *ITC 2023*, *LIPIcs* 267, pages 7:1–7:21. Schloss Dagstuhl, June 2023.

DGKR18. B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT 2018, Part II*, *LNCS* 10821, pages 66–98. Springer, Cham, April / May 2018.

DKLN24. G. Deligios, A. Konring, C. Liu-Zhang, and V. Narayanan. Statistical layered MPC. In *Theory of Cryptography - 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part IV*, *Lecture Notes in Computer Science* 15367, pages 362–394. Springer, 2024.

DKLZN24. G. Deligios, A. Konring, C.-D. Liu-Zhang, and V. Narayanan. Statistical layered MPC. In *TCC 2024, Part IV*, *LNCS* 15367, pages 362–394. Springer, Cham, December 2024.

EFR21. A. Erwig, S. Faust, and S. Riahi. Large-scale non-interactive threshold cryptosystems through anonymity. Cryptology ePrint Archive, Report 2021/1290, 2021.

EHOY20. K. Eldefrawy, S. Hwang, R. Ostrovsky, and M. Yung. Communication-efficient (proactive) secure computation for dynamic general adversary structures and dynamic groups. In *SCN 20*, *LNCS* 12238, pages 108–129. Springer, Cham, September 2020.

ELL22. K. Eldefrawy, T. Lepoint, and A. Leroux. Communication-efficient proactive MPC for dynamic groups with dishonest majorities. In *ACNS 22International Conference on Applied Cryptography and Network Security*, *LNCS* 13269, pages 565–584. Springer, Cham, June 2022.

FHM98. M. Fitzi, M. Hirt, and U. M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In *CRYPTO'98*, *LNCS* 1462, pages 121–136. Springer, Berlin, Heidelberg, August 1998.

GHK+21. C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In *CRYPTO 2021, Part II*, *LNCS* 12826, pages 64–93, Virtual Event, August 2021. Springer, Cham.

GHM+21. C. Gentry, S. Halevi, B. Magri, J. B. Nielsen, and S. Yakoubov. Random-index PIR and applications. In *TCC 2021, Part III*, *LNCS* 13044, pages 32–61. Springer, Cham, November 2021.

GIKR01. R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589. ACM Press, July 2001.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GPS19. Y. Guo, R. Pass, and E. Shi. Synchronous, with a chance of partition tolerance. In *CRYPTO 2019, Part I*, *LNCS* 11692, pages 499–529. Springer, Cham, August 2019.

GRR98. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *17th ACM PODC*, pages 101–111. ACM, June / July 1998.

KLR06. E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In *38th ACM STOC*, pages 109–118. ACM Press, May 2006.

KRY24. S. Kolby, D. Ravi, and S. Yakoubov. Constant-round YOSO MPC without setup. *CiC*, 1(3):30, 2024.

LZMR+24. C.-D. Liu-Zhang, E. Masserova, J. Ribeiro, P. Soni, and S. A. Thyagarajan. Improved YOSO randomness generation with worst-case corruptions. Cryptology ePrint Archive, Report 2024/1268, 2024.

NRO22. J. B. Nielsen, J. L. Ribeiro, and M. Obremski. Public randomness extraction with ephemeral roles and worst-case corruptions. In *CRYPTO 2022, Part I*, *LNCS* 13507, pages 127–147. Springer, Cham, August 2022.

OY91. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In *10th ACM PODC*, pages 51–59. ACM, August 1991.

RB89. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

RS22. R. Rachuri and P. Scholl. Le mans: Dynamic and fluid MPC for dishonest majority. In *CRYPTO 2022, Part I*, *LNCS* 13507, pages 719–749. Springer, Cham, August 2022.

A Additional Preliminaries

In this section, we provide our additional technical preliminaries.

A.1 Secret Sharing

We recall the definition from [DDG+23]. A (t, n)-secure secret sharing scheme takes a secret s as input and computes n shares with the secrecy guarantee that any set of at most t shares reveal nothing about s while any set of $t+1$ correct shares can be used to efficiently reconstruct the secret. A robust secret sharing scheme additionally guarantees correct reconstruction from n shares, even when an arbitrary set of t out of the n shares have been tampered. Formally,

Definition 14 (Secret Sharing). *A secret sharing scheme with secret domain S and share domain U is a pair of algorithms $(\mathsf{Sh}, \mathsf{Rec})$ called the sharing algorithm and the reconstruction algorithm, respectively.*

- ***Sharing:*** *The sharing algorithm Sh takes as input a secret $s \in S$ and randomness ρ chosen uniformly from some domain R and computes $(s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s, \rho)$, where $s_i \in U$ will be called the i-th share of the secret s. If ρ is clear from the context we just write $(s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s)$.*
- ***Reconstruction:*** *The reconstruction algorithm Rec takes as input the set of all shares and computes $\hat{s} \leftarrow \mathsf{Rec}(s_1, \ldots, s_n)$.*

The (t, n)-secret sharing scheme $(\mathsf{Sh}, \mathsf{Rec})$ should satisfy the following properties:

- **Correctness:** *For all $\rho \leftarrow\!\!\$\ R$, and any $s \in S$, we have $\mathsf{Rec}(\mathsf{Sh}(s, \rho)) = s$.*
- **t-security:** *For any $s, s' \in S$, and for all subsets $\mathcal{I} \subset [n]$ such that $|\mathcal{I}| \leq t$, we have,*

$$\begin{aligned}\big\{\{s_i\}_{i\in\mathcal{I}} \mid \rho \leftarrow\!\!\$\ R, (s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s, \rho)\big\} \\ \equiv \big\{\{s_i\}_{i\in\mathcal{I}} \mid \rho \leftarrow\!\!\$\ R, (s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s', \rho)\big\}\end{aligned}$$

The scheme $(\mathsf{Sh}, \mathsf{Rec})$ is said to be robust against t corruptions if it additionally satisfies the following reconstruction property:

- **t-robustness:** *For all $\rho \leftarrow\!\!\$\ R$, a secret $s \in S$, a set $\mathcal{I} \subset [n]$ of size at most t, and shares $\hat{s}_i \in U$, for $i \in \mathcal{I}$, we have,*

$$\mathsf{Rec}(\{s_i\}_{i\in[n]\setminus\mathcal{I}}, \{\hat{s}_i\}_{i\in\mathcal{I}}) = s, \text{ where } (s_1, \ldots, s_n) \leftarrow \mathsf{Sh}(s, \rho)$$

A.2 Statistical Message Authentication Codes

We recall the definition and construction of the statistical message authentication codes from [DKLN24].

Definition 15. *A message authentication code scheme MAC is a couple $(\mathsf{Aut}, \mathsf{Vf})$ where $\mathsf{Aut} : \mathcal{M} \times \mathcal{K} \to \mathcal{T}$ and $\mathsf{Vf} : \mathcal{T} \times \mathcal{M} \times \mathcal{K} \to \{0, 1\}$ are efficient algorithms such that,*

$$\mathsf{Vf}(\mathsf{Aut}(m, k), m, k) = 1$$

for all $m \in \mathcal{M}$ and $k \in \mathcal{K}$.

Definition 16. *Let $\mathcal{D}$ be a distribution on $\mathcal{K}^\ell$. We say that a MAC scheme MAC is $\mathcal{D}$-linear if $\mathcal{M}$, $\mathcal{K}$, and $\mathcal{T}$ are $\mathbb{F}$-vector spaces and for all $\ell \in \mathbb{N}$, for keys $(k_1, \ldots, k_\ell) \leftarrow_{\mathcal{D}} \mathcal{K}^\ell$, all messages $(m_1, \ldots, m_\ell) \in \mathcal{M}^\ell$, and all linear functions $L : \mathbb{F}^\ell \to \mathbb{F}$ it holds that,*

$$\mathsf{Vf}(L(k_1, \ldots, k_\ell), L(m_1, \ldots, m_\ell), L(\mathsf{Aut}(m_1, k_1), \ldots, \mathsf{Aut}(m_\ell, k_\ell))) = 1$$

Definition 17. *A $\mathcal{D}$-linear MAC scheme $\mathsf{MAC} = (\mathsf{Aut}, \mathsf{Vf})$ is statistically $(\mathcal{D}, q, \epsilon)$-unforgeable if for all adversaries $\mathcal{A}$ it holds that $\mathsf{Adv}^{\mathsf{CorrForge}}_{\mathcal{D},q}(\mathcal{A}) \leq \epsilon$, where the security game $\mathsf{CorrForge}^{(\mathsf{Aut},\mathsf{Vf})}_{\mathcal{D},q}$ is defined in Figure 23.*

$\mathsf{CorrForge}^{(\mathsf{Aut},\mathsf{Vf})}_{\mathcal{D},q}(\mathcal{A})$

1 : Sample $(k_1, \ldots, k_\ell) \leftarrow_{\mathcal{D}} \mathcal{K}^\ell$
2 : $\mathcal{A}$ can make up to $q \leq \ell$ queries $(m_1, j_1), \ldots, (m_q, j_q) \in \mathcal{M} \times [\ell]$:
3 : Receive the corresponding tags $t_i = \mathsf{Aut}(m_i, k_{j_i})$
4 : $\mathcal{A}$ provides a triple (m', t', L) where $m' \in \mathcal{M}$, $t' \in \mathcal{T}$ and $L : \mathbb{F}^\ell \to \mathbb{F}$
5 : **return** 1 iff $\mathsf{Vf}(t', m', f_L(k_{j_1}, \ldots, k_{j_q})) = 1$ and $m' = L(m_1, \ldots, m_q)$

Fig. 23. $\mathsf{CorrForge}^{(\mathsf{Aut},\mathsf{Vf})}_{\mathcal{D},q}$ Security Game for $\mathcal{D}$-linear $\mathsf{MAC} = (\mathsf{Aut}, \mathsf{Vf})$ scheme.

Construction. Let $\mathbb{F}$ be a finite field. Let $\mathcal{M} = \mathcal{T} = \mathbb{F}$ and $\mathcal{K} = \mathbb{F}^2$. We define $\mathsf{Aut} : \mathbb{F} \times \mathbb{F}^2 \to \mathbb{F}$ as

$$(m, (\alpha, \beta)) \to \alpha \cdot m + \beta \tag{1}$$

Let $A, B_1, \ldots, B_\ell$ be independent uniform random variables on $\mathbb{F}$. Consider the distribution $\mathcal{D}$ of $(K_1, \ldots, K_\ell)$ where $K_i = (A, B_i)$.

A.3 The GIKR VSS protocol [GIKR01]

We describe a corruption aware n-parallel VSS functionality $\mathcal{F}_{\mathsf{VSS}}$ in Figure 24 along the lines of [BGW88]. Next, we describe the GIKR protocol [GIKR01] realizing this functionality.

Public Parameters: Dealers and Share-holders $P_1, \ldots, P_n$
Secret Inputs: Each party P_i holds a degree-t polynomial $f_i(x)$ such that $f_i(0) = s_i$.
Additional Input to functionality: Corruption Set $\mathcal{I}$.

Share:
- Upon receiving $(\mathsf{Share}, sid, f_i)$, from $P_i \notin \mathcal{I}$, $\mathcal{F}_{\mathsf{VSS}}$ stores the polynomial $f_i^* = f_i$.
- $\mathcal{F}_{\mathsf{VSS}}$ sends to the (ideal) adversary $\mathcal{S}$ the shares $\{f_i(j)\}_{i, P_i \notin \mathcal{I}}$ for all j where $P_j \in \mathcal{I}$.
- $\mathcal{F}_{\mathsf{VSS}}$ receives a degree-t polynomial $f_{\mathcal{S},i}$ from the (ideal) adversary on behalf of all $P_i \in \mathcal{I}$ and set $f_i^* = f_{\mathcal{S},i}$.
- $\mathcal{F}_{\mathsf{VSS}}$ sends $\{f_i^*(j)\}_{i \in [n]}$ to party P_j.

Reveal: If all honest $P_i \notin \mathcal{I}$ send $(\mathsf{Reconstruct}, sid)$, then send $f^*(0)$ to all parties.

Figure 24: Verifiable Secret Sharing functionality $\mathcal{F}_{\mathsf{VSS}}$

We use the VSS protocol described by Gennaro, Ishai, Kushelevitz and Rabin in [GIKR01]. It is based on the common approach for VSS schemes based on bi-variate polynomials over a finite field $\mathbb{F}$. We describe the protocol below for a single dealer:

Multi-Dealers Π_{VSS}. One could just run the above protocol (in Figure 25) in parallel, without each party P_i being a dealer with input $f_i(x)$, to realise $\mathcal{F}_{\mathsf{VSS}}$. As each instantiation of the Π_{VSS} protocol incurs a communication complexity of $\mathcal{O}(n^2)$ field elements, the parallel VSS protocol would incur $\mathcal{O}(n^3)$ field elements. In general, for any VSS with communication complexity $\mathcal{O}(\mathsf{comm}_{\mathsf{VSS}})$, we would incur $\mathcal{O}(n \cdot \mathsf{comm}_{\mathsf{VSS}})$

Lemma 18. *Protocol Π_{VSS} (in Figure 25) realises the functionality $\mathcal{F}_{\mathsf{VSS}}$ (in Figure 24), with perfect security for $t < n/3$.*

We observe that the above protocol is a 4-round protocol, and we can use it as is within any committee participating in an Multi-Layered MPC protocol as described above as long as the committee stays for sufficiently many rounds before a state transfer. We note that one can alternatively use the 3-round VSS protocol as described by Applebaum, Kachlon and Patra in [AKP20]. While we get a round optimal VSS in this case, we incur a cubic cost in communication, while the one described in 25 incurs quadratic complexity.

Public Parameters: Dealer P_D, Shareholders $P_1 \ldots, P_n$, A field description $\mathbb{F}$.
Secret Input: P_D, holds a polynomial $f(x)$.

Sharing Phase $\mathsf{VSS.Share}$**:**

1. The dealer P_D samples a polynomial $F(x, y)$ uniformly at random, such that $F(0, 0) = s$. It sends to party P_i the polynomials, $f_i(x) = F(x, i)$ and $g_i(y) = F(i, y)$. Each party P_i also sends to each P_j, a uniform random pad r_{ij}
2. Each party P_i broadcasts $a_{ij} = f_i(j) \oplus r_{ij}, b_{ij} = g_i(j) \oplus r_{ji}$.
3. For every conflicting pair, where $(a_{ij} \neq b_{ji})$, the parties broadcast their local points, i.e., P_i broadcasts $f_i(j)$, P_j broadcasts $g_j(i)$ and P_D broadcasts $F(j, i)$. We say that there is a conflict with respect to a party P_i if its broadcasted value does not match the dealer's value. If there are more than t conflicts, P_D is discarded.
4. For each conflict with respect to any party P_i, the dealer broadcasts the polynomial $f_i(x)$, and each party P_j that does not have a conflict broadcasts $g_j(i)$. Any party P_i that has $g_j(i)) \neq f_i(j)$ for the polynomial $f_i(x)$ broadcasted by the dealer, a conflict is raised. If the number of conflicts is more than t, then the dealer P_D is disqualified. Otherwise, each party P_i store the share as $s_i = g_i(0)$.

Reconstruction $\mathsf{VSS.Reconstruct}$**:**

1. Every P_i broadcasts it's share s_i, and parties interpolate a polynomial f' of degree atmost t through these broadcasted shares. Output $f'(0)$.

Figure 25: Protocol Π_{VSS}

A.4 Random Sharing

We describe a functionality $\mathcal{F}_{\mathsf{RandomShare}}$ that deals shares of a random value to a set of shareholders in Figure 26. Next, we describe a standard way to realize this functionality.

Public Parameters: Shareholders $P_1, \ldots, P_n$
Notations: We will denote this functionality by $\langle s, f(x)\rangle \leftarrow \$$.

Share: Upon receiving $(\mathsf{RandShare}, sid)$ from all honest $P_i \notin \mathcal{I}$, $\mathcal{F}_{\mathsf{RandomShare}}$ samples $s \xleftarrow{\$} \mathbb{F}$, and $c_i \xleftarrow{\$} \mathbb{F}$ for each $i \in [t]$. Let $f(x) = s + \sum_{i=1}^{t} c_i x^i$. The functionality outputs $(\mathsf{OutRandShare}, sid, f(j))$ to P_j for each $j \in [n]$, hence distributing $\langle s, f(x)\rangle$.

Figure 26: Functionality $\mathcal{F}_{\mathsf{RandomShare}}$

Public Parameters: Shareholders $P_1, \ldots, P_n$
Sub-Routines: a VSS functionality $\mathcal{F}_{\mathsf{VSS}}$
Output: $\langle s, f(x)\rangle$
Additional input to functionality: $\mathcal{I}$

1. For each $i \in [t + 1]$, P_i samples $s_i \xleftarrow{\$} \mathbb{F}$ and $c_{i,l} \xleftarrow{\$} \mathbb{F}$ for $l \in [t]$. Define a polynomial $f_{s_i}(x) = s_i + c_{i,1}x + \ldots c_{i,t}x^t$. Each P_i then sends $(\mathsf{Share}, sid, f_{s_i})$ $\mathcal{F}_{\mathsf{VSS}}$ to create a sharing $\langle s_i, f_{s_i}(x)\rangle$ with $P_1, \ldots, P_n$ as shareholders, and $\mathcal{I}$ as an additional input.
2. Each party P_i then locally computes their corresponding share of $\langle s, f(x)\rangle$ as $\langle \sum_{j=1}^{t+1} s_j, \sum_{j=1}^{t+1} f_j(x)\rangle = \sum_{j=1}^{t+1} \langle s_j, f_j(x)\rangle$.

Figure 27: Protocol $\Pi_{\mathsf{RandomShare}}$ in the $\mathcal{F}_{\mathsf{VSS}}$-hybrid model.

As described above, $\Pi_{\mathsf{RandomShare}}$ makes 1 call to the parallel n-VSS functionality $\mathcal{F}_{\mathsf{VSS}}$. Hence, the communication cost of the random sharing protocol matches that of a protocol instantating $\mathcal{F}_{\mathsf{VSS}}$, *i.e.,* $\mathcal{O}(n \cdot \mathsf{comm}_{\mathsf{VSS}})$.

Lemma 19. *For $n, t \in \mathbb{N}$, the protocol $\Pi_{\mathsf{RandomShare}}$ (in Figure 27) realizes the functionality $\mathcal{F}_{\mathsf{RandomShare}}$ (in Figure 26) in the $\mathcal{F}_{\mathsf{VSS}}$-hybrid model, with perfect security for $t < n/3$.*

A.5 Information Theoretic Signatures in the standard setting

As part of our statistically secure Rumors MPC protocol, we will use standard Information Theoretical Signatures (IT-Sig) where the Setup and Reveal phases are executed in the same committee (besides our new IT-Sig protocol where the Setup phase happens in one committee and the Reveal phase happens in the next). We present an IT-Sig functionality $\mathcal{F}_{\mathsf{ITSigClassical}}$ in Figure 28 and then present a standard protocol for realizing this functionality.

Public Parameters: Sender P_S, Intermediate Receiver P_{int}, and Receiver Set $\mathcal{P}_{\mathcal{R}}$.
Secret Inputs: P_S holds messages $(m_1, \ldots, m_{n'}) \in M$. In addition, there is a public value L agreed upon by honest $P_i \notin \mathcal{I}$ where $L : M^k \mapsto M$ is a linear operator.

Setup: Upon receiving $(\mathsf{SetupSig}, sid, (m_1, \ldots, m_{n'}))$ from P_S, store $(m_1^*, \ldots, m_{n'}^*) = (m_1, \ldots, m_{n'})$ if $P_S \notin \mathcal{I}$. Otherwise, receive $(m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$ from the (ideal) adversary $\mathcal{S}$ and set $(m_1^*, \ldots, m_{n'}^*) = (m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$.

Reveal: If P_{int} sends $(\mathsf{RevealSig}, sid, L, \mathcal{P}_{\mathcal{R}})$, then:
- If P_{int} is honest, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_{\mathcal{R}}$. If $\mathcal{P}_{\mathcal{R}} = P_i$ for some i, then send $L(m_1^*, \ldots, m_{n'}^*)$ to the (ideal) adversary $\mathcal{S}$.
- If P_{int} is corrupt:
  - P_S is honest: Receive boolean $\mathsf{reveal} \in \{0, 1\}$ from the (ideal) adversary. If $\mathsf{reveal} = 1$, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_{\mathcal{R}}$, else send $\perp$.
  - P_S is corrupt: Receive $l_{\mathcal{S}}^*$ from the (ideal) adversary $\mathcal{S}$ and send $l_{\mathcal{S}}^*$ to all parties in $\mathcal{P}_{\mathcal{R}}$.

Figure 28: IT-Sig Functionality $\mathcal{F}_{\mathsf{ITSigClassical}}$

In Figure 29, we present a classical IT-Sig protocol following the blueprint of [RB89] where the entire protocol is supposed to be executed by a single committee.

Notice that when we run IT-Sig within the same committee (*i.e.* with both Sender and Receiver in the same committee) the round complexity is 5 rounds for the Setup phase and 2 rounds for the Reveal phase. Saving 1 round for the Setup phase in comparison to the Multi-Layered IT-Sig protocol we constructed.

Lemma 20. *Protocol $\Pi_{\mathsf{ITSigClassical}}$ (in Figure 29) realizes the functionality $\mathcal{F}_{\mathsf{ITSigClassical}}$ (in Figure 28), with perfect security for $t < n/3$.*

A.6 Distributed Commitments in the standard setting

As part of our statistically secure Rumors MPC protocol, we will use standard Distributed Commitments where the Setup and Reveal phases are executed in the same committee (besides our new Distributed Commitment protocol where the Setup phase happens in one committee and the Reveal phase happens in the next). We present a Distributed Commitment functionality $\mathcal{F}_{\mathsf{ITSigClassical}}$ in Figure 30 and then present a standard protocol for realizing this functionality.

In Figure 31 we described a classical distributed commitment following the blueprint of [RB89] where the protocol is executed by a single committee. This protocol follows is a straightforward manner from our classical IT-Sig functionality.

Notice that when we run distributed commitments within the same committee (*i.e.* with both Sender and Receivers in the same committee) the round complexity is 5 rounds for the Setup phase and 2 rounds for the Reveal phase. Saving 1 round for the Setup phase in comparison to the Multi-Layered distributed commitments we constructed.

Lemma 21. *Protocol $\Pi_{\mathsf{DistCommitClassical}}$ (in Figure 31) realizes the functionality $\mathcal{F}_{\mathsf{DistCommitClassical}}$ (in Figure 30), with perfect security for $t < n/3$.*

A.7 Random Beacon

In Figure 32, we define Functionality $\mathcal{F}_{\mathsf{Beacon}}$, which captures a random beacon activated by committee $\mathcal{C}_l$ and delivering a uniformly random output to $\mathcal{C}_{l'}$ for $l' > l$. Such a beacon was constructed as part of the YOSO [GHK+21] and Statistically layered MPC [DKLZN24] protocols as well as in [NRO22, LZMR+24].

Public Inputs: Sender P_S, Verifiers $P_1, \cdots, P_n$, Intermediate Receiver P_{int}.
Secret Inputs: P_S holds input $(m_1, \ldots, m_{n'}) \in M$.

Setup Phase $\mathsf{ITSig.Setup}$**:**

- Sender P_S generates keys $K_{i,j,k}$ for $i \in [n'], j \in [n], k \in [\kappa]$ and computes $M_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m_i)$. It sends $\{K_{i,j,k}\}_{i\in[n'],k\in[\kappa]}$ to P_j^{l+1} and $(m_1, \ldots, m_{n'}), \{M_{1,j,k}, \ldots, M_{n',j,k}\}_{j\in[n],k\in[\kappa]}$ to P_{int}.
- Each P_j chooses half the indices at random, say INX_j, and broadcasts $K_{i,j,k}$ for $i \in [n'], k \in INX_j$.
- Sender P_S broadcasts $\bar{K}_{i,j,k}$ for all $i \in [n'], j \in [n]$, and for all $k \in INX_j$.
- If there exists indices i and j for which $\mathsf{MAC}_{\bar{K}_{i,j,k}}(m_i) \neq M_{i,j,k}$, then the receiver P_{int} broadcasts a complaint, and P_S broadcasts $L(m_1^*, \ldots, m_{n'}^*)$ and their corresponding tags.

Reveal Phase $\mathsf{ITSig.Reveal}$**:** If P_{int} forced a reveal in $\mathsf{ITSig.Setup}$, then output the same message $L(\bar{m}_1, \ldots, \bar{m}_{n'})$. Else:

- When $\mathcal{P}_{\mathcal{R}} = \{P_1, \cdots, P_n\}$ do the following:
  - P_{int} broadcasts $(L(m_1, \ldots, m_{n'}), \{L(M_{1,j,k}, \ldots, M_{n',j,k})\}_{j\in[n],k\notin INX_j})$.
  - Set the number of votes for $L(m_1, \ldots, m_{n'})$ to be number of j's for which $L(\bar{K}_{1,j,k}, \ldots, \bar{K}_{n',j,k}) \neq L(K_{1,j,k}, \ldots, K_{n',j,k})$, for some $k \in INX_j$. For all other j's, P_j^{l+1} broadcasts $L(K_{1,j,k}, \ldots, K_{n',j,k})$ for $k \notin INX_j$. If $\mathsf{MAC}_{L(K_{1,j,k},\ldots,K_{n',j,k})}(L(m_1, \ldots, m_{n'})) = L(M_{1,j,k}, \ldots, M_{n',j,k})$ for any one of the recovered values, then increment the vote by 1. If there are $t+1$ votes, output $L(m_1, \ldots, m_{n'})$.
- When $\mathcal{P}_{\mathcal{R}} = P_r$ for some r, do the following:
  - P_{int} sends $(L(m_1, \ldots, m_{n'}), \{L(M_{1,j,k}, \ldots, M_{n',j,k})\}_{j\in[n],k\notin INX_j})$ to P_r.
  - Set the number of votes for $L(m_1, \ldots, m_{n'})$ to be number of j's for which $L(\bar{K}_{1,j,k}, \ldots, \bar{K}_{n',j,k}) \neq L(K_{1,j,k}, \ldots, K_{n',j,k})$, for some $k \in INX_j$. For all other j's, P_j^{l+1} sends $L(K_{1,j,k}, \ldots, K_{n',j,k})$ for $k \notin INX_j$ to P_r. If $\mathsf{MAC}_{L(K_{1,j,k},\ldots,K_{n',j,k})}(L(m_1, \ldots, m_{n'})) = L(M_{1,j,k}, \ldots, M_{n',j,k})$ for any one of the recovered values, then increment the vote by 1. If there are $t+1$ votes, output $L(m_1, \ldots, m_{n'})$.

Figure 29: Protocol $\Pi_{\mathsf{ITSigClassical}}$

Public Parameters: Sender P_S and a receiver set $\mathcal{P}_{\mathcal{R}}$.
Secret Inputs: P_S holds messages $(m_1, \ldots, m_{n'}) \in M$. In addition, there is a public value L agreed upon by honest $P_i \notin \mathcal{I}$ where $L : M^k \mapsto M$ is a linear operator.

Setup: Upon receiving $(\mathsf{SetupDC}, sid, (m_1, \ldots, m_{n'}))$ from P_S, store $(m_1^*, \ldots, m_{n'}^*) = (m_1, \ldots, m_{n'})$ if $P_S \notin \mathcal{I}$. Otherwise, receive $(m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$ from the (ideal) adversary $\mathcal{S}$ and set $(m_1^*, \ldots, m_{n'}^*) = (m_{\mathcal{S},1}, \ldots, m_{\mathcal{S},n'})$.
Reveal: If all honest $P_i \notin \mathcal{I}$ send $(\mathsf{RevealDC}, sid, L, \mathcal{P}_{\mathcal{R}})$, then:

- If $P_S \notin \mathcal{I}_0$, then send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_{\mathcal{R}}$. If $\mathcal{P}_{\mathcal{R}} = P_i$, some i, then send $L(m_1^*, \ldots, m_{n'}^*)$ to the (ideal) adversary $\mathcal{S}$.
- If $P_S \in \mathcal{I}$, then receive from (ideal) adversary $\mathcal{S}$ a boolean value $\mathsf{open} \in \{0, 1\}$. If $\mathsf{open} = 1$, send $L(m_1^*, \ldots, m_{n'}^*)$ to all parties in $\mathcal{P}_{\mathcal{R}}$, else deliver $\perp$.

Figure 30: Distributed Commitment Functionality $\mathcal{F}_{\mathsf{DistCommitClassical}}$

Public Inputs: Committer P_S, Share holders and verifiers, $P_1, \cdots, P_n$
Secret Inputs: P_S holds input $(m_1, \ldots, m_{n'}) \in M$.

Setup Phase $\mathsf{DistCommit.Setup}$**:**

- For all $i \in [n']$, P_S computes $\mathsf{Sh}(m_i)$, say $(m_{i,1}, \cdots, m_{i,n})$ for $n \leq 2t+1$. It then sends $(\mathsf{SetupSig}, sid, (m_{1,j}, \ldots, m_{n',j}))$ to $\mathcal{F}_{\mathsf{ITSigClassical}}$ with P_j as the intermediate receiver.

Reveal Phase $\mathsf{DistCommit.Reveal}$**:**

- Each P_i for $i \in [n]$, sends $(\mathsf{RevealSig}, sid, L, \mathcal{P}_{\mathcal{R}})$ to $\mathcal{F}_{\mathsf{ITSig}}$ to reveal $l_j = L(m_{1,j}, \ldots, m_{n',j})$. Interpolate through valid l_j to obtain a degree-t polynomial. Output it's constant term if it succeeds, else output $\perp$.

Figure 31: Protocol $\Pi_{\mathsf{DistCommitClassical}}$

Public Parameters: The space $\mathcal{R}$ of $\mathcal{F}_{\mathsf{Beacon}}$ outputs.
Sample: Upon receives (Sample, sid) from all honest $P_i^l \in \mathcal{C}_l \cap P_i^l \notin \mathcal{I}_l$.
Deliver: $\mathcal{F}_{\mathsf{Beacon}}$ stores samples $r \xleftarrow{\$} \mathcal{R}$ and delivers r to all $P_i^{l'} \in \mathcal{C}_{l'}$.

Figure 32: Functionality $\mathcal{F}_{\mathsf{Beacon}}$

B Proofs for Perfectly Secure Rumors MPC protocols

In this section we present the proofs for the protocols constructed as a part of the perfectly secure Rumors MPC of Section 3.

B.1 Proof of Reshare Protocol

Here, we present a full proof for Lemma 2. We first describe the following simulator $\mathcal{S}$.

1. It emulates t instances of the Random Sharing functionality $\mathcal{F}_{\mathsf{RandomShare}}$ by running the code as is, to create sharings $\langle \alpha_k, f_{\alpha_k}(x)\rangle_l$ for each $k \in [t]$. It sends shares $\{f_{\alpha_k}(j)\}_{k\in[t]}$ to all parties $P_j^l \in \mathcal{I}_l$.
2. It receives from $\mathcal{F}_{\mathsf{Reshare}}$, the values $s'_i = f'(i)$ for all corrupt parties $P_i^{l+1} \in \mathcal{I}_{l+1}$. Then, sample a random degree-t polynomial $f'_{s_i}(.)$ such that $f'_{s_i}(0) = s_i$ and set $s'_{i,j} = f'_{s_i}(j)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$ and for all $j \in [n]$.
3. On behalf of every $P_i^l \notin \mathcal{I}_l$, it sends $s'_{i,j}$ to $P_j^{l+1} \in \mathcal{I}_{l+1}$.

Figure 33: Simulator $\mathcal{S}$ for the protocol Π_{Reshare}

The correctness is straightforward. By linearity of Shamir secret sharing and security of the random sharing, the shares received by corrupt output clients are identically distributed irrespective of s and $f(x)$. Hence, the view of an adversary can be simulated even if it knows s, $f(x)$. It is also easy to observe that this protocol is secure under parallel composition provided we use a parallel random sharing functionality, that can be realised by a parallel VSS protocol.

B.2 Proof of Reinforced Resharing

Here, we present a full proof for Lemma 3. We first describe the following simulator $\mathcal{S}$.

1. It emulates t instances of the Random Sharing functionality $\mathcal{F}_{\mathsf{RandomShare}}$ by running the code as is, to create sharings $\langle \alpha_k, f_{\alpha_k}(x)\rangle_l$ for each $k \in [t]$. It sends shares $\{f_{\alpha_k}(j)\}_{k\in[t]}$ to all parties $P_j^l \in \mathcal{I}_l$. Additionally, create a sharing for a random value s, say $\langle s, f(x)\rangle_l$. Then, for each $i \in [n]$, define
$$\langle s_i, g'_i(x)\rangle_l = \langle s, f(x)\rangle_l + \sum_{k=1}^{t} i^k \langle \alpha_k, g_k(x)\rangle_l$$
where $f'(x) = s + \sum_{k=1}^{t} \alpha_k x^k$ and $s_i = f'(i)$.
2. It receives from $\mathcal{F}_{\mathsf{DoubleReshare}}$ the following values. $(f'(j), \{f'_{s_i}(j)\}_{P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_l})$. Then, sample a random degree-t polynomial $f'_{s'_i}(.)$ such that $f'_{s'_i}(0) = s'_i$ and set $s'_{i,j} = f'_{s'_i}(j)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$ and for all $j \in [n]$.
3. On behalf of every $P_i^l \notin \mathcal{I}_l$, it sends $s'_{i,j}$ to $P_j^{l+1} \in \mathcal{I}_{l+1}$.
4. For $i \in [n]$, when each $P_j^l \in \mathcal{I}_l$ sends $(\mathsf{Reshare}, sid, g'_i(j))$ to $\mathcal{S}$ to obtain $\langle s_i, f'_i(x)\rangle_{l+1}$, it samples a random degree-t polynomial $f'_{s'_{i,j}}(.)$ such that $f'_{s'_{i,j}}(0) = f'_{s_i}(j)$ and set $s'_{i,j,k} = f'_{s'_{i,j}}(k)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$ and for all $k \in [n]$. it sends $s'_{i,j,k}$ to $P_k^{l+1} \in \mathcal{I}_{l+1}$.

Figure 34: Simulator $\mathcal{S}$ for the protocol $\Pi_{\mathsf{DoubleReshare}}$

The correctness and the security of the above protocol follows directly from invoking the resharing functionality in parallel. In addition, it is also easy for the same reason to see that the security is preserved under parallel composition.

B.3 Proof of Multiplication with Helper Party

Here, we present a full proof for Lemma 4. We first describe a simulator $\mathcal{S}$ and show that the joint distribution of the outputs of the adversary and the honest parties in the real(hybrid) world where the

adversary interacts with Protocol $\Pi_{\mathsf{MultHelper}}$ and functionalities $\mathcal{F}_{\mathsf{VSS}}$ and $\mathcal{F}_{\mathsf{DoubleReshare}}$ is identical to that in the ideal world where the adversary interacts with $\mathcal{F}_{\mathsf{MultHelper}}$. We write the simulator for two cases, one corresponding to the the helper party P_h^l being corrupt and the other when it is honest.

1. Based on the helper P_h^l being honest or corrupt, $\mathcal{S}$ does the following:
   - If $P_h^l \in \mathcal{I}_l$: The simulator $\mathcal{S}$ receives $\{(\mathsf{Share}, sid, f_{\alpha_k})\}_{k\in[t]}$, $\{(\mathsf{Share}, sid, f_{\beta_k})\}_{k\in[t]}$, and $\{(\mathsf{Share}, sid, f_{\gamma_k})\}_{k\in[0,2t]}$ from P_h^l and emulates $4t$ instances of the VSS functionality $\mathcal{F}_{\mathsf{VSS}}$ by running the code as is, to create sharings $\langle \alpha_k, f_{\alpha_k}(x)\rangle_l, \langle \beta_k, g_{\beta_k}(x)\rangle_l$ for each $k \in [t]$ and $\langle \gamma_k, h_{\gamma_k}(x)\rangle_l$ for each $k \in [0, 2t]$. It sends shares $\{f_{\alpha_k}(j), g_{\beta_k}(j)\}_{k\in[t]}$, $\{h_{\gamma_k}(j)\}_{k\in[0,2t]}$ to all parties $P_j^l \in \mathcal{I}_l$. It sets $\gamma := \gamma_0$. Additionally, it receives f, g from the ideal functionality. Set $\alpha = f(0), \beta = g(0)$. Then, it creates a sharing $\langle \alpha, f_\alpha\rangle_l, \langle \beta, f_\beta\rangle_l$.
   - If $P_h^l \notin \mathcal{I}_l$: The simulator $\mathcal{S}$ samples $\alpha_k \leftarrow_{\$} \mathbb{F}, \beta_k \leftarrow_{\$} \mathbb{F}$ for each $k \in [t]$. Define $f_\alpha(x) = \alpha + \sum_{k=1}^{t} \alpha_k x^k$, $g_\beta(x) = \beta + \sum_{k=1}^{t} \beta_k x^k$. Let $h_{\gamma_0}(x) = f_\alpha(x) \cdot g_\beta(x) = \sum_{k=0}^{2t} \gamma_k x^k$ and emulates $4t$ instances of $\mathcal{F}_{\mathsf{VSS}}$ as dealer to to create sharings $\langle \alpha_k, f_{\alpha_k}(x)\rangle_l, \langle \beta_k, g_{\beta_k}(x)\rangle_l$ for each $k \in [t]$ and $\langle \gamma_k, h_{\gamma_k}(x)\rangle_l$ for each $k \in [0, 2t]$. It sends shares $\{f_{\alpha_k}(j), g_{\beta_k}(j)\}_{k\in[t]}$, $\{h_{\gamma_k}(j)\}_{k\in[0,2t]}$ to all parties $P_j^l \in \mathcal{I}_l$. It sets $\gamma := \gamma_0$. Additionally, it samples α, β such that $\gamma = \alpha \cdot \beta$. Then, it creates a sharing $\langle \alpha, f_\alpha\rangle_l, \langle \beta, f_\beta\rangle_l$.
2. On behalf of all $P_i^l \notin \mathcal{I}_l$, it sends the following values to all $P_j^{l+1} \in \mathcal{I}_{l+1}$:
$$\hat{f}_\alpha(j) := f_\alpha(i) + j f_{\alpha_1}(i) + \cdots + j^t f_{\alpha_t}(i)$$
$$\hat{g}_\beta(j) := g_\alpha(i) + j g_{\beta_1}(i) + \cdots + j^t g_{\beta_t}(i)$$
$$\hat{h}_\gamma(j) := h_{\gamma_0}(i) + j h_{\gamma_1}(i) + \cdots + j^{2t} h_{\gamma_{2t}}(i)$$
3. It receives from all $P_i^l \in \mathcal{I}_l$ the following values corresponding to all $P_j^{l+1} \notin \mathcal{I}_{l+1}$:
$$\hat{f}_\alpha(j) := f(i) + j f_{\alpha_1}(i) + \cdots + j^t f_{\alpha_t}(i)$$
$$\hat{g}_\beta(j) := g(i) + j g_{\beta_1}(i) + \cdots + j^t g_{\beta_t}(i)$$
$$\hat{h}_\gamma(j) := h_{\gamma_0}(i) + j h_{\gamma_1}(i) + \cdots + j^{2t} h_{\gamma_{2t}}(i)$$
   It then extracts $f(i), g(i)$ for all $P_i^l \in \mathcal{I}_l$ from the above equations to patch it's sharing of $\langle \alpha, f_\alpha\rangle, \langle \beta, g_\beta\rangle$ such that $f_\alpha(i) = f(i)$ and $g_\beta(i) = g(i)$ for all $P_i^l \in \mathcal{I}_l$. Patch $\langle \alpha, \hat{f}_\alpha\rangle, \langle \beta, \hat{g}_\beta\rangle$ accordingly too.
4. Based on the helper P_h^l being honest or corrupt, $\mathcal{S}$ does the following:
   - If $P_h^l \in \mathcal{I}_l$: It sends $(\mathsf{MultHelpShare}, sid, \gamma')$ on behalf of P_h^l to $\mathcal{F}_{\mathsf{MultHelper}}$ and receives the output $(\mathsf{OutMultHelper}, sid, h(i))$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$ or $(\mathsf{OutMultHelper}, sid, \perp)$ based on the value of γ'. It uses these to patch it's sharing of $\langle \gamma_0, h_{\gamma_0}\rangle$ such that $\hat{h}_\gamma(i) = h(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$.
   - If $P_h^l \notin \mathcal{I}_l$: It then sends $(\mathsf{MultHelp}, sid, f(i), g(i))$ on behalf of party $P_i^l \in \mathcal{I}_l$ and receives the output $h(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$. It uses these to patch it's sharing of $\langle \gamma_0, h_{\gamma_0}\rangle$ such that $\hat{h}_\gamma(i) = h(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$.
5. Upon receiving $(\mathsf{DoubleReshare}, sid, h_{\gamma_0}(i))$ from all $P_i^l \in \mathcal{I}_l$, emulates an instances of the $\mathcal{F}_{\mathsf{DoubleReshare}}$ functionality by running the code as is and provide corresponding shares to $P_i^{l+1} \in \mathcal{I}_{l+1}$.
6. If for any $P_i^{l+1} \notin \mathcal{I}_{l+1}$, it holds that $\hat{f}(i) \cdot \hat{g}(i) \neq \hat{h}(i)$, then $\mathcal{S}$ broadcasts a complaint.
7. For any $P_j^{l+1} \in \mathcal{I}_{l+1}$ that has raised a complaint, it makes public $\hat{f}_\alpha(j), \hat{g}_\beta(j), \hat{h}_\gamma(j)$. For all complaints raised, if $\hat{f}_\alpha(i) \cdot \hat{g}_\beta(i) = \hat{h}_\gamma(i)$ it outputs $\hat{h}_\gamma(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$. Otherwise it outputs $\perp$.

Figure 35: Simulator $\mathcal{S}$ for the protocol $\Pi_{\mathsf{MultHelper}}$

We observe that when we have an honest helper, in the entire protocol execution, the adversary can only know upto t shares of α, β. In addition, through the output of the VSS functionality, it can also learn upto t evaluations of f_{α_k}, g_{β_k} for $k \in [t]$ and t evaluations of h_{γ_k} for $k \in [0, 2t]$. It also learns t evaluations of $\hat{f}_\alpha, \hat{g}_\beta, \hat{h}_\gamma$, and t reinforced reshares and t-subshares of each of these reshares of γ. Thus, the adversary's view is identically distributed, irrespective of α, β. Additionally, observe that in the case of an honest helper, the honest parties in $\mathcal{C}_{l+1}$, will have shares of the product $\alpha\beta$, due to the linearity of Shamir sharing and correctness of the reinforced resharing. In the case of a dishonest helper party, note that it gets disqualified iff $\hat{h}(x) \neq \hat{f}(x)\hat{g}(x)$. As far as the parallel execution of this protocol is concerned, since a dishonest helper commits to the coefficients of these polynomials during the VSS execution in $\mathcal{C}_l$, it's disqualification in $\mathcal{C}_{l+1}$ can have no correlation to a parallel execution of

this protocol with an honest helper. So, this along with a parallel reinforced resharing functionality ensures secure parallel execution of the $\Pi_{\mathsf{MultHelper}}$ protocol.

B.4 Proof of Multiplication

Here, we present a full proof for Lemma 5. We first describe a simulator $\mathcal{S}$ and show that the joint distribution of the outputs of the adversary and the honest parties in the real(hybrid) world where the adversary interacts with Protocol $\Pi_{\mathsf{MultHelper}}$ and functionalities $\mathcal{F}_{\mathsf{MultHelper}}$ and $\mathcal{F}_{\mathsf{DoubleReshare}}$ is identical to that in the ideal world where the adversary interacts with $\mathcal{F}_{\mathsf{MultCDM}}$.

1. For every $P_i^l \in \mathcal{I}_l$, receive from the (ideal) functionality $\mathcal{F}_{\mathsf{MultCDM}}$ the values $(f(i), g(i), \{f_j(i), g_j(i)\}_{j\in[n]})$. Additionally, for every $P_i^{l+1} \in \mathcal{I}_{l+1}$, receive $h(i)$ from the (ideal) functionality.
2. For each $P_i^l \in \mathcal{I}_l$ that sends $(\mathsf{MultHelpShare}, \mathsf{sid}, \gamma_i)$ and every $P_j^l \in \mathcal{I}_l$ that sends $(\mathsf{MultHelp}, f_k(j), g_k(j))$ for all $k \in [n]$, emulate n instances of the $\mathcal{F}_{\mathsf{MultHelper}}$ functionality, by using γ_i as the value when the helper node is corrupt, or a random γ_i otherwise to send $h'(i)$ for some polynomial h' such that $h'(i) = \gamma_i$ or $\perp$ to all $P_i^{l+1} \in \mathcal{I}_{l+1}$ as per the code of the functionality,ie.. check if $\gamma_i = f(i) \cdot g(i)$ for corrupt helpers. Using the random γ_i chosen on behalf of the honest helpers, let γ be the reconstruction of these shares.
3. For all $P_i^{l+1} \in \mathcal{I}_{l+1}$, that sends $(\mathsf{DoubleShare}, sid, h'(i))$, output a sharing on polynomial h such that $h(i)$ matches the value sent as the output by the (ideal) functionality and $h(0) = \gamma$.

Figure 36: Simulator $\mathcal{S}$ for the protocol Π_{MultCDM}

The security of the multiplication protocol is straightforward as we invoke n parallel instances of $\mathcal{F}_{\mathsf{MultHelper}}$ functionality, and a reinforced resharing functionality. It is also inherits security under parallel composition from these functionalities.

C Extra Protocols for Statistically Secure Rumors MPC

In this section, we present extra protocols used in our Statistically Secure Rumors MPC of Section 4.

C.1 Resharing Protocol

One can use the resharing protocol as shown in the case for perfectly secure case, by instantiating a VSS protocol in $\mathcal{C}_{l-1}$ to create enough random shares in $\mathcal{C}_l$ to locally compute and send reshares at $\mathcal{C}_{l+1}$. Protocol Π_{Reshare} for the statistically secure case is presented in Figure 37.

Public Parameters: Sending Committee $\mathcal{C}_l$, Receiving Committee $\mathcal{C}_{l+1}$.
Secret Inputs: $\langle s, f(x)\rangle_l$.
Setup: $\langle \alpha_k, f_k(x)\rangle_l \leftarrow \$$ for $k \in [t]$
Output: $\langle s, f'(x)\rangle_{l+1}$, where $f'(x) = s + \sum_{k=1}^{t} \alpha_k x^k$.

Committee $\mathcal{C}_l$:
- Let $s_j = f(j)$ be the share of s held by P_j^l. For each $k \in [t]$, let $\alpha_{k,j} = f_k(j)$ be the share of α_k held by P_j^k. Then, each P_j^l sends $s'_{i,j} = s_j + \sum_{k=1}^{t} i^k \alpha_{k,j}$ to P_i^{l+1} for each $i \in [n]$ by sending $(\mathsf{VSSReconstruct}, sid, L_i, P_i^{l+1})$ to $\mathcal{F}_{\mathsf{StatVSS}}$ where L_i corresponds to the linear operator corresponding to the polynomial evaluation at i as described above.

Committee $\mathcal{C}_{l+1}$: For $i \in [n]$, P_i^{l+1} stores $s'_i = \mathsf{Rec}(s'_{i,1}, \ldots, s'_{i,n})$ as its share of s.

Figure 37: Protocol Π_{Reshare} in the $\mathcal{F}_{\mathsf{StatVSS}}$ hybrid model

D Proofs for Statistically Secure Rumors MPC

In this section, we present the proofs for the protocols constructed as part of the statistically secure Rumors MPC of Section 4.

D.1 Proof for Conditional Future Broadcast

Here we present a proof for Lemma 7

Proof (Proof Sketch for Lemma 7).

Case 1 : For all $P_S \in \mathcal{C}_l \setminus \mathcal{I}_l$

1. For $i \in [n']$, compute independent zero sharings $(m_{i,1}, \ldots, m_{i,n}) = \mathsf{Sh}(0)$. For $i \in [n']$, $j, k \in [n]$, sample $K_{i,j,k}$, compute $M_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m_{i,j})$. For $i \in [n']$, and $P_j^{l+1} \in \mathcal{I}_{l+1}$, send $(m_{i,j}, \{M_{i,j,1}, \ldots, M_{i,j,n}\}, \{K_{i,1,j}, \ldots, K_{i,n,j}\})$ to P_j^{l+1}.
2. If $\mathcal{P}_{\mathcal{R}} = \mathcal{C}_{l+1}$(or $\mathcal{P}_{\mathcal{R}} = P_j^{l'} \in \mathcal{I}_{l'}$ resp.), receive from the ideal functionality $\mathcal{F}_{\mathsf{FB}}$ the value $L(m_1, \ldots, m_{n'})$. Sample $m'_1, \ldots, m'_{n'}$ such that $L(m_1, \ldots, m_{n'}) = L(m'_1, \ldots, m'_{n'})$ and patch each of the earlier zero sharings to be a sharing of m_i, say $(m'_{i,1}, \ldots, m'_{i,n}) = \mathsf{Sh}(m'_i)$ such that $m'_{i,j} = m_{i,j}$ for all $P_j^{l+1} \in \mathcal{I}_{l+1}$ and computing corresponding tags $M'_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m'_{i,j})$. On behalf of the honest parties $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, send $(L(m'_{1,j}, \ldots, m'_{n',j}), L(M'_{1,j,k}, \ldots, M'_{n',j,k})$ to all $P_j^{l+1}, P_k^{l+1} \in \mathcal{I}_{l+1}$. Also, broadcast on behalf of all the honest parties their keys.

Case 2 : For all $P_S \in \mathcal{I}_l$

1. In the setup phase, on behalf of each honest party $P_j^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, receive $(m_{i,j}, \{M_{i,j,1}, \ldots, M_{i,j,n}\}, \{K_{i,1,j}, \ldots, K_{i,n,j}\})$.
2. In the broadcast phase, broadcast received messages and tags, followed by keys in the next round as in the protocol. Look at the broadcasted messages, tags by the corrupt parties and decide on the set of $t+1$ tags that are consistent with the keys. Use the shares corresponding tags to compute the linear combination to be l', or $\perp$ if there is not enough tags. Send l' or $\perp$ to the ideal functionality as $l_{\mathcal{S},j}$ based on that.

Figure 38: Simulator $\mathcal{S}$ for the protocol Π_{FB}

We can see that Π_{FB} satisfies the requirement that for an honest sender P_S with inputs $(m_1, \cdots, m'_n)$, given that $(\mathsf{Sh}, \mathsf{Rec})$ is a (t, n) secret sharing scheme, $\mathsf{FB.Share}$ results in distributing sufficiently many shares in $\mathcal{C}_{l+1}$ to allow for correct reconstruction. For an honest P_S, there would be at least $t+1$ honest parties in $\mathcal{C}_{l+1}$, who would broadcast correct shares and tags, for which the verification condition would pass, as there would be $t+1$ keys also broadcasted honestly. For any party $P_i^{l+1} \in \mathcal{I}_{l+1}$ which broadcasts $L(m^*_{1,j}, \ldots, m^*_{n',j}) \neq (L(m_{1,j}, \ldots, m_{n',j})$ and $(\{M^*_{i,j,1}, \cdots, M^*_{i,j,n}\}) \neq (\{M_{i,j,1}, \cdots, M_{i,j,n}\})$, the verification condition under the broadcasted keys would fail except for negligible probability. The privacy also holds due to the fact that the adversary controls upto t parties in $\mathcal{C}_{l+1}$

More specifically, to simulate the view of the adversary $\mathcal{S}$, for sharings sent by an honest sender $P_{S,i} \notin \mathcal{I}_l$, we could simply compute a sharing of 0, and provide the adversary with these shares of all parties $P_i^l \in \mathcal{I}_{l+1}$. For a dishonest sender $P_{S,i} \in \mathcal{I}_l$, receive from the adversary, $\mathcal{S}$ the shares to be sent to each party $P_j^{l+1} \in \mathcal{C}_{l+1}$. During the reveal phase, for an honest sender $P_{S,i} \notin \mathcal{I}_l$, receive $L_i(m_{i,1}, \ldots, m_{i,n'})$, and let the adversary change the messages for any dishonest sender $P_{S,i} \in \mathcal{I}_l$ by appropriately setting the shares of the corrupt parties in $\mathcal{C}_{l+1}$ as needed to reveal $l^*_{\mathcal{S},i}$.

D.2 Proof for Information Theoretical Signature

Here we present a proof of Lemma 8.

Proof (Proof Sketch of Lemma 8).

One could describe the simulator for the view of adversary $\mathcal{S}$ based on the following cases, say in the case of a single message m being sent by a sender through Π_{ITSig}:

Case 1 - $P_S \in \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$: As the sender is the party providing the input, the simulator can emulate all honest parties and ideal functionality $\mathcal{F}_{\mathsf{FB}}$, and interact with the $\mathcal{S}$. It will finally output whatever $\mathcal{S}$ wants it to output. In such a case, the real and ideal executions are identical, except that the output of honest parties m in the ideal execution are given by the functionality $\mathcal{F}_{\mathsf{ITSig}}$ based on what one of the emulated honest parties would have computed. We will always have that the views of all honest parties in the reveal phase is identical as all messages are broadcast messages, and hence the joint distribution of output of simulator and honest parties together in the real execution is perfectly indistinguishable from that of the output of $\mathcal{S}$ and honest parties.

Case 1 - $P_S \in \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$:

1. On behalf of every honest party $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, receive from P_S keys $\{K_{i,j,k}\}_{i\in[n'],k\in[\kappa]}$. For these keys, just run the protocol as is by selecting half the keys and broadcasting it. In addition, also receive from the P_S the setup messages on behalf of $\mathcal{F}_{\mathsf{FB}}$, and run the code of the functionality as is.
2. When P_{int} broadcasts $L(m'_1, \ldots, m'_{n'})$ and all corresponding tags in the non-chosen set of indices, check if there are enough consistent tags with the keys published like in the protocol. If there is more than $t+1$ votes, send that to the (ideal) functionality as $l^*_{\mathcal{S}}$.

Case 2 - $P_S \in \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$:

1. Receive $(m_1, \ldots, m_{n'}), \{M_{1,j,k}, \ldots, M_{n',j,k}\}_{j\in[n],k\in[\kappa])}$ on behalf of P_{int}, and also $(\mathsf{Setup}, sid, (m_1, \ldots, m_{n'}))$ and $(\mathsf{Setup}, sid, (K_{1,j,k}, \ldots, K_{n',j,k}))$ on behalf of $\mathcal{F}_{\mathsf{FB}}$ for $j \in [n], k \in [\kappa]$ from P_S. Then send to the (ideal) functionality $\mathcal{F}_{\mathsf{ITSig}}$, $(m_1, \ldots, m_{n'})$.
2. On behalf of all the honest parties $P_j^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$, choose an index set INX_j and send to all the corrupt parties $P_j^{l+1} \in \mathcal{I}_{l+1}$, the keys $K_{i,j,k}$ for $i \in [n'], j \in [n], k \in INX_j$. In addition, emulate the reveal of the $\mathcal{F}_{\mathsf{FB}}$ functionality, by running the code as is, to reveal the keys $\bar{K}_{i,j,k}$.
3. If there is any inconsistent tags corresponding to the revealed keys for any of the sent messages to P_{int}, complain as in the real execution, and emulate the reveal of $\mathcal{F}_{\mathsf{FB}}$ functionality, by running the code as is, to reveal the keys $L(\bar{m}_1, \ldots, \bar{m}_n)$.
4. In the reveal phase, as the sender is honest, and there were no complaints in the setup phase, broadcast $L(m_1, \ldots, m_{n'})$ and the linear combination of the corresponding tags.

Case 3 - $P_S \in \mathcal{C}_l \setminus \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$:

1. $\mathcal{S}$ receives from the ideal functionality $\mathcal{F}_{\mathsf{ITSig}}$, the values $(m_1, \ldots, m_{n'})$ in the setup phase of the ideal execution. It forwards it to $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$, along with corresponding tags and messages sent to all $P_j^{l+1} \in \mathcal{I}_{l+1}$. It then emulates all the honest parties in the reveal phase. If P_{int} chooses to reveal $L(m_1, \ldots, m_{n'})$ in the reveal phase, then forward $\mathsf{reveal} = 1$ to $\mathcal{F}_{\mathsf{ITSig}}$, and 0 otherwise.

Case 4 - $P_S \in \mathcal{C}_l \setminus \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$:

1. Emulate an honest sender in the following way: $\mathcal{S}$ generates keys $K_{i,j,k}, i \in [n'], j \in [n], k \in [\kappa]$ and sends it to all $P_j^{l+1} \in \mathcal{I}_{l+1}$. It emulates all honest parties for the rest of the setup phase with these sampled keys. In the reveal phase, it receives from the (ideal) functionality the value $L(m_1, \ldots, m_{n'})$. It then samples $m'_1, \ldots, m'_{n'}$ such that $L(m_1, \ldots, m_{n'}) = L(m'_1, \ldots, m'_{n'})$, and computes $M'_{i,j,k} = \mathsf{MAC}_{K_{i,j,k}}(m'_i)$. It then broadcasts $L(m'_1, \ldots, m'_{n'})$ and all corresponding tags in the non-chosen set of indices.

Figure 39: Simulator $\mathcal{S}$ for the protocol Π_{ITSig}

Case 2 - $P_S \in \mathcal{I}_l$, $P_{\mathsf{int}} \notin \mathcal{I}_{l+1}$: As again, the sender provides the input to this protocol, the simulator can emulate the view of all the honest parties and ideal functionality $\mathcal{F}_{\mathsf{FB}}$. The adversary provides the inputs m and $K_{i,j,k}$ to different instances of the $\mathcal{F}_{\mathsf{FB}}$ functionality. However, in the setup phase in $\mathcal{C}_{l+1}$, we ask the adversary to reveal $\bar{K}_{i,j,k}$ for the chosen indices. If the adversary outputs the correct keys, and the MAC check succeeds, we input message m input to the functionality as the adversary. However, if a check fails, then we simulate $\bar{m}$ being revealed by $\mathcal{F}_{\mathsf{FB}}$. In the reveal phase, we perform the checks as honest parties do and, if they pass input the message m to the functionality. Otherwise, we input $\perp$.

Case 3 - $P_S \notin \mathcal{I}_l$, $P_{\mathsf{int}} \in \mathcal{I}_{l+1}$: In this case the simulator receives a message m from $\mathcal{F}_{\mathsf{ITSig}}$. The simulator then emulates a sender with message m. For a known input, it is easy to emulate the execution of the protocol as discussed above. As the sender is honest, the message set by a receiver in this case will not get accepted except with negligible probability in the verification as only t key holders can be corrupt in $\mathcal{C}_{l+1}$, and that would mean the message will not get $t+1$ votes.

Case 4 - $P_S \notin \mathcal{I}_l$, $P_{\mathsf{int}} \notin \mathcal{I}_{l+1}$: Correctness is straightforward, and to simulate the view of the adversary $\mathcal{S}$, nothing is revealed by $\mathcal{F}_{\mathsf{FB}}$ in the setup phase of Π_{ITSig} and apart from that, all an adversary might receive is keys sent to corrupt parties in $\mathcal{C}_{l+1}$ and these are sampled at random. In the reveal phase, there will be enough votes for the same message P_S set by sender and received by P_{int} to get accepted.

D.3 Proof for Distributed Commitment

Here we present a proof for Lemma 9.

Proof (Proof Sketch for Lemma 9). We deal with the following two cases:

Case 1 - $P_S \in \mathcal{I}_l$: As the sender is the only one with inputs to the protocol, the simulator can emulate a protocol execution by emulating all honest parties and ideal functionality $\mathcal{F}_{\mathsf{ITSig}}$, and interact with the $\mathcal{S}$ on a sharing of the input $\mathcal{S}$ provides. The view of the adversary in this case in the real vs the ideal world is identically distributed due to the fact that the view from the emulation was perfectly indistinguishable to that in the ideal world. The output of the honest parties are also identically distributed to that of the ones emulated in the simulation as the simulator would have fixed the $t+1$ shares to reveal the linear combination of the same messages that would have been sent by the $\mathcal{S}$ as input to the functionality.

Case 2 - $P_S \notin \mathcal{I}_l$: In this case, the adversary receives a set of shares corresponding to the messages of an honest sender in the setup phase. One could easily simulate it with a sharing of 0, distributed amongst the corrupt parties in $\mathcal{C}_{l+1}$, and rely on the privacy of the secret sharing scheme. In the reveal phase, after the adversary learns the output, we could adjust the shares of secret sharing scheme to match that output. The honest parties will have a sharing of the message committed by an honest sender, and we use the IT MACS to show that dishonest parties in $\mathcal{C}_{l+1}$ only have t shares, that may output to $\perp$. However, the $t+1$ shares of the honest parties is sufficient to reconstruct to the same message as the one committed to by an honest sender in $\mathcal{C}_l$.

D.4 Proof of Statistically Secure VSS

Here we present a proof of Lemma 10.

Proof (Proof Sketch of Lemma 10).

Case 1 : For all $P_D \in \mathcal{C}_l \setminus \mathcal{I}_l$

$\mathcal{S}$ receives from the ideal functionality $\mathcal{F}_{\mathsf{StatVSS}}$ all values $f(i)$ for all $P_i^{l+1} \in \mathcal{I}_{l+1}$. It samples a random polynomial $r(x)$ and emulating all the honest parties and running the protocol as is, by creating a sharing of 0 with the shares of the corrupted parties corresponding to what it received from the (ideal) functionality. In the reveal phase, it patches the underlying secret sharing to share the actual message that is revealed in the reveal phase of the (ideal) execution.

Case 2 : For all $P_D \in \mathcal{I}_l$

1. $\mathcal{S}$ receives from the P_D the values $s_i = f(i), \rho_i = r(i)$ on behalf of all honest $P_i^{l+1} \in \mathcal{C}_{l+1} \setminus \mathcal{I}_{l+1}$. It also receives from P_D the following values, $(\mathsf{SetupDC}, sid, f_i)$ and $(\mathsf{SetupDC}, sid, r_i)$ on behalf of $\mathcal{F}_{\mathsf{DistCommit}}$, which it emulates by running the code as is.
2. It then extracts s_i', ρ_i' from the distributed commitments. It sends $(\mathsf{VSSShare}, sid, f)$ to $\mathcal{F}_{\mathsf{StatVSS}}$. It runs the protocol as is, emulating all honest parties and complaining when they would raise a complaint according to the protocol, basically when $s_i + e\rho_i \neq s_i' + e\rho_i'$. In the reveal phase, it emulates the DC reveal phase by running the corresponding code of the functionality

Figure 40: Simulator $\mathcal{S}$ for the protocol Π_{StatVSS}

It is trivial to simulate and argue the view for corrupt Dealer, as one could emulate the execution of all honest parties by using the shares of the corrupted parties provided by the corrupte dealer. In the case of an honest dealer, what is revealed in the setup phase is a shamir share on the polynomial chosen by a dealer, which can be computed as a sharing on 0, for the dishonest parties in $\mathcal{C}_{l+1}$. As there are only t dishonest parties, the shares held by them are independent of m in both the real and ideal execution. Apart from this, the adversary also learns through the output of $\mathcal{F}_{\mathsf{DistCommit}}$ openings to random values $s_i + e\rho_i$ for all $P_i^{l+1} \notin \mathcal{I}_{l+1}$, where e is output by $\mathcal{F}_{\mathsf{Beacon}}$, which can be sampled.

D.5 Proof of Statistically Secure Multiplication

Here we present a proof for Theorem 12.

Proof (Proof Sketch for Theorem 12).

We first argue the correctness and privacy of Π_{PLM} (in Figure 20) in the $(\mathcal{F}_{\mathsf{StatVSS}}, \mathcal{F}_{\mathsf{Beacon}}, \mathcal{F}_{\mathsf{VSSReshare}})$-hybrid model. In the case where a corrupt party P_i^{l+1} operates as a designated party holding its shares, if the condition $c \neq ab$ holds, then with negligible probability, d will be non-zero. Particularly, there

Case 1 : For all $P_i^l \in \mathcal{C}_l \setminus \mathcal{I}_l$

1. It receives from the ideal functionality the values $(f_a(k), g_b(k), \{f_{a,j}(k), g_{b,j}(k)\}_{j\in[0,t]})$ for all $P_k^l \in \mathcal{I}_l$. Sample random values for a, b and the coefficients $\{a_j, b_j\}_{j\in[0,t]}$. Sample polynomials $f'_a, g'_b, \{f'_{a,j}, g'_{b,j}\}_{j\in[0,t]})$ such that $f'_a(0) = a, g'_b(0) = b, f'_{a,j}(0) = a_j, g'_{b,j}(0) = b_j$ and $f'_a(k) = f_a(k), g'_b(k) = g_b(k), f'_{a,j}(k) = f_{a,j}(k), g'_{b,j}(0) = g_{b,j}(k)$. Set $\gamma = a \cdot b$. For each honest party P_i^l, set $\gamma_i = f'_a(i) \cdot g'_b(i)$ and run the ideal functionality $\mathcal{F}_{\mathsf{AugVSS}}$ using its code as is. Additionally, Run Π_{PLM} as is on $f'_a(i), g'_b(i), \gamma_i$ with P_i^l as the prover. In committee $\mathcal{C}_{l+1}$, output the sharings provided by the ideal functionality as output to $\mathcal{S}$, as output of the protocol execution to the corrupted parties.

Case 2 : For all $P_i^l \in \mathcal{I}_l$

1. Receive from a corrupt P_i^l, $(\mathsf{AugVSSShare}, sid, \gamma_i)$ on behalf of $\mathcal{F}_{\mathsf{AugVSS}}$, and run the code of the functionality as is. If $\gamma_i = f_a(i) \cdot g_b(i)$, where $f_a(i)$ and $g_b(i)$ are received from the (ideal) functionality, notice that running Π_{PLM} on these values will succeed, and otherwise fail. Output $f_a(i), g_b(i)$ for all such i where Π_{PLM} fails. Then relay the output sharings of the corrupted parties recevied from the (ideal) functionality to the corrupted parties.

Figure 41: Simulator $\mathcal{S}$ for the protocol Π_{Mult}

would exist an output e from the beacon that satisfies the condition, although such an occurrence happens only with negligible probability. The correctness of the multiplication protocol is inherited from the correctness of the Augmented VSS functionality, resharing functionality,Π_{PLM} and Lagrange interpolation of the value $a \cdot b$ as mentioned in [GRR98]. As Π_{Mult} only executes Π_{PLM} and $\mathcal{F}_{\mathsf{AugVSS}}$ on the product of it's own shares, one can simulate the view of an adversary it by computing an augmented VSS of 0 for the parties that are honest, and then emulate the protocol execution by taking as input from adversary $\mathcal{S}$ the shares for the corrupt parties as inputs to PLM and Augmented VSS functionality.