# Feasibility of Broadcast with Dynamic Committees

Gabriel Dettling[1], Chen-Da Liu-Zhang[1] ✉, Elisaweta Masserova[2], Matthieu Rambaud[3], and Antoine Urban[4]

[1] Lucerne University of Applied Sciences and Arts
[2] Carnegie Mellon University
[3] Télécom Paris
[4] DFNS

**Abstract.** A significant number of works have considered the problem of multi-party computation over dynamic committees in synchronous networks, including YOSO MPC [Crypto'21], Fluid MPC [Crypto'21], SCALES MPC [TCC'22] and Layered MPC [Crypto'23]. However, prior works assume that every party has access to an ideal synchronous broadcast channel towards the next committee.
While this assumption is partly justified due to the seminal work of Garay [WDAG'94] stating that deterministic broadcast with dynamic committees is impossible, it is open whether there are randomized solutions.
We answer this question in the affirmative, by providing a complete characterization of broadcast with dynamic committees. We use the formalization introduced in the Layered MPC setting and achieve the following results for layered broadcast:

- A **statistically secure** protocol tolerating $t < n/3$ corruptions with no setup.
- A **computationally secure** protocol tolerating $t < n/2$ corruptions, assuming only a bulletin-board PKI for signatures.
- A matching **impossibility result** showing that broadcast is impossible for $t \geq n/2$ corruptions.

Using our broadcast, we achieve the following polynomial-time results:

- YOSO MPC protocols without broadcast (statistical for $t < n/3$ without setup; and computational for $t < n/2$ assuming a plain PKI for signatures).
- Assuming plain PKIs for signatures and public-key encryption, a Layered MPC protocol without broadcast for $t < n/2$.
- Assuming homomorphic commitments, a Layered MPC without broadcast for $(t+1)^2 \leq n$. To achieve this, we introduce a secure-message-transmission protocol for $(t+1)^2 \leq n$ which has linear communication in $\ell$ and polynomial communication in $n$ when transmitting a message across $\ell$ layers. This result is of independent interest.

## 1 Introduction

Multi-party computation (MPC) [68, 46, 6, 25, 65] allows a set of mutually distrusting parties to perform a joint computation over their private inputs in

such a way that nothing about their data beyond the output of the computation is leaked. Most of the MPC protocols proposed in the literature focused on the traditional setting where there is a fixed set of participants that stay online throughout the computation, and security is guaranteed as long as up to a threshold of the parties is compromised. However, this static model is not well suited for capturing modern large-scale applications that are long-lived (such as blockchains or training federated learning models), and operate in environments where parties may join and leave the computation.

To address these limitations, several models for MPC with dynamic committees have recently been proposed, including the You-Only-Speak-Once model (YOSO) MPC [43], Fluid MPC [27], Layered MPC [29] and SCALES [4] models (see [57] for a discussion). In these models, computation is structured as a sequence of rounds, where in each round a different set of active parties, a *committee*, sends messages towards future committees. Security is guaranteed as long as up to a threshold of the parties is compromised in each committee.

Protocols in these models capture the dynamism of MPC with changing committees. Unfortunately, to the best of our knowledge, all known protocols assume an ideal broadcast channel: whenever a party in one committee sends a message, it is reliably and consistently delivered to the next committee (and often implicitly to all future committees[5]). This raises the fundamental question of whether broadcast itself can be achieved given dynamic committees.

This question was previously studied in the consensus and blockchain literature [26, 15, 40, 60, 1, 50, 62] under so-called player-replaceable protocols[6]. Unfortunately, all player-replaceable broadcast protocols assume a trusted setup (a threshold signature setup, threshold verifiable random functions, or sharings of random values to generate unpredictable common randomness), which is a very strong assumption since realizing such setup in an environment where the set of parties is dynamic is challenging. This motivates our central question:

*Is it possible to achieve broadcast assuming point-to-point channels in the setting of dynamic committees without trusted setup?*

It is worth noting that the seminal work of Garay [41] showed that a deterministic broadcast protocol is impossible in the setting of dynamic committees even assuming that there is only one single corruption in each committee (see also [29]). This partly justified the assumption of a broadcast channel in prior works, hinting that it may be a necessary assumption. As we show, however, this impossibility can be circumvented using randomization.

[5] This is without loss of generality assuming honest majority per committee, since one can always broadcast to the next committee and further propagate the message towards future committees by forwarding and taking majority decisions.

[6] The original motivation of player-replaceability and YOSO was to avoid strong adaptive DDoS attacks: If a secretly-chosen committee only speaks once, corrupting it after it spoke does not harm the protocol since the committee already did their job.

### 1.1 Our Contributions

We give a complete characterization of broadcast protocols for dynamic committees. We use the Layered MPC formalism [29, 33] (equivalent to Fluid MPC with maximal fluidity [27]) as it cleanly models the dynamic committee setting, but our protocols work under other formalisms for MPC with dynamic committees.

In Layered MPC parties are placed in a communication network that has the shape of a layered graph, with $n$ parties in each layer (representing one committee). A party in any layer has access to point-to-point secure channels that allow them to securely communicate to each party in the immediate next layer. Contrary to the original formulation, we do not assume broadcast channels. This means that parties in any layer simply receive private messages from the parties in the previous layer, and send private messages to the parties in the next layer. The adversary is restricted to corrupting a subset of up to $t$ out of $n$ parties in each layer, a specialization of the notion of general adversary structure [51].

We aim to realize broadcast over a layered network, allowing a sender in $\mathcal{L}_0$ to consistently send a message towards all parties belonging to a final layer $\mathcal{L}_d$.

*Result 1.* We introduce the first broadcast protocol for $t < n/3$ in the Layered MPC setting without any setup and while achieving statistical security.

**Theorem 1.** *Let $\lambda$ be a security parameter. There exists a statistical layered broadcast protocol for $t < n/3$ with $\mathcal{O}(\lambda n^{19})$ communication and $\mathcal{O}(\lambda)$ rounds.*

*Result 2.* As broadcast for $t \geq n/3$ requires some form of setup [55], we next explore the setting where parties have bulletin-board PKI for signatures[7].

**Theorem 2.** *Let $\lambda$ be a security parameter. Assuming a bulletin-board PKI for signatures, there exists a layered broadcast protocol for $t < n/2$ with $\mathcal{O}(\lambda n^7)$ communication and $\mathcal{O}(\lambda)$ rounds.*

*Result 3.* We then obtain a matching result by observing that layered broadcast is impossible for $t \geq n/2$. This is in contrast to the setting with a fixed number of parties, where broadcast is possible for any $t < n$ [35].

**Theorem 3.** *There is no layered broadcast protocol for $t \geq n/2$.*

Our broadcast protocols can be plugged into existing MPC protocols with dynamic committees. Plugging our broadcast protocols into existing (abstract) YOSO MPC protocols [43] leads to the first (abstract) YOSO MPC protocols without broadcast and with guaranteed output delivery, as stated by the following corollary.

**Corollary 1.** *There are YOSO MPC polynomial protocols with guaranteed output delivery without assuming broadcast for 1) $t < n/3$ corruptions with statistical security and no setup, and 2) $t < n/2$ corruptions assuming a plain PKI.*

[7] In a bulletin-board PKI the keys from corrupted parties can be chosen adversarially. See [17] for a nice discussion.

The abstract YOSO MPC model [43, 22] is similar to Layered MPC / maximally Fluid MPC, where parties are denoted *roles* and execute a single protocol step, and each role assumes that channels to all future parties are available (instead of only to the immediate next committee), as noted in [57]. We further note that [22] presents a compiler using non-committing encryption, that compiles any abstract YOSO MPC into an adaptively secure YOSO MPC protocol in the real world (a.k.a. natural YOSO), by instantiating appropriate committee-election and communication channel protocols.

Our broadcast protocols can also be plugged into existing protocols in the Fluid and Layered MPC settings [29, 33]. A subtlety, however, is that plugging in a broadcast protocol introduces a number of *new intermediate layers* equal to the round complexity of broadcast. Since in Fluid and Layered MPC each committee $\mathcal{L}_k$ only has access to channels towards the immediate next committee $\mathcal{L}_{k+1}$, and our broadcast protocols require a number of rounds proportional to the security parameter $\lambda$, any secret message that was being transmitted from $\mathcal{L}_k$ to $\mathcal{L}_{k+1}$ now needs to be transmitted from $\mathcal{L}_k$ to $\mathcal{L}_{k+\Theta(\lambda)}$. This is not a problem in the YOSO MPC model (since channels to future committees are available), or if we assume a plain PKI for public-key encryption (since transmitting a message towards a specific party in a future committee can be done simply by encrypting under the receiver's public key, and forwarding the message until it reaches the receiver). This observation leads to the following theorem

**Corollary 2.** *Assuming bulletin-board PKIs for signatures and public-key encryption, there is a maximally Fluid / Layered MPC polynomial protocol with guaranteed output delivery without assuming broadcast for $t < n/2$ corruptions.*

However, without public-key encryption, all known protocols for secure message transmission (SMT) from point-to-point secure channels in the maximally Fluid and Layered MPC settings have exponential communication in both the number of committees and the committee size when transmitting messages across $\Theta(\lambda)$ layers (we need this because our broadcast incurs $\Theta(\lambda)$ rounds).

To mitigate this, we introduce the first SMT protocol that incurs a communication complexity that increases only linearly with the number of committees (its communication complexity is still exponential in the committee size).

**Theorem 4.** *Assuming non-interactive homomorphic commitments, there is an SMT protocol that transmits a message across $\ell$ committees with communication $\mathcal{O}(\ell \cdot \binom{n}{t+1} \cdot n^3)$ that is secure up to $t < n/2$ corruptions.*

Based on this protocol, which we denote $\mathsf{SMT}$, we introduce a variation of it, called $\mathsf{SMT}^*$, which works for $(t+1)^2 \leq n$ and only incurs polynomial communication. To the best of our knowledge, all layered SMT protocols that can transmit a message $\Theta(\lambda)$ layers incur exponential communication even when considering only fail-stop corruptions and a constant number of corruptions.

**Corollary 3.** *Assuming non-interactive homomorphic commitments, there is an SMT protocol that transmits a message across $\ell$ committees with polynomial communication $\mathcal{O}(\ell \cdot n^{\frac{3}{2}})$ that is secure up to $t \leq \sqrt{n} - 1$ corruptions.*

By integrating our secure message transmission protocol and our statistical broadcast protocol into the layered MPC protocol in [29], we obtain the first Layered MPC protocol without assuming setup and without a broadcast channel.

**Corollary 4.** *Assuming non-interactive homomorphic commitments, there is a maximally Fluid / Layered MPC polynomial protocol with guaranteed output delivery without assuming broadcast secure up to $t \leq \sqrt{n} - 1$ corruptions.*

### 1.2 Related Work

Byzantine Broadcast [56, 35, 64, 24] is one of the fundamental building blocks in cryptography and distributed protocols, and a significant line of work has been devoted to exploring the limits of its round complexity [35, 39, 53, 26, 60, 2, 40, 44, 32, 45] and communication complexity [8, 15, 31, 11, 14, 10].

Several works design player-replaceable protocols that operate with dynamically chosen committees [26, 15, 49, 48] to achieve adaptive security or improve the communication pattern. Protocols for consensus and broadcast with player-replaceability have been proposed [26, 15, 40, 60, 1, 50, 62], but to the best of our knowledge, all of them require a trusted setup, typically in the form of a unique threshold signatures, verifiable random functions or a sequence of Shamir sharings of random values, to generate unpredictable randomness.

Since then, a broad spectrum of models [43, 27, 29, 66, 4, 38] have been proposed to capture the notion of *multi-party computation with dynamic committees*. We summarize the models and their recent developments in the area of MPC with dynamic committees (see also [57]). To the best of our knowledge, all known protocols in these models assume ideal broadcast channels.

**Fluid MPC [27] and Layered MPC [29].** Fluid MPC considers a scenario where the parties in a committee can only send private messages to the next committee, and where there are up to $t$ corruptions in each committee. Each party is required to participate in a certain minimum number of consecutive committees, a parameter referred to as the protocol's *fluidity*. Layered MPC is as a bare-bones version of the *maximally* fluid MPC[8], which ignores some of the additional features of the Fluid MPC framework (such as how to sample the committees from the overall set of parties). The original Fluid MPC [27] introduced a maximally Fluid statistical protocol for security with abort under honest majority. Bienstock et al. improved the communication complexity of maximally Fluid protocols to linear in the statistical [12] and perfect [13] settings. Rachuri and Scholl [66] obtained protocols in the dishonest-majority setting. For guaranteed output delivery, David et al. [27] and Deligios et al. [33] obtained perfect and statistical Layered MPC with $t < n/3$ and $t < n/2$, respectively.

[8] In maximally fluid MPC a party may exit the computation after a single round of receiving and sending messages.

**YOSO MPC [43].** The YOSO MPC model considers committees that speak only once, and an idealized communication channel called the role assignment functionality, which ensures that every party in a committee can send private messages to any recipient in an arbitrary future round. In contrast to layered MPC, a dishonest sender cannot change the message while it is traveling to the recipient and is, effectively, committed to its input.[9] Recent works provided improvements to YOSO MPC across various directions. Braun et al. [18] designed distributed key generation for linearly homomorphic threshold encryption in the YOSO setting. Escudero et al. [37] showed how to obtain YOSO protocols in the offline-online model with (amortized) constant online communication, and Kolby et al. [54] introduced constant-round YOSO MPC protocols. Recently, Bhadauria et al. [9] designed FHE-based YOSO MPC which uses only one honest majority committee. Finally, a number of works focused on obtaining efficient role-assignment protocols [7, 23, 22, 19, 28].

**SCALES MPC [4].** SCALES can be seen as a variant of YOSO MPC, where the committee of input and output clients is the same, and, in contrast to YOSO, these clients are allowed to speak twice. Moreover, SCALES requires that the communication is done purely via broadcast (no secure channels available), as well as the complexity of each input client only depends on the size of their inputs and the number of participating computing parties (in particular, it does not depend on the size of the computation). While the original work on SCALES [4] focused on the semi-honest setting, Acharaya et al. [3] provided protocols secure against fully malicious adversaries. Most recently, Acharaya et al. [5] improved efficiency of SCALES for permutation branching programs.

**PASSO [61, 58].** A variant of YOSO, called *Parties Speak Sequentially Once* was considered in [61, 59, 58] for the problem of randomness generation. PASSO considers a setting where committees are executed in sequence, and the adversary can corrupt a total of $t$ committees. Each committee here typically consists of a single party, and parties have access to channels to the future.

## 2 Technical Overview

We summarize the main technical challenges and ideas behind our layered broadcast protocols and their integration into layered MPC.

### 2.1 Layered Broadcast

We first consider a layered version of consensus, where every party $P_i \in \mathcal{L}_0$ has an input $x_i$ and parties in layer $\mathcal{L}_d$ want to agree on the same bit. If all honest parties in $\mathcal{L}_0$ input the same bit $b$, every honest party in $\mathcal{L}_d$ should output $b$. Broadcast follows by having the sender send its input $x$ to all parties in $\mathcal{L}_1$, and then running consensus. See appendix C for details.

[9] The original YOSO work also considered random corruptions instead of worst-case corruptions, but this assumption is not crucial for their protocols.

**Background: Feldman-Micali Paradigm.** Our starting point is the seminal Feldman-Micali (FM) paradigm [39] for building probabilistic consensus. At a high level, protocols in this paradigm are organized as a sequence of iterations, where each iteration consists of an invocation of two primitives: *weak consensus* and a *weak common coin*.

A weak common coin is a protocol that, with constant probability $p$, outputs a uniform common random bit $c \in \{0,1\}$ to all parties. In weak consensus (WC), each party $P_i$ inputs a bit $x_i \in \{0,1\}$ and outputs a value $z_i \in \{0,1,\perp\}$, with the guarantee that there is a value $z$ such that each honest party $P_i$ outputs either $z$ or $\perp$, i.e., honest parties do not output contradicting bits. Moreover, if each honest $P_i$ inputs the same bit $b$, they all output $b$.

To achieve consensus, each honest party $P_i$ in every iteration **(1)** invokes $z_i \leftarrow \mathsf{WC}(x_i)$, and **(2)** runs the common coin protocol to obtain $c_i$, setting

$$y_i = \begin{cases} z_i, & \text{if } z_i \in \{0,1\}, \\ c_i, & \text{if } z_i = \perp. \end{cases}$$

Note that after the invocation of $\mathsf{WC}$, honest values are within the interval $\{b, \perp\}$ for some bit $b \in \{0,1\}$. This means that if the coin is successful (outputs a uniform common random bit to all parties) and equal to $b$, parties reach agreement. This happens with probability $p/2$.

By repeating the iteration steps $\mathcal{O}(\lambda)$ times for security parameter $\lambda$, where the input to a new iteration is set to be the output of the previous iteration, agreement is reached with overwhelming probability.

In order to adapt this approach to achieve layered consensus, the main technical challenge is in designing the weak common coin protocol (this is the place where prior work assumes trusted setup), although there are other subtleties that need to be solved along the way. For example, in each iteration, after running $\mathsf{WC}$, the layer holding the output of weak consensus, say $\mathcal{L}_{d_1}$, needs to securely transmit the output to the layer that will hold the output of the common coin $\mathcal{L}_{d_1+d_2}$ (which takes, say, $d_2$ rounds); see Section 7 for a mechanism to solve this.

**Layered Weak Common Coin.** We adapt the leader election protocol of Katz and Koo [53] to achieve a layered weak common coin. This protocol uses an $n$-party primitive called *moderated VSS*, which has two designated parties, the dealer and the moderator. It guarantees that either the dealer successfully shares a secret, or, if the moderator is corrupt, any failure is detected by all parties. "False negatives" are possible: a corrupt moderator may select a subset of honest parties and make them think that the protocol failed. Using $n^2$ instances of moderated VSS, one for each possible choice of dealer (sharing a random value) and moderator, the protocol then assigns a random *ticket* value to each moderator $P_m$, namely the sum of all values shared in an instance of moderated VSS where $P_m$ acted as moderator. Then, the secrets are reconstructed, and every party $P$ selects the moderator with the lowest ticket value as their leader, discarding all moderators which $P$ detected as having acted incorrectly.

This works as, although parties may disagree on which moderators to discard, it is guaranteed that every moderator considered acceptable by some honest party got assigned a uniformly random ticket value. Thus, there is a reasonably high probability of at least $\frac{n-t}{n} - \frac{1}{n^2}$ that the unique lowest value is assigned to an honest moderator, in which case all honest parties choose the same leader.

**Converting the Leader Election to a Coin.** To obtain a random coin rather than an oblivious leader election protocol, we duplicate this process, assigning two values to every moderator in the same fashion as described above. The first value (ticket value) is used for choosing the leader as before, while the second value assigned to the leader is used as the coin value. The idea is that while the adversary may influence the choice of leader (by corrupting and disqualifying selected parties), the coin value output by any honest party is uniform random.

Achieving a layered weak common coin presents several challenges.

**Challenge 1: Stateless Moderator.** First, in the layered setting, it is not possible for one party to act as a moderator for the entire sharing protocol, as prior moderated VSS protocols require the moderator to keep a state throughout multiple rounds (in prior constructions the moderator is in charge of emulating a broadcast channel throughout the protocol execution). To address this, instead of relying on a single stateless moderator, we introduce a new abstraction *layered moderated VSS* which extends the prior definition to consider *multiple* moderators. More concretely, there is a sequence of moderators $P_{m_1}, \ldots, P_{m_k}$ belonging to different layers, and the primitive guarantees that the dealer successfully shares a secret only in the case where all the moderators are honest. Our construction of layered moderated VSS generalizes the technique from [53], introducing a compiler from layered VSS (which use a broadcast channel) to layered moderated VSS (without a broadcast channel). See §5 for details.

Note that in the coin protocol, this exponentially decreases the fraction of fully honest moderator sequences $\left(\frac{n-t}{n}\right)^k$ with the number of moderators $k$. This is not an issue, since our scheme uses only constantly many moderators.

**Challenge 2: Propagating the Layered Moderated-VSS Outcome.** In the standard setting with a static set of parties, the parties keep a state throughout the protocol on whether any given moderator is considered honest or not. In our setting, this information needs to be further propagated throughout future layers. We achieve this by invoking one weak consensus protocol for each moderator sequence, and each party only takes into account moderator sequences where the weak consensus outputs 1 (and we have a method to securely propagate weak consensus outputs to future layers). This guarantees that an honest moderator sequence is considered honest throughout future layers; and any moderator sequence with a failed moderated VSS instance is always considered disqualified.

However, this introduces a new vulnerability: a rushing adversary is able to make one honest party $P_{h_1}$ view all corrupt moderators as disqualified after the Sharing phase of the moderated-VSS instances, while another honest party $P_{h_2}$ views all of them as successful. Note that in this case, all moderated-VSS instances are secure (since $P_{h_2}$ considers them successful). During the Reconstruction phase, the adversary can first see the reconstructed secret values, and

based on this, influence the outputs of the weak consensus protocols for the corrupted moderators. If a corrupted moderator has the lowest ticket value, the adversary can choose to adaptively disqualify this moderator during the weak consensus. As a consequence, the output of the coin is biased.

Nevertheless, we show that the protocol still achieves a weaker definition of a common coin, namely that every output value occurs with constant positive probability, which suffices for our overall construction.

### 2.2 Achieving Layered MPC without Broadcast

As mentioned before, our broadcasts can be plugged into existing MPC protocols with dynamic committees. However, plugging a broadcast introduces a number of intermediate layers equal to the round complexity of broadcast, which is $\Theta(\lambda)$. This is not a problem in YOSO MPC as channels to future committees are available, or if we assume a plain PKI for public-key encryption (transmitting a message towards a specific party in a future committee can be done by encrypting under the receiver's public key, and forwarding the message until it reaches the receiver). However, without public-key encryption, all known protocols for secure message transmission from point-to-point secure channels in the Fluid and Layered MPC settings have a communication that increases exponentially with the number of committees $d$ from sender to receiver, thus the resulting protocol has exponential communication. We address this by introducing a new secure message transmission protocol which has communication complexity is polynomial when $(t+1)^2 \leq n$, using homomorphic commitments.

**Background: Strawman Approach.** To illustrate why prior approaches require exponential communication, let us recap a standard approach for $t < n/3$ corruptions. Consider a sender $P_{\text{sdr}} \in \mathcal{L}_0$ with input $s$ and a receiver $P_{\text{rcv}} \in \mathcal{L}_d$. In the case where $d = 1$ (sender and receiver are in subsequent layers), they can directly communicate via the provided secure channel. If $d = 2$, a simple solution consists of sender distributing a $t$-out-of-$n$ Shamir secret sharing of $s$ to layer $\mathcal{L}_1$, so that each party in this layer holds a share. Each $P_i \in \mathcal{L}_1$ then forwards their share to $P_{\text{rcv}} \in \mathcal{L}_2$, who robustly reconstructs the secret. The general case ($d > 2$) can be done recursively. Assume an SMT protocol $\Pi$ for distance $d' = d - 1$. The sender can again distribute a Shamir sharing of the secret towards layer $\mathcal{L}_1$. Each of the shares is then forwarded by parties in layer $\mathcal{L}_1$ to the receiver in $\mathcal{L}_d$ using a separate instance of $\Pi$. Note, however, that the communication complexity of this protocol is $\mathcal{O}(n^{d-1})$, since number of values received and communicated by each party increases exponentially with the number of intermediate layers $d$. To the best of our knowledge, all layered SMT protocols suffer from a problem of this kind and incur exponential communication.

**Challenge 3: Difficulty in Reducing the State Size.** One may wonder whether it is possible to reduce the state size in the previous approach. Indeed, after the sender $s$ distributes Shamir shares to $\mathcal{L}_1$, one could attempt to invoke a proactive secret sharing scheme so that parties in $\mathcal{L}_2$ hold a re-randomized Shamir sharing of the same secret. For example, by letting every $P_i \in \mathcal{L}_1$ reshare their share $s_i$ into sub-shares $(s_{i,1}, \ldots, s_{i,n})$ and sending $s_{i,j}$ to $P_j \in \mathcal{L}_2$, who

can re-combine the shares $(s_{1,j}, \ldots, s_{n,j})$ into a single share $s'_j$ using Lagrange interpolation and linearity of Shamir sharing, so that the new shares $(s'_1, \ldots, s'_n)$ constitute a valid degree-$t$ Shamir sharing of the original secret $s$. Unfortunately, for this (well-known [34, 42]) approach to work, it is crucial that all honest parties in $\mathcal{L}_2$ agree on a consistent set $S \subset \mathcal{L}_1$ of parties that correctly shared their share. At a high level, agreeing on such a set requires broadcast. Note that using our broadcast protocol would again introduce $\Theta(\lambda)$ additional intermediate layers between $\mathcal{L}_1$ and $\mathcal{L}_2$, and the protocol also requires securely sending private subshares from $\mathcal{L}_1$ to $\mathcal{L}_2$. Therefore, this becomes a circular problem.

**Layered Secure Message Transmission.** To securely transmit messages across intermediate layers without using PKI or broadcast channels, we introduce a novel proactive secret sharing scheme based on non-interactive linearly-homomorphic commitments. At a high level, the sender will distribute a commitment $C$ of the secret input to all parties (this commitment can be propagated for $t < n/2$ across all future layers by letting parties forward it to all parties and by taking a majority decision). Parties in each layer will hold multiple Shamir-sharings of the secret with consistent commitments to shares as described below.
*Valid Sharing.* Given commitment $C$, a *valid share* of $P_i$ is a tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$, with $\mathbf{C}_i^\ell = (C_1^\ell, \ldots, C_n^\ell)$, such that 1) $\mathsf{Com}(s_i^\ell, r_i^\ell) = C_i^\ell$, and 2) the points $(0, C)$ and $\{(k, C_k^\ell)\}_{k\in[n]}$ lie on a degree-$t$ polynomial.

We call value $\ell$ the label of this share. It acts as the **unique** identifier of the share $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ among the shares of party $P_i$, i.e. each $P_i$ has at most one tuple with each label $\ell$. Looking ahead, the label will encode the set of parties of prior layer whose shares were used in the construction of this one.

We then say that parties in layer $\mathcal{L}$ jointly hold *a valid sharing* for label $\ell$ iff each honest party holds a valid share with label $\ell$, and the vector of commitments contained in each such share is the same for all honest parties.[10]

Our protocol ensures that, if the dealer is honest, in each layer $\mathcal{L}$ parties jointly hold a valid sharing for at least one label. As mentioned above, honest parties are always in agreement on the commitment of the secret $C$. However, parties will not know *which* label corresponds to a joint valid sharing. The fact that they never have to explicitly agree on a right label is what allows us to circumvent the circular problem mentioned above.

Reshare *Protocol.* We describe our resharing protocol (Reshare, in Figure 13):

**Each party $P_i$ in the old layer $\mathcal{L}_{\mathsf{old}}$** has as input (1) the commitment to the secret $C$, and, (2) a set of valid shares $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ for $\ell \in \mathcal{Q}$, where $\mathcal{Q} := \binom{[n]}{t+1}$ It forwards the commitment to the secret $C$ to each party of the next layer. Additionally, it reshares each of the valid shares $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ as follows. First, it computes Shamir shares of $s_i^\ell$, $r_i^\ell$, denoted by $(s_{i,1}^\ell, \ldots, s_{i,n}^\ell)$ and $(r_{i,1}^\ell, \ldots, r_{i,n}^\ell)$ respectively. Then, it computes the vector of commitments to these subshares $\widetilde{\mathbf{C}}_i^\ell[j] = \mathsf{Com}(s_{i,j}^\ell, r_{i,j}^\ell)$. Finally, for each $j \in [n]$ it sends $(\ell, \widetilde{\mathbf{C}}_i^\ell, s_{i,j}^\ell, r_{i,j}^\ell)$ to party $P_j$ of the next layer $\mathcal{L}_{\mathsf{new}}$.

[10] Note that there may be valid shares that do not correspond to any joint valid sharing.

**Each party $P_j$ of the new layer $\mathcal{L}_{\mathsf{new}}$** sets the commitment to the secret to be the commitment value sent by the majority of parties in the old layer. Denote this commitment (again) as $C$.

Then, **for each $(t+1)$-sized subset $I \in \mathcal{Q}$** of parties of the prior layer, $P_j$ attempts to combine subshares (and their commitments) for the same label into a single valid share. More formally:

$P_j$ finds the **maximum** label $\ell^*$ in common between all parties of $I$ such that for the tuples $(\ell^*, \widetilde{\mathbf{C}}_i^*, s_{i,j}^*, r_{i,j}^*)$, where $i \in I$, the following check **(†)** holds:

- Check that $\widetilde{\mathbf{C}}_i^*[i] = \mathsf{Com}(s_{i,j}^*, r_{i,j}^*)$ and that $\widetilde{\mathbf{C}}_i^*$ lies on a degree-$t$ polynomial. Let $C_i^*$ denote the value of this polynomial at 0.
- Check that $(0, C)$ and $\{(i, C_i^*)\}_{i \in I}$ lie on a degree-$t$ polynomial.

If there exist no such labels, $P_j$ continues to the next index subset. Otherwise, it combines the sub-shares into one share using Lagrange interpolation. That is:

- Let $f_j^I$ be a degree-$t$ polynomial s.t. $f_j^I(i) = s_{i,j}^*$ for $i \in I$. Let $s_j^{\mathsf{new},I} = f_j^I(0)$.
- Let $g_j^I$ be a degree-$t$ polynomial s.t. $g_j^I(i) = r_{i,j}^*$ for $i \in I$. Let $r_j^{\mathsf{new},I} = g_j^I(0)$.
- Let $h_j^I$ be a degree-$t$ polynomial s.t. $h_j^I(i) = \widetilde{\mathbf{C}}_i^*$ for $i \in I$. Let $\mathbf{C}_j^{\mathsf{new},I} = h_j^I(0)$.

Finally, $P_j$ stores $(I, \mathbf{C}_j^{\mathsf{new},I}, s_j^{\mathsf{new},I}, r_j^{\mathsf{new},I})$.

*Analysis Idea.* The idea is that if there is a label $\ell^*$ corresponding to a valid sharing initially stored in the old layer $\mathcal{L}_{\mathsf{old}}$, there will again be a label $I^*$ with a valid sharing stored in the new layer $\mathcal{L}_{\mathsf{new}}$. Let $H$ be a set of $t+1$ (honest) indices, and $\mathcal{L}_H \subset \mathcal{L}_{\mathsf{old}}$ the corresponding set consisting of $t+1$ honest parties. Since $t < n/2$, such a set exists. By assumption, parties in $\mathcal{L}_H$ hold at least one valid sharing corresponding to some label $\ell$. For such $\ell$, and an honest sender, the requirement **(†)** will be satisfied: We have that every $P_i \in \mathcal{L}_H$ holds a commitment $C$ to the secret, a common list of commitments $\mathbf{C}^\ell = (C_1^\ell, \ldots, C_n^\ell)$ consistent with $C$, and an opening $(s_i^\ell, r_i^\ell)$. of the $i$-th commitment. Each $P_i \in \mathcal{L}_H$ computes sub-shares $(s_{i,1}^\ell, \ldots, s_{i,n}^\ell)$ and $(r_{i,1}^\ell, \ldots, r_{i,n}^\ell)$, as well as commitments $\widetilde{\mathbf{C}}_i^\ell[j] = \mathsf{Com}(s_{i,j}^\ell, r_{i,j}^\ell)$ and sends to each $P_j \in \mathcal{L}_{\mathsf{new}}$ the tuple $(\ell, \widetilde{\mathbf{C}}_i^\ell, s_{i,j}^\ell, r_{i,j}^\ell)$.

Note that any honest party $P_j \in \mathcal{L}_{\mathsf{new}}$ considers at some point the (index) subset $I = H$ (since $P_j$ iterates over **all** index subsets of size $(t+1)$). When doing so, all $P_j \in \mathcal{L}_{\mathsf{new}}$, obtain the same, nonempty set of valid labels (i.e. labels where **(†)** is satisfied). Then, every honest $P_j \in \mathcal{L}_{\mathsf{new}}$ picks the same maximum label $\ell^*$, combines the received sub-shares and computes a valid share tuple while labelling it $H$. As they all picked the same $\ell^*$, layer $\mathcal{L}_{\mathsf{new}}$ holds a valid sharing. By pruning all tuples received from $I$ but one, we keep the state constant when repeating resharing, achieving a linear communication in the number of layers.

*Improving Communication Complexity.* This resharing protocol enables us to construct a secure message transmission protocol without using a PKI, secure for any $t < n/2$, where the communication complexity per layer is $\Omega(\binom{n}{t+1})$. To obtain a more efficient SMT protocol with polynomial complexity in both the number of layers and the number of parties per layer, we consider the setting

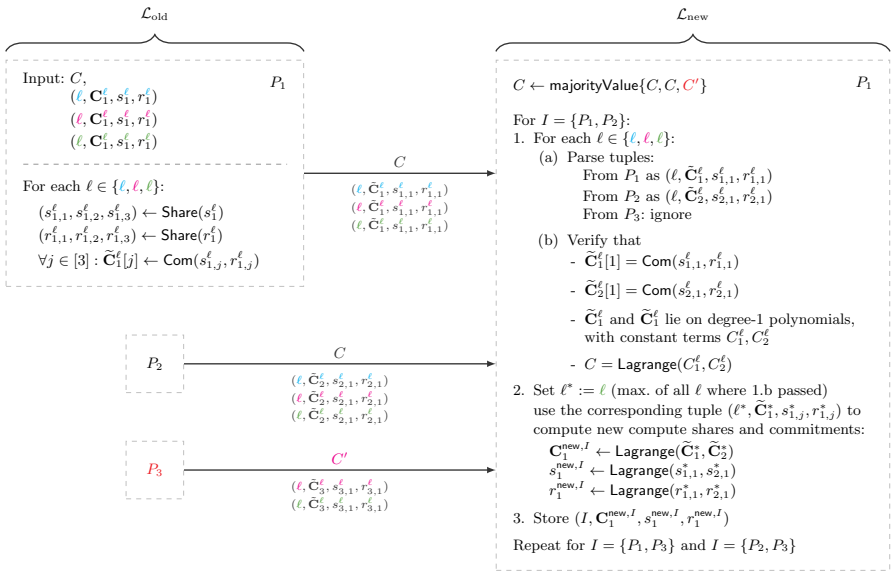



**Fig. 1.** Reshare protocol with $n = 3$ and $t = 1$, where $P_1 \in \mathcal{L}_{\mathsf{new}}$ is processing set $I = \{1, 2\}$, and $P_3 \in \mathcal{L}_{\mathsf{old}}$ is corrupted. The function Lagrange interpolates the value at position 0 of a degree-1 polynomial on the pair of input values at positions 1 and 2.

$(t+1)^2 \leq n$. The idea is that, for every committee we simply choose $t+1$ fixed, pairwise disjoint subsets of size $t+1$ each. If only $t$ parties are corrupt, one of these sets is guaranteed to be fully honest. Thus, it suffices to iterate $I$ over the $t+1$ sets, and keep $t+1$ labels rather than $\binom{n}{t+1}$. See Section 9 for details.

## 3 Preliminaries

### 3.1 Model

Our protocols can be phrased in any existing framework for protocols with dynamic committees [43, 27, 29] that assume secure point-to-point channels. For simplicity, we describe our protocols following the Layered MPC formalization in [29], with a slightly different network model. Namely, parties do not have access to broadcast channels (we construct them from the point-to-point channels). For the ease of presentation, throughout the sections constructing the broadcast protocol, we will assume that parties can send private messages to any party in a future layer. This can be realized from channels between consecutive layers via a secure message transmission protocol (see also functionality $\mathcal{F}_{\text{smt}}$).

A layered MPC protocol can be viewed as a special case of standard MPC with a general adversary structure, specialized in the following way: 1) the interaction pattern is defined by a layered graph, and 2) the adversary can corrupt at most $t$ parties in each layer.

**Definition 1 (Layered Protocol).** *Let $n, t, d$ be positive integers. An $(n, t, d)$-*layered protocol *is a synchronous protocol $\Pi$ over secure point-to-point channels with the following special features.*

- **Parties.** *There are $N = n(d+1)$ parties partitioned into $d+1$ layers $\mathcal{L}_i$, $0 \leq i \leq d$, where $|\mathcal{L}_i| = n$. Parties in the first layer $\mathcal{L}_0$ and the last layer $\mathcal{L}_d$ are referred to as* input clients *and* output clients*, respectively.*
- **Interaction pattern.** *The interaction consists of $d$ rounds, where in round $i$ parties in $\mathcal{L}_{i-1}$ may send messages to parties in $\mathcal{L}_i$ over secure point-to-point channels.*
- **Functionalities.** *We consider functionalities $f$ that take inputs from input clients and deliver outputs to output clients.*
- **Adversaries.** *We consider a static adversary which may corrupt up to $t$ parties in every layer $\mathcal{L}_i$ for $0 \leq i \leq d$.*

*We say that a protocol $\Pi$ is a layered protocol for $\mathcal{F}$ if it realizes $\mathcal{F}$ in the sense of MPC with general adversary structures* [21, 20, 51] *for the adversary structure described above.*

**Comparison with other models** Our model is equivalent to Layered MPC and maximally Fluid MPC with static corruptions, except that we do not assume a broadcast channel. Some works also consider the setting where different committees may be of different sizes. For ease of presentation, we avoid discussing this but our results hold in this setting as well.

In the abstract YOSO model [43, 22], protocols are defined in terms of "roles" which can send messages to any future role. Abstract YOSO protocols can then be compiled to real-world protocols, where roles are assigned to physical machines [22]. Our protocols straightforwardly work in this model, as the only difference is that abstract YOSO assumes channels to all future committees.

### 3.2 Cryptographic Primitives

*Signature Scheme.* Some of our protocols will make use of signatures. We briefly recall the syntax of a standard signature scheme.

**Definition 2.** *A signature scheme $\Sigma$ is a triple of efficient algorithms $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Vrf})$, with the following syntax:*

1. ***Key Generation:*** $\mathsf{Keygen}$ *takes security parameter $\lambda$ and outputs a signature-verification key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk}, \mathsf{pk} \in \{0,1\}^\lambda$.*
2. ***Signature Generation:*** $\mathsf{Sign}$ *takes the message $m$ and the signing key $\mathsf{sk}$ as inputs and outputs a signature $\sigma$.*
3. ***Verification:*** $\mathsf{Vrf}$ *takes the message $m$, a signature $\sigma$ and the verification key $\mathsf{pk}$ as inputs and output a decision-bit $b \in \{0,1\}$.*

*We require signatures to satisfy correctness, i.e., $\mathsf{Vrf}(m, \mathsf{Sign}(m, \mathsf{sk}), \mathsf{pk}) = 1$ for all $m \in \{0,1\}^*$ and for $(\mathsf{sk}, \mathsf{pk}) = \mathsf{Keygen}(1^\lambda)$. We also require our signatures to satisfy the standard unforgeability against chosen message attacks [47].*

In our computational protocols we use any signature scheme with keys of size $\mathcal{O}(\lambda)$ and signatures of size $\mathcal{O}(\lambda)$ per message of $\mathcal{O}(\lambda)$ bits [36, 67, 52, 16].

Our protocols using signatures will assume a bulletin-board PKI (also known as plain PKI). This means that every $P_i \in \mathcal{L}_k$ holds a secret key $\mathsf{sk}_i^k$ and all parties from all layers hold the respective public key $\mathsf{pk}_i^k$. Moreover, the keys from corrupted parties can be chosen adversarially.

*Commitment Scheme.* We will make use of commitments in our secure message transmission protocols. We briefly recap the syntax here.

**Definition 3.** *A non-interactive commitment scheme with message space $\mathcal{M}$, randomness space $\mathcal{R}$, and commitment space $\mathcal{C}$ consists of a triple of PPT algorithms with the following syntax:* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $C \leftarrow \mathsf{Com}(\mathsf{pp}, m, r)$, *and* $\mathsf{b} \leftarrow \mathsf{Open}(\mathsf{pp}, C, m, r)$. *We require the commitment to be computationally hiding, computationally binding and linearly homomorphic. See Definition 13 for a full definition.*

Commitments with these properties can be instantiated using a variety of assumptions such as discrete-log (like Pedersen commitments [63]).

### 3.3 Building Blocks

*Secure Message Transmission.* We first formalize the functionality $\mathcal{F}_{\text{smt}}$ that allows parties to communicate securely towards future layers.

**Definition 4 (Secure Message Transmission).** *An $(n, t, d)$-layered protocol $\Pi$ is a $t$-secure message transmission protocol if it realizes $\mathcal{F}_{\text{smt}}$.*

**Secure Channel $\mathcal{F}_{\text{smt}}$**

**Public Parameters:** Sender $P_{\text{sdr}} \in \mathcal{L}_0$, receiver $P_{\text{rcv}} \in \mathcal{L}_{d \geqslant 1}$. The domain $M$ of the message.
**Private Variables:** $m = \perp$.

- Upon receiving a message $x \in M$ from $P_{\text{sdr}} \in \mathcal{L}_0$, set $m = x$ and notify the adversary $\mathcal{A}$ that $P_{\text{sdr}}$ sent a message. If $P_{\text{rcv}}$ is corrupted, leak $m$.
- If $P_{\text{sdr}}$ is corrupt, upon receiving $m'$ from $\mathcal{A}$ before the end of round $d$, set $m = m'$ ($\mathcal{A}$ can input $m'$ in a rushing fashion, after the honest parties in $\mathcal{L}_d$ sent their messages)
- Output $m$ to $P_{\text{rcv}}$ at the end of round $d$.

*Consensus and Variants.* Throughout the paper we will use different variants of consensus protocols, including weak consensus, graded consensus, consensus, and their single-sender variant like graded broadcast and broadcast. As usual in this area, we defined most of these primitives using properties. However, for the case of broadcast we use a functionality $\mathcal{F}_{\mathsf{bc}}$ (so that it matches exactly the functionality assumed in the prior MPC papers with dynamic committees). See appendix C for a description and realization of it.

**Definition 5 (Layered Weak Consensus).** *Let $\Pi$ be a protocol executed over $(d+1)$ layers, where each party $P_j \in \mathcal{L}_0$ holds input $x_j \in \{0,1\}$. Parties in layer $\mathcal{L}_d$ terminate upon generating an output $y_j \in \{0,1,\perp\}$. We say protocol $\Pi$ is an $(n,t,d)$-layered weak consensus, if, whenever up to $t$ out of $n$ parties are corrupt in each layer, it satisfies the following properties:*

- ***Validity:*** *If there is a value $x$ such that each honest input holder $P_i$ in $\mathcal{L}_0$ has input $x_i = x$, then each honest party $P_j$ in $\mathcal{L}_d$ outputs $y_j = x$.*
- ***Consistency:*** *There exists a value $y$, such that every honest party in $\mathcal{L}_d$ outputs either $y$ or $\perp$.*

**Definition 6 (Layered Graded Consensus).** *Let $\Pi$ be a protocol executed over $(d+1)$ layers, where each party $P_j \in \mathcal{L}_0$ holds input $x_j \in \{0,1\}$. Parties in layer $\mathcal{L}_d$ terminate upon generating an output $(y_j, g_j) \in (\{0,1\} \times \{1,2\}) \cup \{(\perp, 0)\}$. We say protocol $\Pi$ is an $(n,t,d)$-layered graded consensus, if, whenever up to $t$ out of $n$ parties are corrupt in each layer, it satisfies the following:*

- ***Validity****: If there is a value $x$ such that each honest input holder in $\mathcal{L}_0$ has input $x_j = x$, then each honest party in $\mathcal{L}_d$ outputs $(x, 2)$.*
- ***Graded Consistency****: If some honest $P_i \in \mathcal{L}_d$ outputs $(y_i, 2)$, then for every honest $P_j \in \mathcal{L}_d$ we have $y_j = y_i$ and $g_j \geq 1$. Moreover, if some honest $P_i \in \mathcal{L}_d$ outputs $(y_i, 1)$, no honest party $P_j \in \mathcal{L}_d$ outputs $(y_j, 1)$ with $y_j \neq y_i$.*

**Definition 7 (Layered Graded Broadcast).** *Let $\Pi$ be a protocol executed over $(d+1)$ layers, where a designated party $P_s \in \mathcal{L}_0$ holds input $x \in \{0,1\}$. Parties in layer $\mathcal{L}_d$ terminate upon generating an output $(y_j, g_j) \in (\{0,1\} \times \{1,2\}) \cup \{(\perp, 0)\}$. We say protocol $\Pi$ is an $(n,t,d)$-layered graded broadcast if, whenever up to $t$ out of $n$ parties are corrupt in each layer, it satisfies the following:*

- ***Validity****: If $P_s \in \mathcal{L}_0$ is honest, then every honest $P_i$ outputs $(x, 2)$.*
- ***Graded Consistency****: If some honest $P_i \in \mathcal{L}_d$ outputs $(y_i, 2)$, then for every honest $P_j \in \mathcal{L}_d$ we have $y_j = y_i$ and $g_j \geq 1$. Moreover, if some honest $P_i \in \mathcal{L}_d$ outputs $(y_i, 1)$, no honest party $P_j \in \mathcal{L}_d$ outputs $(y_j, 1)$ with $y_j \neq y_i$.*

**Definition 8 (Layered Consensus).** *Let $\Pi$ be a protocol executed over $d+1$ layers, where each party $P_j \in \mathcal{L}_0$ holds input $x_j \in \{0,1\}$. Parties in layer $\mathcal{L}_d$ terminate upon generating an output $y_j \in \{0,1\}$. We say protocol $\Pi$ is an $(n,t,d)$-layered consensus protocol if whenever up to $t$ out of $n$ parties are corrupt in each layer, it satisfies the following properties:*

- ***Validity:*** *If there is a value $x$ such that each honest input holder $P_i \in \mathcal{L}_0$ has input $x_i = x$, then each honest party $P_j \in \mathcal{L}_d$ outputs $y_j = x$.*
- ***Consistency:*** *There exists a value $y$, such that every honest party in $\mathcal{L}_d$ outputs $y$.*

*Verifiable Secret Sharing.* We define layered VSS (and the moderated version).

**Definition 9 (Verifiable Secret Sharing).** *An $(n,t,d,d')$-layered verifiable secret sharing (VSS) protocol consists of two subprotocols:*

– *A protocol* Share *runs over $d$ layers $\mathcal{L}_0, \ldots, \mathcal{L}_d$, where a designated dealer $P_s \in \mathcal{L}_0$ holds an input $s$, and every $P_i$ in $\mathcal{L}_d$ outputs a value* $\mathsf{sh}_i$.
– *A protocol* Reconstruct *executed over $d'$ layers $\mathcal{L}_d, \ldots, \mathcal{L}_{d+d'}$. At the end of the protocol, every honest party $P_i \in \mathcal{L}_{d+d'}$ outputs a value $s'$.*

*Such that there is a unique value $s'$ defined by the joint views of all honest parties in $\mathcal{L}_d$ such that (1) if $P_s$ is honest, $s' = s$ and the joint view of all corrupt parties in layers $\mathcal{L}_0, \ldots, \mathcal{L}_d$ is independent of $s$ and (2) after executing the protocol* Reconstruct*, every honest party in $\mathcal{L}_{d+d'}$ outputs $s'$.*

We generalize *moderated VSS (mVSS)* [53] to the layered setting:

**Definition 10 (Layered Moderated VSS).** *An $(n, t, d, d')$-layered moderated VSS (lmVSS) protocol consists of two subprotocols:*

– *A protocol* Share *executed over $d$ layers $\mathcal{L}_0, \ldots, \mathcal{L}_d$, with a designated dealer $P_s \in \mathcal{L}_0$ holding input $s$, as well as a sequence of designated parties $P_{m_1}, \ldots, P_{m_k}$, called the moderators, with $P_{m_i} \in \mathcal{L}_{l_i}$ for some $l_1 < \cdots < l_k < d$, where every $P_i$ in $\mathcal{L}_d$ outputs a value $\mathsf{sh}_i$ and a flag $\mathsf{success}_i \in \{0, 1\}$.*
– *A protocol* Reconstruct *executed over $d'$ layers $\mathcal{L}_d, \ldots, \mathcal{L}_{d+d'}$. At the end of the protocol, every honest party $P_i \in \mathcal{L}_{d+d'}$ outputs a value $s'$.*

*Such that the following is satisfied:*

– *If all moderators are honest, every honest $P_i$ outputs $\mathsf{success}_i = 1$ after the protocol* Share*.*
– *If some honest $P_i$ outputs $\mathsf{success}_i = 1$ after running* Share*, then there is a unique value $s'$ defined by the joint views of all honest parties in $\mathcal{L}_d$ such that (1) if $P_s$ is honest, $s' = s$ and the joint view of all corrupt parties in layers $\mathcal{L}_0, \ldots, \mathcal{L}_d$ is independent of $s$ and (2) after executing the protocol* Reconstruct*, every honest party in $\mathcal{L}_{d+d'}$ outputs $s'$.*

## 4 Layered Weak and Graded Consensus

Proofs for this section can be found in appendix A.

### 4.1 Perfect Security for $t < n/3$

For completeness, we describe protocols from [8, 39] that achieve layered weak consensus and layered graded consensus and prove them secure, see Figures 2 and 3. We prove their security in appendix A.

**Theorem 5.** *Protocol* WC *defined in Figure 2 is an $(n, t, 1)$-layered weak consensus protocol for any $t < \frac{n}{3}$.*

**Theorem 6.** *Protocol* GC *defined in Figure 3 is an $(n, t, 2)$-layered graded consensus protocol for any $t < \frac{n}{3}$.*

**Weak Consensus Protocol** WC

**Inputs:** Every $P_i \in \mathcal{L}_0$ holds $x_i \in \{0,1\}$.
**Outputs:** Every $P_i \in \mathcal{L}_1$ outputs $y_i \in \{\perp, 0, 1\}$.

$\mathcal{L}_0$**:** Every $P_i$ sends $x_i$ to every $P_j \in \mathcal{L}_1$.
$\mathcal{L}_1$**:** If $P_i$ received one value $x$ from at least $n-t$ parties, $P_i$ outputs $x$. Otherwise, $P_i$ outputs $\perp$.

**Fig. 2.** Weak Consensus for $t < n/3$

**Graded Consensus Protocol** GC

**Inputs:** Every $P_i \in \mathcal{L}_0$ holds $x_i$.
**Outputs:** Every $P_i \in \mathcal{L}_2$ outputs $(y_i, g_i)$.

$\mathcal{L}_0$**:** Every $P_i$ sends $x_i$ to every $P_j \in \mathcal{L}_1$.
$\mathcal{L}_1$**:** If $P_i$ received a message $x$ from at least $n-t$ parties, $P_i$ sends $x$ to all $P_j \in \mathcal{L}_2$.
$\mathcal{L}_2$**:** If $P_i$ received a message $x$ from at least $n-t$ parties, $P_i$ outputs $(x, 2)$. Otherwise, if $P_i$ received a message $x'$ from at least $t+1$ parties, $P_i$ outputs $(x', 1)$. Otherwise, $P_i$ outputs $(\perp, 0)$.

**Fig. 3.** Graded Consensus for $t < n/3$

### 4.2 Computational Security for $t < n/2$

We now modify weak and graded consensus schemes [60, 40, 53] to get security in the layered setting assuming a plain PKI (see Appendix A for security proofs).

In the following, we let a **degree-0-certificate for** $b \in \{0,1\}$ be a list of valid signatures on $b$ by at least $n-t$ parties in $\mathcal{L}_0$. We define a **degree-$k$-certificate for** $b \in \{0,1\}$ to be a list $(c_i, \sigma_i)_{i \in \mathcal{I}}$ such that each $c_i$ is a degree-$(k-1)$-certificate for $b$, each $\sigma_i$ is a valid signature on $c_i$ by $P_i \in \mathcal{L}_k$ and $|\mathcal{I}| \geq n-t$.

**Protocol** aWC

**Inputs:** Every $P_i \in \mathcal{L}_0$ holds $x_i \in \{0,1\}$
**Outputs:** Every $P_i \in \mathcal{L}_2$ outputs $y_i \in \{\perp, 0, 1\}$

$\mathcal{L}_0$**:** Every $P_i$ sends $(x_i, \mathsf{Sign}_{\mathsf{sk}_i^0}(x_i))$ to all $P_j \in \mathcal{L}_1$.
$\mathcal{L}_1$**:** If $P_i$ is able to form a degree-0-certificate $c_i$ for $b \in \{0,1\}$ from the received messages[a], $P_i$ sends $c_i$ to every $P_j \in \mathcal{L}_2$
$\mathcal{L}_2$**:** If $P_i$ received at least $n-t$ degree-0-certificates for $b$ and no degree-0-certificate for $(1-b)$, $P_i$ outputs $b$. Otherwise, $P_i$ outputs $\perp$.

[a] i.e. $P_i$ received $\geq n-t$ messages with a valid signature on $b$

**Fig. 4.** Authenticated Weak Consensus for $t < n/2$

**Theorem 7.** *Assuming a plain PKI, protocol* aWC *defined in Figure 4 is an* $(n, t, 2)$*-layered weak consensus protocol for any* $t < \frac{n}{2}$.

We now introduce an authenticated layered graded consensus protocol based on the graded broadcast protocol in [60], but achieving a slightly stronger consistency level at the expense of an extra round.

**Authenticated Graded Consensus Protocol** aGC

**Inputs:** Every $P_i \in \mathcal{L}_0$ holds $x_i$.
**Outputs:** Every $P_i \in \mathcal{L}_3$ outputs $(y_i, g_i)$.

$\mathcal{L}_0$**:** Every $P_i$ sends $(x_i, \mathsf{Sign}_{\mathsf{sk}_i^0}(x_i))$ to all $P_j \in \mathcal{L}_1$.
$\mathcal{L}_1$**:** If $P_i$ is able to form a degree-0-certificate $c_i$ for $b \in \{0, 1\}$ from the received messages, $P_i$ sends $(c_i, \mathsf{Sign}_{\mathsf{sk}_i^1}(c_i))$ to every $P_j \in \mathcal{L}_2$
$\mathcal{L}_2$**:** If $P_i$ received at least $n - t$ degree-0-certificates for $b$ (with valid signatures) and no degree-0-certificate for $(1 - b)$, $P_i$ forms a degree-1-certificate $c_i$ and sends $(c_i, \mathsf{Sign}_{\mathsf{sk}_i^2}(c_i))$ to every $P_j \in \mathcal{L}_3$
$\mathcal{L}_3$**:** If $P_i$ received at least $n - t$ degree-1-certificates for $b$ (with valid signatures) and no degree-1-certificate for $(1 - b)$, $P_i$ forms a degree-2-certificate $c_i$ and sends $(c_i, \mathsf{Sign}_{\mathsf{sk}_i^3}(c_i))$ to every $P_j \in \mathcal{L}_4$
$\mathcal{L}_4$**:** If $P_i$ received at least $n - t$ degree-2-certificates for $b$ and none for $(1 - b)$, $P_i$ outputs $(b, 2)$.
Otherwise, if $P_i$ received at least one degree-2-certificates for $b$ and none for $(1 - b)$, $P_i$ outputs $(b, 1)$.
Otherwise, $P_i$ outputs $(\perp, 0)$.

**Fig. 5.** Authenticated Graded Consensus Protocol for $t < n/2$

**Theorem 8.** *Assuming a plain PKI, protocol* aGC *defined in Figure 5 is an* $(n, t, 4)$*-layered graded consensus protocol for any* $t < \frac{n}{2}$.

### 4.3 Graded Broadcast Towards Multiple Layers

First, note that graded broadcast can be achieved from graded consensus simply by letting the sender $P_{\text{sdr}} \in \mathcal{L}_0$ send its input to all parties in the next layer $\mathcal{L}_1$, who will then jointly execute a graded consensus protocol. Therefore, using theorem 6 and 8, we obtain the following corollary:

**Corollary 5.** *There is an* $(n, t, 3)$*-layered graded broadcast protocol for any* $t < n/3$. *Moreover, assuming a plain PKI for signatures, there is an* $(n, t, 5)$*-layered graded broadcast protocol for any* $t < n/2$.

In Section 5, we assume graded broadcast (a.k.a. gradecast) channels which deliver output towards two committees with graded consistency holding across these committees. We briefly discuss a technique to achieve this. A similar

method is used to propagate the output of a weak consensus protocol from one layer to subsequent ones, see Section 7.

**Proposition 1.** *Assume $t < n/2$. Let $\pi$ be an $(n, t, d)$-layered gradecast protocol and $d' > d$. Then there exists an $(n, t, d')$-layered protocol $\pi'$ which achieves gradecast towards $\mathcal{L}_d \cup \mathcal{L}_{d'}$, with graded consistency holding for any pair of honest parties $P, P' \in \mathcal{L}_d \cup \mathcal{L}_{d'}$.*

## 5 Moderated Layered VSS

We present a transformation from any layered VSS protocol $\Pi$ relying on (layered) broadcast to a moderated layered VSS protocol $\Pi'$ that uses only layered graded broadcast channels $\mathsf{GradeCast}_\ell$ and $\mathsf{GradeCast}_{\ell,2\ell}$, generalizing the scheme from [53]. Here, $\mathsf{GradeCast}_\ell$ delivers outputs at layer $\ell$, and $\mathsf{GradeCast}_{\ell,2\ell}$ delivers outputs at layers $\ell$ and $2\ell$ (assuming that gradecasts start at layer 0) (see §4.3).
**Protocol Overview.** Informally, the transformation of the sharing phase $\mathsf{Share}$ of protocol $\Pi$ works as follows. For every committee $\mathcal{L}_k$:

- If no broadcast channel is used in $\Pi$ from $\mathcal{L}_k$ to $\mathcal{L}_{k+1}$, then parties in $\mathcal{L}_k$ proceed as in $\mathsf{Share}$.
- If some parties in $\mathcal{L}_k$ of $\Pi$ broadcast a message, we "expand" the protocol by $2\ell$ layers, and use a moderator in an intermediate committee $\mathcal{L}_\ell$ to simulate these broadcasts (see Figure 6). This process is not guaranteed to work, but if it fails, a warning bit $y = 1$ will be sent to parties in the final committee of $\Pi$'s sharing phase. Furthermore, the private messages that parties in $\Pi$ would normally send from $\mathcal{L}_k$ to $\mathcal{L}_{k+1}$ are instead sent to the corresponding recipients $2\ell$ layers down the line.
- Parties in the final committee output $\mathsf{success} = 0$ iff they received a warning (i.e. $y = 1$) from $t + 1$ parties (or more) from within any one committee.

Intuitively, this transformation works because the moderator can neither alter an honest senders message nor try to distribute an inconsistent message without all honest parties detecting this based on the gradecast outputs.
More formally, let $(\{m_{i,j}\}_{j\in[n]}, m_{\mathsf{bc}}) = \mathsf{NextMessage}_\Pi(i, k, r_i, S_i)$ denote the next message function of $\Pi$ which specifies the behaviour of party $P_i$ in layer $\mathcal{L}_k$, $\mathsf{fin} > k \geq 0$. Above, $i$ denotes the index of the party, $k$ denotes the layer, $r_i$ the randomness, $S_i$ the set of received messages, $m_{i,j}$ the message that $P_i$ sends to $P_j$ from the next layer, $m_{\mathsf{bc}}$ the message that $P_i$ broadcasts.
We further let $s_i = \mathsf{OutputShare}_\Pi(i, r_i, S_i)$ denote the function which specifies the behaviour of party $P_i$ in the final sharing layer of $\Pi$. Above, $s_i$ denotes the VSS share, and $\mathsf{success}_i$ the VSS success flag.

We describe the transformation in Figure 7. For ease of presentation, we assume that every layer of $\Pi$ is expanded into $2\ell$ layers (in practice, this is only necessary for layers with broadcasts). Our compiler achieves the following result:

**Theorem 9.** *Assume there exists a layered VSS protocol $\Pi$, tolerating up to $t < n/2$ corruptions, which may additionally use a broadcast channel in its*

**Simulated BC Channel of** $\mathsf{Share}$

**Input:** Let $m_{i,\mathsf{bc}}$ denote messages that $P_i$ wishes to broadcast. Let $m_{i,j}$ further denote a message (if any) that $P_i$ wishes to send to $P_j$. Let $M$ denote the moderator index. Let $\mathsf{fin}$ denote the index of the final committee.

$\mathcal{L}_0$**:** $P_i$ sends $m_{i,j}$ to $P_j$ in $\mathcal{L}_{2\ell}$ and enters $m_{i,\mathsf{bc}}$ into a gradecast protocol $\mathsf{GradeCast}_{\ell,2\ell}$.

$\mathcal{L}_\ell$**:** Moderator $P_M$: Upon receiving a message $m$ from $\mathsf{GradeCast}_{\ell,2\ell}$, input $m$ into $\mathsf{GradeCast}_\ell$.

$\mathcal{L}_{2\ell}$**:** For any gradecast with output $(m, g)$ that originated from $\mathcal{L}_0$, let $(m', g')$ denote the output of the corresponding gradecast initiated by the moderator in $\mathcal{L}_\ell$. Send a bit $y = 1$ to every party in $\mathcal{L}_{\mathsf{fin}}$, if, for any of the gradecasts, at least one of $g' \neq 2$ or ($m \neq m'$ and $g = 2$) holds. Output all $m'$.

**Fig. 6.** Subprotocol for BC Channels

*sharing phase. Then there exists a moderated layered VSS protocol* $\Pi'$*, which also tolerates up to* $t$ *corruptions and only uses a gradecast channel.*

*Proof.* Let $\Pi$ be a layered VSS protocol tolerating up to $t < n/2$ corruptions. We now show that applying our compiler (Figure 7) to $\Pi$ yields a protocol $\Pi'$ that is a moderated VSS protocol tolerating up to $t$ corruptions.

First, we show that **(i)** if all moderators $P_{M^k}$, $\mathsf{fin} > k \geq 0$, are honest, then every honest $P_i$ in $\mathcal{L}_{2\ell\mathsf{fin}}$ outputs $\mathsf{success}_i = 1$. By protocol description, $\mathsf{success}_i = 0$ only if $P_i$ computes $y = 1$ or $P_i$ received at least $t + 1$ warning bits $y_{j,i,a} = 1$ from parties in some committee $\mathcal{L}_a$. Thus, to prove claim **(i)**, it suffices to show that if all moderators are honest, no honest party computes a warning bit $y = 1$: An honest party in layer $\mathcal{L}_{2\ell k}$ ($\mathsf{fin} \geq k > 0$) computes $y = 1$ if and only if for some gradecast $\mathsf{GradeCast}_\ell$ it holds that $g'_i \neq 2$, or alternatively ($m_i \neq m'_i$ and $g_i = 2$). Whenever $P_{M^k}$ is honest, by validity of $\mathsf{GradeCast}_\ell$ we get that every honest party $P_i$ in $\mathcal{L}_{2\ell k}$ outputs $g'_i = 2$. Further, if $g_i = 2$ for some honest party $P_i$ in $\mathcal{L}_{2\ell k}$, by the graded consistency of $\mathsf{GradeCast}_{\ell,2\ell}$, we get that every honest party $P_j$ in layers $\mathcal{L}_{2\ell k}$ and $\mathcal{L}_{2\ell k-\ell}$ receives the same output $m = m_i$ from $\mathsf{GradeCast}_{\ell,2\ell}$. In particular, the moderator $P_{M^k}$ in $\mathcal{L}_{2\ell k-\ell}$ also receives $m$. If $P_{M^k}$ is honest, by validity of $\mathsf{GradeCast}_\ell$ we then get that every honest party $P_i$ in $\mathcal{L}_{2\ell k}$ will output $m'_i = m$. Thus, if $P_{M^k}$ is honest, no honest party from $\mathcal{L}_{2\ell k}$ computes $y = 1$. This proves **(i)**.

Next, we show **(ii)** if an honest $P_i$ outputs $\mathsf{success}_i = 1$, then our schemes has the layered VSS properties. Note that if an honest party $P_i$ outputs $\mathsf{success}_i = 1$ at the end of the sharing phase, then from every layer $\mathcal{L}_{2\ell k}$, $\mathsf{fin} > k > 0$, $P_i$ received at most $t$ warning bits $y_{j,i,2\ell k} = 1$. As $t < n/2$, we get that in every layer $\mathcal{L}_{2\ell k}$, there is at least one honest party $P_i$ that did not send a warning bit to the final committee. Hence, (by the protocol description) it must be that $g'_i = 2$ for this $P_i$. By graded consistency, every honest party $P_j$ in the same layer obtained $m'_j = m'_i$ (thus, we preserve the consistency of the broadcast used in

**Moderated Layered VSS - Sharing Phase**

**Public Parameters:** Let $|\mathsf{Share}|$ denote the number of layers in the sharing phase of $\Pi$, which is a layered VSS protocol with broadcast. Let $\mathsf{fin} = |\mathsf{Share}| - 1$ denote the final layer of $\Pi$'s sharing phase. Let $M = (M^0, \cdots, M^{\mathsf{fin}})$ denote the moderator index sequence.

**$\mathcal{L}_0$:** Let $r_i$ denote the randomness of $P_i$. Each $P_i$ executes

$$(\{m_{i,j}\}_{j\in[n]}, m_{i,\mathsf{bc}}) = \mathsf{NextMessage}_\Pi(i, 0, r_i, \bot),$$

$P_i$ enters $m_{i,\mathsf{bc}}$ into a gradecast protocol $\mathsf{GradeCast}_{\ell,2\ell}$ and sends $m_{i,j}$ to $P_j$ in $\mathcal{L}_{2\ell}$.

**$\mathcal{L}_{2\ell k+\ell,\ \mathsf{fin}>k\geq 0}$:** Party $P_{M^k}$: Upon receiving a message $m$ from $\mathsf{GradeCast}_{\ell,2\ell}$, input $m$ into $\mathsf{GradeCast}_\ell$.

**$\mathcal{L}_{2\ell k,\ \mathsf{fin}>k>0}$:** For any gradecast $\mathsf{GradeCast}_{\ell,2\ell}$ that started out from layer $\mathcal{L}_{2\ell(k-1)}$, let $(m_i, g_i)$ denote the output of party $P_i$ in $\mathcal{L}_{2\ell k}$, and let $(m'_i, g'_i)$ denote $P_i$'s output of the corresponding gradecast $\mathsf{GradeCast}_\ell$ that was initiated by the moderator in $\mathcal{L}_{2\ell k-\ell}$. Let $m'_{i,\mathsf{bc}}$ denote the set of all messages $m'_i$ received by $P_i$ in this way. Let $m_{j,i}$ further denote a message that $P_i$ received from party $P_j$ of layer $\mathcal{L}_{2\ell(k-1)}$ (if any). **(†)**
Each $P_i$ does the following:

- $P_i$ sends a warning bit $y = 1$ to each party of the final committee $\mathcal{L}_{2\ell\mathsf{fin}}$, if for any of the gradecasts, at least one of $g'_i \neq 2$ or ($m_i \neq m'_i$ and $g_i = 2$) holds.
- Additionally, $P_i$ executes

$$(\{m^*_{i,j}\}_{j\in[n]}, m^*_{i,\mathsf{bc}}) = \mathsf{NextMessage}_\Pi(i, k, r_i, \{m_{j,i}\}_{j\in[n]} \cup m'_{i,\mathsf{bc}}),$$

where $r_i$ denote the randomness of $P_i$. $P_i$ then enters $m^*_{i,\mathsf{bc}}$ into a gradecast protocol $\mathsf{GradeCast}_{\ell,2\ell}$ and sends $m^*_{i,j}$ to $P_j$ in $\mathcal{L}_{2(k+1)\ell}$.

**$\mathcal{L}_{2\ell\mathsf{fin}}$:** Let $m_{j,i}$, $m'_{i,\mathsf{bc}}$ and $y$ denote the messages received resp. the warning bit computed by party $P_i$ in the same way as in **(†)**. Let $P_i$ execute

$$s_i = \mathsf{OutputShare}_\Pi(i, r_i, \{m_{j,i}\}_{j\in[n]} \cup m'_{i,\mathsf{bc}}),$$

where $r_i$ is randomness of $P_i$. Further, let $y_{j,i,a}$ denote the warning bit sent to party $P_i$ by party $P_j$ in layer $\mathcal{L}_a$.
Then, $P_i$ outputs $(s_i, \mathsf{success} = 1)$, if $y = 0$ and for every layer $\mathcal{L}_a$, $P_i$ received at most $t$ warning bits $y_{j,i,a} = 1$. Otherwise, $P_i$ outputs $(\bot, \mathsf{success} = 0)$.

**Fig. 7.** Compiler: From Layered VSS with Broadcast to Moderated Layered VSS with Gradecast (No Broadcast).

$\Pi$). As for the gradecasts initiated by honest parties every $P_j$ outputs $g_j = 2$ and $m_j = m$ (where $m$ was the message gradecasted by the honest sender), it must be $m_i = m'_i$ (otherwise the honest $P_i$ would have sent a warning bit). Together, we get $m'_j = m'_i = m_i = m$. Thus, our compiler preserves the validity of $\Pi$'s broadcasts, and hence in this case we achieve the original VSS properties. □

Instantiating this construction with the VSS protocol from [29], Sec. 5, resp. with the one from [30], Sec. 5 and from [33], Sec. 8, as well as the gradecast protocols discussed in Sec. 4, we obtain the following results:

**Corollary 6.** *There exists a perfectly secure $(n, t, 14, 1)$-layered moderated VSS protocol, which uses 3 moderators for $t < n/3$.*

**Corollary 7.** *Assuming plain PKI for signatures and non-interactive homomorphic commitments, there exists a cryptographically secure $(n, t, 16, 1)$-layered moderated VSS with 2 moderators for $t < n/2$.*

**Corollary 8.** *Assuming plain PKI for signatures, there exists a cryptographically secure $(n, t, 60, 2)$-layered moderated VSS with 15 moderators for $t < n/2$.*

## 6 Random Coin from Moderated VSS

We now adapt the oblivious leader election protocol from [53] to construct a so-called *weak layered random coin protocol* assuming the existence of a constant-round layered weak consensus protocol and a constant round layered moderated VSS protocol. Formally, we define a weak layered coin protocol as follows:

**Definition 11 (Weak Layered Random Coin).** *A protocol $\Pi$ over $d_1 + d_2$ layers is a $(n, t, d_1, d_2)$ stat. (resp., comp.) secure weak layered random coin protocol with domain $\mathcal{D}$ and success probability $\delta$ if, for any unconditional (resp., PPT) adversary $\mathcal{A}$ corrupting up to $t$ parties in each layer, the probability that all honest parties output the same value is at least $\delta$, and, for any $b, b' \in \mathcal{D}$, the view of $\mathcal{A}$ during the first $d_1$ rounds given that all honest parties output $b$ is (comp.) indistinguishable from the case where all honest parties output $b'$ .*

**Theorem 10.** *Assuming an $(n, t, d_1, d_2)$-layered moderated VSS using $k$ moderators and an $(n, t, d_2)$-layered weak consensus, protocol* Coin *(Fig. 8) is a weak $(t, d_1, d_2)$-layered coin with domain $\mathbb{Z}_q$ and success probability $\delta := 0.99 \cdot \left(\frac{n-t}{n}\right)^k$.*

*Proof.* After round $d_1$, every moderator sequence $\boldsymbol{j}$ is either disqualified, i.e. for some $i \in [n]$, every honest party in $\mathcal{L}_{d_1}$ has obtained $\mathsf{success} = 0$ from $\mathsf{Share}_{i,\boldsymbol{j}}$, or the moderator sequence acted correctly, in which case the properties of moderated VSS guarantee that there exists a value $(a_{\boldsymbol{j}}, b_{\boldsymbol{j}}) \in \mathbb{Z}_{100 \cdot n^{2k}} \times \mathbb{Z}_q$ such that every honest $P_l \in \mathcal{L}_{d_1+d_2}$ will compute $\left(a_{\boldsymbol{j}}^{(l)}, b_{\boldsymbol{j}}^{(l)}\right) = (a_{\boldsymbol{j}}, b_{\boldsymbol{j}})$, no matter what the corrupt parties in $\mathcal{L}_{d_1}, \ldots, \mathcal{L}_{d_1+d_2}$ do. Moreover, $(a_{\boldsymbol{j}}, b_{\boldsymbol{j}})$ is independent of the view of the adversary up until round $d_1$ and, as some honest parties input uniformly random summands, uniformly distributed. Thus, if we denote by $\mathcal{T} \subseteq [n]^k$ the set of moderator sequences which are not disqualified in the above sense, there is a probability of at least $\delta = 0.99 \cdot \left(\frac{n-t}{n}\right)^k$ that all the $(a_{\boldsymbol{j}})_{\boldsymbol{j} \in \mathcal{T}}$ are pairwise distinct and that the minimum of them corresponds to a fully honest moderator sequence $\boldsymbol{j}^*$. In this event, all honest parties in $\mathcal{L}_{d_1+d_2}$ are guaranteed to output $b_{\boldsymbol{j}^*}$, which is uniformly random given the view of the adversary until round $d_1$. □

**Protocol** Coin

**Public Parameters:** $(n, t, d_1, d_2)$-layered moderated VSS protocol $(\mathsf{Share}, \mathsf{Reconstruct})$; $k < d_1$: the number of moderators used in $\mathsf{Share}$; $(n, t, d_2)$-layered weak consensus protocol $\mathsf{WC}$; Number of coin sides $q$.

$\mathcal{L}_0$**:** Every $P_i$ chooses independent uniformly random values $a_{i,\boldsymbol{j}} \in \mathbb{Z}_{100 \cdot n^{2k}}$ and $b_{i,\boldsymbol{j}} \in \mathbb{Z}_q$ for each $\boldsymbol{j} \in [n]^k$. Then, for each $i \in [n]$ and $\boldsymbol{j} \in [n]^k$, all parties execute a layered moderated VSS protocol instance, denoted $\mathsf{Share}_{i,\boldsymbol{j}}$, where $\boldsymbol{j}$ is the sequence of moderators and $P_i$ is the dealer with input $(a_{i,\boldsymbol{j}}, b_{i,\boldsymbol{j}})$

$\mathcal{L}_1, \ldots, \mathcal{L}_{d_1-1}$**:** All parties participate in $\mathsf{Share}_{i,\boldsymbol{j}}$ for all $i \in [n]$ and $\boldsymbol{j} \in [n]^k$

$\mathcal{L}_{d_1}$**:** For every $\boldsymbol{j} \in [n]^k$, one instance of weak consensus (denoted $\mathsf{WC}_{\boldsymbol{j}}$) is started. $P_i$ inputs 1 into $\mathsf{WC}_{\boldsymbol{j}}$ if $P_i$ output $\mathsf{success} = 1$ from $\mathsf{Share}_{l,\boldsymbol{j}}$ for every $l \in [n]$, otherwise $P_i$ inputs 0.
Additionally, for every $i \in [n]$, $\boldsymbol{j} \in [n]^k$, all parties start a protocol instance $\mathsf{Reconstruct}_{i,\boldsymbol{j}}$ to reconstruct the secret shared in $\mathsf{Share}_{i,\boldsymbol{j}}$.

$\mathcal{L}_{d_1+1}, \ldots, \mathcal{L}_{d_1+d_2-1}$**:** All parties participate in $\mathsf{Reconstruct}_{i,\boldsymbol{j}}$ and $\mathsf{WC}_{\boldsymbol{j}}$ for all $i \in [n]$ and $\boldsymbol{j} \in [n]^k$.

$\mathcal{L}_{d_1+d_2}$**:** Write $\left(a_{i,\boldsymbol{j}}^{(l)}, b_{i,\boldsymbol{j}}^{(l)}\right)$ for $P_l$'s outputs from $\mathsf{Reconstruct}_{i,\boldsymbol{j}}$ and $\mathsf{success}_{\boldsymbol{j}}^{(l)}$ for the value $P_l$ outputs from $\mathsf{WC}_{\boldsymbol{j}}$. Further define

$$a_{\boldsymbol{j}}^{(l)} := \sum_{i \in [n]} a_{i,\boldsymbol{j}}^{(l)} \mod 100 \cdot n^{2k},$$

$$b_{\boldsymbol{j}}^{(l)} := \sum_{i \in [n]} b_{i,\boldsymbol{j}}^{(l)} \mod q$$

Finally, $P_l$ outputs $b_{\boldsymbol{j}^{(l)}}^{(l)}$, where $\boldsymbol{j}^{(l)}$ is defined as

$$\boldsymbol{j}^{(l)} := \operatorname{argmin}\{a_{\boldsymbol{j}}^{(l)} \mid \boldsymbol{j} \in [n]^k, \mathsf{success}_{\boldsymbol{j}}^{(l)} = 1\}$$

(If there is no unique minimum, $P_j$ outputs a random value).

**Fig. 8.** Random coin protocol from weak consensus and moderated VSS

Instantiating this construction with the protocols from Sections 4 and 5, we obtain the results below. Note that in the $t < n/2$ case, our simplifying assumption that the reconstruction phase and weak consensus have equal round complexity $d_2$ is not met. This can be easily addressed, e.g. by having $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_1+2}$ execute $\mathsf{Reconstruct}$ (instead of $\mathcal{L}_{d_1}$ and $\mathcal{L}_{d_1+1}$).

**Corollary 9.** *There exists a weak $(n, t, 14, 1)$-layered coin protocol with success probability $\delta \geq 0.29$, for $t < n/3$.*

**Corollary 10.** *Assuming a PKI for signatures and non-interactive homomorphic commitments, there exists a computationally secure weak $(n, t, 16, 2)$-layered coin protocol with success probability $\delta \geq 0.24$, for $t < n/2$.*

**Corollary 11.** *Assuming a PKI for signatures, there exists a computationally secure weak $(n, t, 60, 2)$-layered coin protocol with success probability $\delta \geq 0.99 \cdot (1/2)^{15}$, for $t < n/2$.*

## 7 Layered Consensus

We now introduce our layered consensus protocol. In the following we assume a weak $(n, t, d_1, d_2)$-layered coin protocol $\mathsf{Coin}$ with success probability $\delta > 0$, and a weak consensus protocol $\mathsf{WC}$ running over $d_1$ layers. We write $D := d_1 + d_2$.

**Protocol** LayeredConsensus

**Inputs:** Every $P_i \in \mathcal{L}_0$ holds $x_i \in \{0, 1\}$
**Public Parameters:** Security parameter $\lambda$, $(n, t, d_1)$-layered weak consensus protocol $\mathsf{WC}$, $(n, t, d_1, d_2)$-layered weak coin protocol $\mathsf{Coin}$

$\mathcal{L}_0$**:** Input $x_i$ into $\mathsf{WC}_1$, start $\mathsf{Coin}_1$
**for** $\ell = 0, \ldots, \lambda$**:**
  $\mathcal{L}_{\ell D+1}, \ldots, \mathcal{L}_{\ell D+(d_1-1)}$**:** Participate in $\mathsf{WC}_\ell$ and $\mathsf{Coin}_\ell$
  $\mathcal{L}_{\ell D+d_1}$**:** Participate in $\mathsf{Coin}_\ell$. Additionally, every $P_i$ sends their output from $\mathsf{WC}_\ell$ to all parties in $\mathcal{L}_{\ell D+(d_1+d_2)}$,
  $\mathcal{L}_{\ell D+(d_1+1)}, \ldots, \mathcal{L}_{\ell D+(d_1+d_2-1)}$**:** Participate in $\mathsf{Coin}_\ell$
  $\mathcal{L}_{\ell D+(d_1+d_2)}$**:** Let $y_i^\ell$ denote the median of all values $P_i$ received from parties in $\mathcal{L}_{\ell D+d_1}$ (using $0 < \perp < 1$). Define $x_i^\ell$ to be the output from $\mathsf{Coin}_\ell$ if $y_i^\ell = \perp$, and $x_i^\ell := y_i^\ell$ otherwise.
  Then, if $\ell < \lambda$, $P_i$ inputs $x_i^\ell$ into $\mathsf{WC}_{\ell+1}$ and starts $\mathsf{Coin}_{\ell+1}$.
  Otherwise, $P_i$ terminates with output $x_i^\lambda$.

**Fig. 9.** Layered consensus protocol

**Theorem 11.** *The protocol* LayeredConsensus *(Figure 9) achieves consensus with probability* $1 - (1 - \delta/2)^\lambda$.

*Proof.* We first show validity: If all honest $P_i \in \mathcal{L}_0$ input $x_i = b$ for some $b \in \{0, 1\}$, then by validity of $\mathsf{WC}$, every honest party in $\mathcal{L}_{d_1}$ will output $b$ from $\mathsf{WC}_1$. Since $t < n/2$, this implies that every honest $P_i$ in $\mathcal{L}_{(d_1+d_2)}$ will compute $x_i^1 = b$. By induction, the same holds for all $x_i^\ell$ computed by honest $P_i$ in $\mathcal{L}_{(\ell+1)D}$ for $\ell = 1, \ldots, \lambda$.

Consistency can be shown as follows: For any $\ell = 0, \ldots, \lambda$, the values $y_i^\ell$ computed by honest parties in $\mathcal{L}_{\ell D+(d_1+d_2)}$ are guaranteed to be in the set of output values honest parties got from $\mathsf{WC}_\ell$ (since $t < n/2$). Therefore, they are either all $\perp$ (if every honest party output $\perp$ from $\mathsf{WC}_\ell$) or they are contained in a set $\{b, \perp\}$ for some $b \in \{0, 1\}$) which was determined by what happened up to $\mathcal{L}_{\ell D+d_1}$. Therefore, the probability that all honest parties in $\mathcal{L}_{\ell D+(d_1+d_2)}$ output $b$ from the $\mathsf{Coin}_\ell$ is at least $\delta/2$. If this happens, the argument used in proving

validity of LayeredConsensus proves that all honest parties in $\mathcal{L}_{(\lambda+1)D}$ output $b$. Finally, the probability that this happens at least once is $1-(1-\frac{\delta}{2})^{\lambda}$. □

By instantiating this construction with our protocols from Sections 4 and 6, we obtain the following results:

**Corollary 12.** *There exist layered consensus protocols with (1) statistical security for $t < n/3$ and no setup assumptions and (2) cryptographic security for $t < n/2$ assuming a plain PKI.*

Note that as in the previous section, our assumption that the weak consensus protocol runs for $d_1$ rounds is not met here. This can be adressed by a similar technique as discussed in Sec. 4.3.

## 8 Impossibility for $t \geq n/2$

We show the impossibility of layered broadcast for $t \geq n/2$. This is in contrast to the classical setting with a fixed number of parties, where broadcast is possible for any number of corruptions $t < n$ [35].

**Theorem 12.** *For $t \geq n/2$ corruptions, layered broadcast with a fixed number of rounds and error probability below $\frac{1}{4}$ is impossible.*

*Proof.* Assume there exists a (possibly authenticated, possibly randomized) broadcast protocol $\Pi$ running over $k$ layers and tolerating $t \geq n/2$ corruptions, such that validity and consistency hold with probability $1-\epsilon$ for some $\epsilon < \frac{1}{4}$.

Consider the following four execution scenarios:

- A The sender $P_{\text{sdr}} \in \mathcal{L}_0$ is honest with input 0. In every layer, there are $t$ corrupt parties who do nothing.
- B The sender $P_{\text{sdr}} \in \mathcal{L}_0$ is corrupt, and pretends to be honest with input 0 towards all honest parties. In layers $\mathcal{L}_1, \ldots, \mathcal{L}_{k-1}$, there are $t$ corrupt parties each, $n-t$ of which participate (together with $P_s$) in a simulated parallel execution $\Pi$ with input 1 by $P_s$. Communication with the honest parties is blocked, up until the second-to-last layer, where all corrupt parties send the messages from their simulated execution to one honest party (say $P_1 \in \mathcal{L}_k$).
- C Analogous to B, but with the input values switched, i.e. $P_{\text{sdr}}$ sends 1 to the honest parties, and 0 in the simulated execution of $\Pi$.
- D Analogous to A but with input 1.

By validity, all honest parties must output 0 in scenario A with probability $\geq 1-\epsilon$. Since all honest parties in $\mathcal{L}_k$ (except $P_1$) have identical views in executions A and B, they must output 0 in scenario B with probability $\geq 1-\epsilon$ as well. Thus, by consistency, $P_1$ outputs 0 in scenario B with probability $\geq 1-2\epsilon$. But $P_1$ has identical views in scenario B and C, so by the same argument all other honest parties in $\mathcal{L}_k$ must output 0 in scenario C with probability $\geq 1-3\epsilon$. Finally, all honest parties in $\mathcal{L}_k$ except $P_1$ have identical views in scenarios C and D, so honest parties should output 0 with probability $\geq 1-3\epsilon$ in D as well. For $\epsilon < \frac{1}{4}$, this contradicts validity as honest parties must output 1 with probability $1-\epsilon > 3\epsilon$ in scenario D. □

## 9 Protocol SMT for Secure Message Transmission

We now describe our secure message transmission protocol SMT, defined in Figure 10, which realizes the functionality $\mathcal{F}_{\text{smt}}$ (Definition 4). At a high level, the scheme lets the sender $P_{\text{sdr}} \in \mathcal{L}_0$ secret-share the original secret message using a Shamir-sharing protocol with commitments to shares. Every intermediate layer performs a resharing of the secret by distributing sub-shares with commitments, and the receiving layer performs a non-trivial method to select subsets of sub-shares to combine into new shares without increasing the state size at each step. Finally, the second to last layer reconstructs the message towards $P_{\text{rcv}} \in \mathcal{L}_d$.

**Protocol** SMT

**Inputs:** A sender $P_{\text{sdr}} \in \mathcal{L}_0$ holds a secret $s \in \mathbb{F}_p$, Receiver $P_{\text{rcv}} \in \mathcal{L}_d$.

$\mathcal{L}_0, \mathcal{L}_1$**:** $P_{\text{sdr}}$ shares $s$ towards $\mathcal{L}_1$ using the protocol Share (Figure 11) .
$\mathcal{L}_r, \mathcal{L}_{r+1}$ $(1 \leq r \leq d-2)$**:** Parties from the "old" layer $\mathcal{L}_r$ and the "new" layer $\mathcal{L}_{r+1}$ execute the protocol Reshare (Figure 13).
$\mathcal{L}_{d-1}, \mathcal{L}_d$**:** Parties of layer $\mathcal{L}_{d-1}$ reconstruct $s$ towards $P_{\text{rcv}} \in \mathcal{L}_d$ using the Open protocol (Figure 12)

**Fig. 10.** Secure Message Transmission protocol

We prove the following result in Appendix B.2.

**Theorem 13.** *Assuming non-interactive homomorphic commitments, protocol* SMT *realizes* $\mathcal{F}_{\text{smt}}$ *for* $d$ *layers with communication* $\mathcal{O}(d \cdot \binom{n}{t+1} \cdot n^3)$ *and is secure up to* $t < n/2$ *corruptions per layer*

**Optimization.** Protocol SMT as presented has a communication complexity $\Omega(\binom{n}{t+1} \cdot n^3)$ per layer. We introduce an optimization, called SMT$^*$, which is applicable when $(t+1)^2 \leqslant n$. It consists of replacing the set $\mathcal{Q}$ used within the Share, Reshare and Open protocols by some fixed set of $t+1$ pairwise disjunct subsets of $[n]$, each of size $t+1$. If at most $t$ parties are corrupted, at least one of these is guaranteed to be fully honest, thus preserving security.

**Corollary 13 (Corollary 3).** *Assuming non-interactive homomorphic commitments, protocol* SMT$^*$ *realizes* $\mathcal{F}_{\text{smt}}$ *for* $d$ *layers with polynomial communication* $\mathcal{O}(d \cdot n^{\frac{3}{2}})$*, and is secure up to* $t \leq \sqrt{n} - 1$ *corruptions per layer.*

### 9.1 Proactive Secret Sharing Scheme

We now describe the proactive secret sharing scheme, consisting on the sub-protocols Share, Open and Reshare for initially sharing the secret by the sender, reconstructing the secret in the end towards the receiver, and re-sharing the secret throughout intermediate layers (see Figures 11 to 13) .

*Notation.* We write $[n] := \{1, \ldots, n\}$. Bold symbols such as $\mathbf{C}$ are vectors in $\mathbb{G}^n$ (where $\mathbb{G}$ is the commitment space, see Definition 13). The $i$-th component of $\mathbf{C}$ is denoted by $\mathbf{C}[i]$. When we say $C, \mathbf{C}$ are on a degree-$t$ polynomial, we mean that there exists a degree-$t$ polynomial $f$ with $f(0) = C$ and $f(i) = \mathbf{C}[i]$ for $i \in [n]$. We write $\mathcal{Q} := \binom{[n]}{t+1}$ (note that this applies to $\mathsf{SMT}$, for $\mathsf{SMT}^*$ we use $\mathcal{Q} := \{\{k(t+1)+1, \ldots, (k+1)(t+1)\} \mid k = 0, \ldots, t\}$). Sometimes we write $I \subseteq \mathcal{L}$ for $I \in \mathcal{Q}$ to denote $\{P_i \in \mathcal{L} \mid i \in I\}$ or, analogously $P_i \in I$ instead of $P_i$ with $i \in I$.

**Sharing State.** Before describing the protocol, let us introduce the sharing state that the parties maintain throughout all intermediate layers (in-between the sender and receiver layers): Let $\mathcal{L}$ be such an intermediate layer. Every honest party $P_i \in \mathcal{L}$ holds a commitment $C$ of the original secret from the sender, and a list of size $|\mathcal{Q}|$. For every $\ell \in \mathcal{Q}$, the $\ell$-th value in the list is either $\perp$ or a tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ where $s_i^\ell, r_i^\ell \in \mathbb{F}_p$ are two Shamir shares and $\mathbf{C}_i^\ell \in \mathbb{G}^n$ is a vector of commitments. We will refer to $\ell$ as the *label* of the tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$.

**Invariants.** The sharing state held by honest parties in an intermediate layer $\mathcal{L}$ satisfies two invariants, denoted $\mathsf{validSharing}$ and $\mathsf{consistentCommitments}$. The first invariant is defined w.r.t. a commitment $C \in \mathbb{G}$, and states that every commitment vector held by an honest party is consistent with $C$:

$\mathsf{consistentCommitments}(\mathcal{L}, C)$ : Every honest $P_i \in \mathcal{L}$ has stored the commitment $C$, and for any tuple $(\ell, \mathbf{C}_i^\ell, \cdot, \cdot)$ appearing in the list of an honest $P_i \in \mathcal{L}$, we have that $C$, $\mathbf{C}_i^\ell$ lie on a polynomial of degree at most $t$.

The second invariant states that there is one label $\ell \in \mathcal{Q}$ for which all honest $P_i \in \mathcal{L}$ hold the same commitment vector $\mathbf{C}_i^\ell$ and corresponding shares:

$\mathsf{validSharing}(\mathcal{L})$*:* For any set $H \subseteq \mathcal{L}$ of honest parties, there exists a label $\ell \in \mathcal{Q}$ and $\mathbf{C}^\ell \in \mathbb{G}^n$ such that every $P_i \in H$ has stored a tuple $(\ell, \mathbf{C}^\ell, s_i^\ell, r_i^\ell)$ where $(s_i^\ell, r_i^\ell)$ is an opening to $\mathbf{C}^\ell[i]$.

*Protocol Descriptions.* The protocol $\mathsf{Share}$ (Figure 11) simply lets the sender with secret input $s$ compute a Shamir-sharing $(s_1, \ldots, s_n)$ of $s$ as well as a Shamir-sharing $(r_1, \ldots, r_n)$ of a random value $r$. The sender computes a commitment $C$ to $s$ as well as commitments $C_i$ to each share $s_i$ of $s$, using randomness $r$ resp. using the share $r_i$ of $r$. To each $P_i \in \mathcal{L}_1$, the sender sends all the computed commitments and the private opening $(s_i, r_i)$. Then, $P_i$ stores the commitment $C$, and for each label $\ell \in \mathcal{Q}$, a separate tuple $(\ell, \mathbf{C}_i^\ell, s_i, r_i)$, where $\mathbf{C}_i^\ell = (C_1, \ldots, C_n)$. Note that at this point, for an honest sender, all tuples held by a given party contain the same vector of commitments and the same shares, but this might not be the case after the invocations of the $\mathsf{Reshare}$ protocol.

In protocol $\mathsf{Open}$ (Figure 12) , every party $P_i \in \mathcal{L}_0$ (we denote the starting layer $\mathcal{L}_0$) holds a commitment $C$ to the original sender's secret, and for each label $\ell \in \mathcal{Q}$, a separate tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$. Even after the intermediate $\mathsf{Reshare}$ steps, the tuples will satisfy the two invariants outlined above if $P_{\text{sdr}}$ was honest. Each $P_i \in \mathcal{L}_0$ simply sends $C$ and all tuples $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ to the receiver $P_{\text{rcv}} \in \mathcal{L}_1$. The receiver checks if, for some label $\ell \in \mathcal{Q}$, there exists $\mathbf{C}^\ell \in \mathbb{G}^n$ such that $C$ and $\mathbf{C}^\ell$ lie on a degree-$t$ polynomial, and enough $P_i$ sent shares $s_i, r_i$ which form

**Protocol** Share

**Public Parameters:** sender $P_{\text{sdr}} \in \mathcal{L}_0$, corruption threshold $t$, set $\mathcal{Q}$.
**Inputs:** $P_{\text{sdr}}$ holds a secret message $s \in \mathbb{F}_p$.

$\mathcal{L}_0$**:** $P_{\text{sdr}}$ samples $r \xleftarrow{\$} \mathbb{F}_p$ and Shamir sharings $(s_1, \ldots, s_n)$ and $(r_1, \ldots, r_n)$ of $s$ and $r$ respectively. Then, $P_{\text{sdr}}$ computes $C := \mathsf{Com}(s, r)$ and $\mathbf{C} := (\mathsf{Com}(s_i, r_i))_{i \in [n]}$. Finally, $P_{\text{sdr}}$ sends $(C, \mathbf{C}, s_j, r_j)$ to every $P_j \in \mathcal{L}_1$

$\mathcal{L}_1$**:** Every $P_i \in \mathcal{L}_1$, upon receiving $(C, \mathbf{C}, s_i, r_i)$ from $P_{\text{sdr}}$ $P_i$ outputs $\big(C, \{(\ell, \mathbf{C}, s_i, r_i)\}_{\ell \in \mathcal{Q}}\big)$.

**Fig. 11.** Sharing Protocol

a correct opening of the $i$-th component of $\mathbf{C}^\ell$. If $t+1$ such shares were received, $P_{\text{rcv}}$ is able to reconstruct an opening of $C$.

**Protocol** Open

**Public Parameters:** corruption threshold $t$, set $\mathcal{Q}$, receiving party $P_{\text{rcv}} \in \mathcal{L}_1$. **Inputs:** Every $P_i \in \mathcal{L}_0$ holds a set of tuples $\{(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)\}_{\ell \in \mathcal{Q}}$, and a commitment $C$.

$\mathcal{L}_0$**:** Every $P_i$ sends $C$ and $\{(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)\}_{\ell \in \mathcal{Q}}$ to $P_{\text{rcv}}$.

$\mathcal{L}_1$**:** $P_{\text{rcv}}$ : Set $C$ to the value received from a majority of parties in $\mathcal{L}_0$. Then, for a tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$ received from $P_i \in \mathcal{L}_0$, consider it to be *valid* iff the following hold:
(a) $C$ and $\mathbf{C}_i^\ell$ lie on a polynomial of degree $t$, and
(b) $(s_i^\ell, r_i^\ell)$ is an opening of the $i$-th component of $\mathbf{C}_i^\ell$.
If there exists a label $\ell$ and $\mathbf{C}^\ell \in \mathbb{G}^n$ such that $P_{\text{rcv}}$ got valid tuples $(\ell, \mathbf{C}^\ell, s_i^\ell, r_i^\ell)$ from all $P_i$ with $i \in I$ for some $I \in \mathcal{Q}$, interpolate a degree-$t$ polynomial $f$ through $\{(i, s_i^\ell)\}_{i \in I}$ and output $f(0)$.
Otherwise, output $\perp$.

**Fig. 12.** Open protocol

In protocol Reshare (Figure 13), every party $P_i \in \mathcal{L}_0$ holds a commitment $C$ and, for each label $\ell \in \mathcal{Q}$, a separate tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)$. Then, for every $\ell \in \mathcal{Q}$, every $P_i$ will distribute Shamir sharings of $s_i^\ell, r_i^\ell$ among $\mathcal{L}_1$, together with commitments to the new subshares of $s_i^\ell$ (using the shares of $r_i^\ell$ as randomness). Parties in $\mathcal{L}_1$ iterate through all subsets $I \in \mathcal{Q}$, and use the commitments to check if there is a sharing which all parties in $I$ reshared. Then, they select just one of these (the one with maximum label $\ell$), and use the received subshares and commitments to subshares to interpolate new shares, as well as a vector of commitments to all new shares, which they will store under the label $I$. Essentially, this amounts to performing the classical resharing technique, which may fail if some parties in $I$ are corrupt. However, one of the $I \in \mathcal{Q}$ is guaranteed to be fully honest. Thus, we can show that if the previous sharing state was valid

(i.e. fulfilled the two invariants stated above), the same holds for the new sharing state. See Appendix B for proofs.

**Protocol** Reshare

**Public:** threshold $t$, set $\mathcal{Q} = \{S \subset [n] : |S| = t+1\}$.
**Input of each old party** $P_i \in \mathcal{L}_{\mathsf{old}}$**:** set of tuples $\{(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell)\}_{\ell \in \mathcal{Q}}$, commitment $C$ to the secret.

**Old party** $P_i \in \mathcal{L}_{\mathsf{old}}$**:**

Send $C$ to each $P_j \in \mathcal{L}_{\mathsf{new}}$. Further, for each tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell) \neq \perp$:

1. Sample degree-$t$ Shamir sharings $(s_{i,1}^\ell, \ldots, s_{i,n}^\ell) \leftarrow \mathsf{Share}(s_i^\ell)$ and $(r_{i,1}^\ell, \ldots, r_{i,n}^\ell) \leftarrow \mathsf{Share}(r_i^\ell)$.
2. For all $j \in [n]$, set $\widetilde{\mathbf{C}}_i^\ell[j] \leftarrow \mathsf{Com}(s_{i,j}^\ell, r_{i,j}^\ell)$.
3. Send to each $P_j \in \mathcal{L}_{\mathsf{new}}$:

$$\left(\ell, \widetilde{\mathbf{C}}_i^\ell, s_{i,j}^\ell, r_{i,j}^\ell\right)$$

**New party** $P_j \in \mathcal{L}_{\mathsf{new}}$**:**

1. Set $C$ to the value received from a majority of parties in $\mathcal{L}_{\mathsf{old}}$.
2. For each $I \in \mathcal{Q}$:
   (a) Let $\ell^*$ be the maximum label such that all the following hold:
      i. Every $P_i \in I$ sent a tuple $(\ell^*, \widetilde{\mathbf{C}}_i^*, s_{i,j}^*, r_{i,j}^*)$
      ii. For every $i \in I$: $\widetilde{\mathbf{C}}_i^*[j] = \mathsf{Com}(s_{i,j}^*, r_{i,j}^*)$
      iii. For every $i \in I$, there exists $C_i^* \in \mathbb{G}$ such that $C_i^*$, $\widetilde{\mathbf{C}}_i^*$ lie on a degree-$t$ polynomial.
      iv. The points $(0, C)$, $\{(i, C_i^*)\}_{i \in I}$ lie on a degree-$t$ polynomial.
   If no such $\ell^*$ exists, store $\perp$ and proceed to the next set.
   (b) Let $f_j^I(x)$ (resp. $g_j^I(x)$) be the degree-$t$ polynomial going through the points $\{(i, s_{i,j}^*)\}_{i \in I}$ (resp. $\{(i, r_{i,j}^*)\}_{i \in I}$). Let $h_j^I(x)$ be the degree-$t$ polynomial going through the points $\{(i, \widetilde{\mathbf{C}}_i^*)\}_{i \in I}$.
   (c) Compute

$$(s_j^{\mathsf{new},I}, r_j^{\mathsf{new},I}) \leftarrow \left(f_j^I(0), g_j^I(0)\right)$$

$$\mathbf{C}_j^{\mathsf{new},I}[k] \leftarrow h_j^I(0) \text{ (for all } k \in [n])$$

   (d) Store $\left(I, \mathbf{C}_j^{\mathsf{new},I}, s_j^{\mathsf{new},I}, r_j^{\mathsf{new},I}\right)$ for $I$.

**Fig. 13.** Re-sharing from old committee $\mathcal{L}_{\mathsf{old}}$ to new committee $\mathcal{L}_{\mathsf{new}}$.

## 10 Conclusions

We gave a full characterization on the feasibility of broadcast and consensus in the layered setting. Our protocols can be directly plugged into prior MPC pro-

tocols with dynamic committees to achieve the results without the need for ideal broadcast channels. In particular, a complete characterization of YOSO MPC without assuming broadcast channels. In addition, we construct secure message transmission (SMT) protocols for the layered MPC model which do not assume broadcast or PKI, and which have linear communication complexity in the number of intermediate layers. Using homomorphic commitments, we propose a secure message transmission protocol for $t < n/2$ (with communication complexity exponential in $n$) and one for $t \leq \sqrt{n} - 1$ (with communication complexity polynomial in $n$). Using the latter SMT, we obtain the first polynomial Layered MPC protocol with guaranteed output and without setup for the same threshold. Our work leaves open the question of whether polynomial-communication Layered MPC with guaranteed output delivery is possible without setup nor broadcast for the optimal $t < n/3$.


### Acknowledgments

Gabriel Dettling and Chen-Da Liu-Zhang were supported by the SNSF grant number 2000-1-243069. Elisaweta Masserova was supported by the CBI postdoctoral fellowship. Matthieu Rambaud was supported by ANR-21-CE39-0009-BARRACUDA.


## References


1. Abraham, I., Chan, T.-H.H., Dolev, D., Nayak, K., Pass, R., Ren, L., Shi, E.: Communication Complexity of Byzantine Agreement, Revisited. Robinson, P., Ellen, F. (eds.) 38th ACM PODC, pp. 317–326. ACM (2019). https://doi.org/10.1145/3293611.3331629
2. Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous Byzantine Agreement with Expected $O(1)$ Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience. International Conference on Financial Cryptography and Data Security, pp. 320–334 (2019)
3. Acharya, A., Hazay, C., Kolesnikov, V., Prabhakaran, M.: Malicious Security for SCALES - Outsourced Computation with Ephemeral Servers. Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX. LNCS, vol. 14928, pp. 3–38. Springer, Heidelberg (2024). https://doi.org/10.1007/978-u3-u031-u68400-u5_1
4. Acharya, A., Hazay, C., Kolesnikov, V., Prabhakaran, M.: SCALES - MPC with Small Clients and Larger Ephemeral Servers. Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, pp. 502–531. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u22365-u5_18
5. Acharya, A., Hazay, C., Kolesnikov, V., Prabhakaran, M.: Towards Building Efficient SCALES Protocols. IACR Cryptol. ePrint Arch. (2025). https://eprint.iacr.org/2025/1487
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). 20th ACM STOC, pp. 1–10. ACM Press (1988). https://doi.org/10.1145/62212.62213

7. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a Public Blockchain Keep a Secret? Pass, R., Pietrzak, K. (eds.) Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I. LNCS, vol. 12550, pp. 260–290. Springer, Heidelberg (2020). https://doi.org/10.1007/978-u3-u030-u64375-u1_10
8. Berman, P., Garay, J.A., Perry, K.J.: "Bit Optimal Distributed Consensus". Computer Science: Research and Applications. Ed. by R. Baeza-Yates and U. Manber. Boston, MA: Springer US, 1992, pp. 313–321. ISBN: 978-1-4615-3422-8. https://doi.org/10.1007/978-u1-u4615-u3422-u8_27.
9. Bhadauria, R., Chiang, J.H., Ravi, D., Sternad, J., Yakoubov, S.: Minimizing the Use of the Honest Majority in YOSO MPC with Guaranteed Output Delivery. IACR Cryptol. ePrint Arch. (2024). https://eprint.iacr.org/2024/2059
10. Bhangale, A., Liu-Zhang, C.-D., Loss, J., Nayak, K.: Efficient Adaptively-Secure Byzantine Agreement for Long Messages. Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, pp. 504–525. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u22963-u3_17
11. Bhangale, A., Liu-Zhang, C.-D., Loss, J., Nayak, K., Yandamuri, S.: Leader Election with Poly-Logarithmic Communication Per Party. Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part II. LNCS, pp. 37–68. Springer, Cham (2025). https://doi.org/10.1007/978-u3-u032-u01878-u6_2
12. Bienstock, A., Escudero, D., Polychroniadou, A.: On Linear Communication Complexity for (Maximally) Fluid MPC. Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 263–294. Springer, Cham (2023). https://doi.org/10.1007/978-u3-u031-u38557-u5_9
13. Bienstock, A., Escudero, D., Polychroniadou, A.: Perfectly Secure Fluid MPC with Abort and Linear Communication Complexity. IACR Commun. Cryptol. **1**(4), 28 (2024). https://doi.org/10.62056/AESG89N4E
14. Blum, E., Boyle, E., Cohen, R., Liu-Zhang, C.-D.: Communication lower bounds for cryptographic broadcast protocols: E. Blum et al. Distributed Computing **38**(1), 1–17 (2025)
15. Blum, E., Katz, J., Liu-Zhang, C.-D., Loss, J.: Asynchronous Byzantine Agreement with Subquadratic Communication. Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, pp. 353–380. Springer, Cham (2020). https://doi.org/10.1007/978-u3-u030-u64375-u1_13
16. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. International conference on the theory and application of cryptology and information security, pp. 514–532 (2001)
17. Boyle, E., Cohen, R., Goel, A.: Breaking the $O(\sqrt{(}n))$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party. Miller, A., Censor-Hillel, K., Korhonen, J.H. (eds.) 40th ACM PODC, pp. 319–330. ACM (2021). https://doi.org/10.1145/3465084.3467897
18. Braun, L., Damgård, I., Orlandi, C.: Secure Multiparty Computation from Threshold Encryption Based on Class Groups. Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 613–645. Springer, Cham (2023). https://doi.org/10.1007/978-u3-u031-u38557-u5_20
19. Campanelli, M., David, B., Khoshakhlagh, H., Konring, A., Nielsen, J.B.: Encryption to the Future - A Paradigm for Sending Secret Messages to Future (Anonymous) Committees. Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS,

pp. 151–180. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u22969-u5_6
20. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology **13**(1), 143–202 (2000). https://doi.org/10.1007/s001459910006
21. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. 42nd FOCS, pp. 136–145. IEEE Computer Society Press (2001). https://doi.org/10.1109/SFCS.2001.959888
22. Canetti, R., Kolby, S., Ravi, D., Soria-Vazquez, E., Yakoubov, S.: Taming Adaptivity in YOSO Protocols: The Modular Way. Theory of Cryptography (2023)
23. Cascudo, I., David, B., Garms, L., Konring, A.: YOLO YOSO: Fast and Simple Encryption and Secret Sharing in the YOSO Model. Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, pp. 651–680. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u22963-u3_22
24. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. OSDI, pp. 173–186 (1999)
25. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols (Abstract) (Informal Contribution). Pomerance, C. (ed.) CRYPTO'87. LNCS, p. 462. Springer, Berlin, Heidelberg (1988). https://doi.org/10.1007/3-u540-u48184-u2_43
26. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theoretical Computer Science **777**, 155–183 (2019)
27. Choudhuri, A.R., Goel, A., Green, M., Jain, A., Kaptchuk, G.: Fluid MPC: Secure Multiparty Computation with Dynamic Participants. Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, pp. 94–123. Springer, Cham, Virtual Event (2021). https://doi.org/10.1007/978-u3-u030-u84245-u1_4
28. Damgård, I.B., Kamp, S.H., Loss, J., Nielsen, J.B.: Asynchronous YOSO a la Paillier. IACR Cryptol. ePrint Arch. (2025)
29. David, B., Deligios, G., Goel, A., Ishai, Y., Konring, A., Kushilevitz, E., Liu-Zhang, C.-D., Narayanan, V.: Perfect MPC over Layered Graphs. Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, pp. 360–392. Springer, Cham (2023). https://doi.org/10.1007/978-u3-u031-u38557-u5_12
30. Deligios, G., Goel, A., Liu-Zhang, C.-D.: Maximally-Fluid MPC with Guaranteed Output Delivery, Cryptology ePrint Archive, Paper 2023/415 (2023). https://eprint.iacr.org/2023/415.
31. Deligios, G., Hirt, M., Liu-Zhang, C.-D.: Round-Efficient Byzantine Agreement and Multi-party Computation with Asynchronous Fallback. Nissim, K., Waters, B. (eds.) TCC 2021, Part I. LNCS, pp. 623–653. Springer, Cham (2021). https://doi.org/10.1007/978-u3-u030-u90459-u3_21
32. Deligios, G., Klasovita, I., Liu-Zhang, C.-D.: Asymptotically Optimal Early Termination for Dishonest Majority Broadcast. Fehr, S., Fouque, P.-A. (eds.) EUROCRYPT 2025, Part V. LNCS, pp. 116–144. Springer, Cham (2025). https://doi.org/10.1007/978-u3-u031-u91092-u0_5
33. Deligios, G., Konring, A., Liu-Zhang, C.-D., Narayanan, V.: Statistical Layered MPC. Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part IV. LNCS, pp. 362–394. Springer, Cham (2024). https://doi.org/10.1007/978-u3-u031-u78023-u3_12
34. Desmedt, Y., Jajodia, S.: Redistributing secret shares to new access structures and its applications, (1997). Tech Report, George Mason U.
35. Dolev, D., Strong, H.R.: Authenticated algorithms for Byzantine agreement. SIAM Journal on Computing **12**(4), 656–666 (1983)

36. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074
37. Escudero, D., Masserova, E., Polychroniadou, A.: Brief Announcement: Towards Scalable YOSO MPC via Packed Secret-Sharing. Balliu, A., Kuhn, F. (eds.) Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025, pp. 162–165. ACM (2025). https://doi.org/10.1145/3732772.3733556
38. Escudero, D., Masserova, E., Polychroniadou, A.: Towards Scalable YOSO MPC via Packed Secret-Sharing. IACR Cryptol. ePrint Arch. (2025). https://eprint.iacr.org/2025/635
39. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous Byzantine agreement. SIAM Journal on Computing **26**(4), 873–933 (1997)
40. Fitzi, M., Liu-Zhang, C.-D., Loss, J.: A New Way to Achieve Round-Efficient Byzantine Agreement. Miller, A., Censor-Hillel, K., Korhonen, J.H. (eds.) 40th ACM PODC, pp. 355–362. ACM (2021). https://doi.org/10.1145/3465084.3467907
41. Garay, J.A.: Reaching (and Maintaining) Agreement in the Presence of Mobile Faults (Extended Abstract). Tel, G., Vitányi, P.M.B. (eds.) Distributed Algorithms, 8th International Workshop, WDAG '94, Terschelling, The Netherlands, September 29 - October 1, 1994, Proceedings. LNCS, vol. 857, pp. 253–264. Springer, Heidelberg (1994). https://doi.org/10.1007/BFB0020438
42. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. PODC (1998)
43. Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J.B., Rabin, T., Yakoubov, S.: YOSO: You Only Speak Once - Secure MPC with Stateless Ephemeral Roles. Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, pp. 64–93. Springer, Cham, Virtual Event (2021). https://doi.org/10.1007/978-u3-u030-u84245-u1_3
44. Ghinea, D., Goyal, V., Liu-Zhang, C.-D.: Round-Optimal Byzantine Agreement. Eurocrypt 2022, Trondheim, Norway (2022)
45. Ghinea, D., Klasovitá, I., Liu-Zhang, C.-D.: Round-Optimal Byzantine Agreement Without Trusted Setup. Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 427–456 (2026)
46. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. Aho, A. (ed.) 19th ACM STOC, pp. 218–229. ACM Press (1987). https://doi.org/10.1145/28395.28420
47. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on computing **17**(2), 281–308 (1988)
48. Goyal, V., Kothapalli, A., Masserova, E., Parno, B., Song, Y.: Storing and Retrieving Secrets on a Blockchain. Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I. LNCS, vol. 13177, pp. 252–282. Springer, Heidelberg (2022). https://doi.org/10.1007/978-u3-u030-u97121-u2_10
49. Goyal, V., Masserova, E., Parno, B., Song, Y.: Blockchains Enable Non-interactive MPC. Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, pp. 162–193. Springer, Cham (2021). https://doi.org/10.1007/978-u3-u030-u90453-u1_6

50. Guo, Y., Pass, R., Shi, E.: Synchronous, with a Chance of Partition Tolerance. Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, pp. 499–529. Springer, Cham (2019). https://doi.org/10.1007/978-u3-u030-u26948-u7_18
51. Hirt, M., Maurer, U.M.: Player Simulation and General Adversary Structures in Perfect Multiparty Computation. Journal of Cryptology **13**(1), 31–60 (2000). https://doi.org/10.1007/s001459910003
52. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). International journal of information security **1**, 36–63 (2001)
53. Katz, J., Koo, C.-Y.: On Expected Constant-Round Protocols for Byzantine Agreement. Dwork, C. (ed.) CRYPTO 2006. LNCS, pp. 445–462. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_27
54. Kolby, S., Ravi, D., Yakoubov, S.: Constant-Round YOSO MPC Without Setup. IACR Commun. Cryptol. **1**(3), 30 (2024). https://doi.org/10.62056/AE5W4FE-u3
55. Lamport, L., Fischer, M.: Byzantine generals and transaction commit protocols. Unpublished (1982)
56. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems **4**(3), 382–401 (1982)
57. Liu-Zhang, C.-D.: A Unified View on MPC with Dynamic Committees: C.-D. Liu-Zhang. Journal of the Indian Institute of Science (2026)
58. Liu-Zhang, C.-D., Masserova, E., Ribeiro, J., Soni, P., Thyagarajan, S.A.K.: Efficient Distributed Randomness Generation from Minimal Assumptions Where PArties Speak Sequentially Once. Fehr, S., Fouque, P.-A. (eds.) EUROCRYPT 2025, Part V. LNCS, pp. 176–206. Springer, Cham (2025). https://doi.org/10.1007/978-u3-u031-u91092-u0_7
59. Liu-Zhang, C.-D., Masserova, E., Ribeiro, J., Soni, P., Thyagarajan, S.A.K.: Improved YOSO Randomness Generation with Worst-Case Corruptions. Clark, J., Shi, E. (eds.) FC 2024, Part II. LNCS, pp. 73–89. Springer, Cham (2024). https://doi.org/10.1007/978-u3-u031-u78679-u2_4
60. Micali, S., Vaikuntanathan, V.: Optimal and player-replaceable consensus with an honest majority. (2017)
61. Nielsen, J.B., Ribeiro, J.L., Obremski, M.: Public Randomness Extraction with Ephemeral Roles and Worst-Case Corruptions. Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, pp. 127–147. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u15802-u5_5
62. Pass, R., Shi, E.: The Sleepy Model of Consensus. Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, pp. 380–409. Springer, Cham (2017). https://doi.org/10.1007/978-u3-u319-u70697-u9_14
63. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. Feigenbaum, J. (ed.) CRYPTO'91. LNCS, pp. 129–140. Springer, Berlin, Heidelberg (1992). https://doi.org/10.1007/3-u540-u46766-u1_9
64. Pfitzmann, B., Waidner, M.: Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. IBM (1996)
65. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). 21st ACM STOC, pp. 73–85. ACM Press (1989). https://doi.org/10.1145/73007.73014
66. Rachuri, R., Scholl, P.: Le Mans: Dynamic and Fluid MPC for Dishonest Majority. Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, pp. 719–749. Springer, Cham (2022). https://doi.org/10.1007/978-u3-u031-u15802-u5_25

67. Schnorr, C.-P.: Efficient identification and signatures for smart cards. Advances in Cryptology—CRYPTO'89 Proceedings 9, pp. 239–252 (1990)
68. Yao, A.C.-C.: How to Generate and Exchange Secrets (Extended Abstract). 27th FOCS, pp. 162–167. IEEE Computer Society Press (1986). https://doi.org/10.1109/SFCS.1986.25

## A Proofs regarding Weak and Graded Consensus

We restate the theorems and show the missing proofs for the protocols from section 4.

**Theorem 14 (Theorem 5).** *Protocol* WC *defined in Figure 2 is an* $(n, t, 1)$*-layered weak consensus protocol for any* $t < \frac{n}{3}$.

*Proof.* First, note that if all honest parties in $\mathcal{L}_0$ hold the same value $x$, then, as there are at least $n-t$ honest parties in $\mathcal{L}_0$, every party in $\mathcal{L}_1$ receives $x$ from at least $n-t$ parties. Additionally, in this case at most $t$ parties in $\mathcal{L}_0$ send values different from $x$. As $t < \frac{n}{3}$, we get $t < n-t$. Thus, every honest party in $\mathcal{L}_1$ outputs $x$. Hence, validity holds.

Second, if an honest party in $\mathcal{L}_1$ outputs a value $y \neq \perp$, then it received $y$ from at least $n-t$ parties of $\mathcal{L}_0$, hence in particular at least $n-2t$ honest parties sent $y$. Thus, there are at most $n-(n-2t) = 2t$ parties in $\mathcal{L}_0$ that could send some $y' \neq y$, and as $2t < n-t$ for $t < \frac{n}{3}$, we get that every honest party in $\mathcal{L}_1$ outputs $y$ or $\perp$. Hence, weak consistency holds. □

**Theorem 15 (Theorem 6).** *Protocol* GC *defined in Figure 3 is an* $(n, t, 2)$*-layered graded consensus protocol for any* $t < \frac{n}{3}$.

*Proof.* First, if all honest parties in $\mathcal{L}_0$ hold the same value $x$, then, as there are at least $n-t$ honest parties in $\mathcal{L}_0$, every party in $\mathcal{L}_1$ receives $x$ from at least $n-t$ parties, and at most $t$ parties in $\mathcal{L}_0$ send values different from $x$. By the protocol description, every honest party in $\mathcal{L}_1$ then forwards $x$ to everyone in $\mathcal{L}_2$. As there are at least $n-t$ honest parties in $\mathcal{L}_1$, every honest $P_i \in \mathcal{L}_2$ outputs $(x, 2)$. Thus, validity holds.

Second, note that if at least one honest party in $\mathcal{L}_1$ sends a message $x$ to all parties in $\mathcal{L}_2$, then no honest party $\mathcal{L}_1$ sends a message $x' \neq x$ to anyone in layer $\mathcal{L}_2$ (otherwise, since $(n-2t)+(n-2t)-(n-t) = n-3t > 0$, at least one honest party in $\mathcal{L}_0$ sent two different messages). **(†)**

Next, if some honest party in layer $\mathcal{L}_2$ outputs $(y, g)$ with $g \geq 1$, then it received $y$ from at least $t+1$ parties of $\mathcal{L}_1$, hence in particular at least one honest party in $\mathcal{L}_1$ sent $y$. By **(†)**, we get that there is no message $y' \neq y$ that an honest party from $\mathcal{L}_1$ sends to parties in layer $\mathcal{L}_2$. Thus, every honest party in $\mathcal{L}_2$ receives at most $t$ messages with some $y' \neq y$, hence they never output $(y', g)$ for $y' \neq y$ and $g \geq 1$.

Moreover, if some honest party in layer $\mathcal{L}_2$ outputs $(y, 2)$, then it received $y$ from at least $n-t$ parties of $\mathcal{L}_1$, hence in particular at least $n-2t$ honest parties

in $\mathcal{L}_1$ sent $y$. As $t < \frac{n}{3}$, we get $n - 2t > t$, hence every $P_i$ in layer $\mathcal{L}_2$ received $y$ from at least $t + 1$ parties. Thus, the protocol satisfies graded consistency.

□

**Theorem 16 (Theorem 7).** *Assuming a plain PKI, protocol* aWC *defined in Figure 4 is an* $(n, t, 2)$*-layered weak consensus protocol for any* $t < \frac{n}{2}$.

*Proof.* First, note that if all honest parties in $\mathcal{L}_0$ hold the same value $x$, every party in $\mathcal{L}_1$ receives correct signatures on $x$ from at least $n-t$ parties. Thus, every honest party in $\mathcal{L}_1$ sends a degree-0-certificate for $x$ to everyone in $\mathcal{L}_2$, hence every party in $\mathcal{L}_2$ receives at least $n - t$ degree-0-certificates for $x$. Furthermore, at most $t$ parties in $\mathcal{L}_0$ send values different from $x$. As $t < \frac{n}{2}$, we get $t < n - t$. Thus there cannot be a degree-0-certificate for $1 - x$. Thus, every honest party in $\mathcal{L}_2$ outputs $x$ and validity holds.

Second, if an honest party in $\mathcal{L}_2$ outputs a value $y \neq \perp$, then it received $n - t$ degree-0-certificates for $y$. As $t < n - t$, at least one of these was sent by an honest party to all parties in $\mathcal{L}_2$. Therefore, no honest party outputs $(1 - y)$, which proves weak consistency.

□

**Theorem 17 (Theorem 8).** *Assuming a plain PKI, protocol* aGC *defined in Figure 5 is an* $(n, t, 4)$*-layered graded consensus protocol for any* $t < \frac{n}{2}$.

*Proof.* We first prove validity. Assume that all honest parties in $\mathcal{L}_0$ hold the same bit $b$. Since $t < n - t$, there are not enough signatures by parties in $\mathcal{L}_0$ on $(1-b)$ to form a degree-0-certificate, which implies no honest party ever receives a certificate of any degree for $(1 - b)$. Moreover, every honest $P_i \in \mathcal{L}_1$ receives $n-t$ signatures on $b$ and sends a degree-0-certificate for $b$ to in $\mathcal{L}_2$. By analogous reasoning, all honest parties in $\mathcal{L}_2, \mathcal{L}_3$ and $\mathcal{L}_4$ receive at least $n - t$ certificates for $b$ of degree 0, 1, and 2 respectively. Thus, all honest parties in $\mathcal{L}_4$ output $(b, 2)$.

We now prove graded consistency. For the first case, assume that one honest party $P_i \in \mathcal{L}_4$ outputs $(b, 2)$. This means that $P_i$ received degree-2-certificates for $b$ from $n-t$ parties in $\mathcal{L}_3$, at least one of which was honest. This implies that all parties in $\mathcal{L}_4$ received a degree-2-certificate for $b$. Moreover, since an honest party in $\mathcal{L}_3$ sent out a degree-2-certificate for $b$, it follows that no honest party in $\mathcal{L}_2$ sent a degree-1-certificate for $(1-b)$ and thus it is impossible for any party in $\mathcal{L}_3$ to form a degree-2-certificate for $(1 - b)$. Therefore, we have that every honest party in $\mathcal{L}_4$ outputs $(b, 1)$ or $(b, 2)$.

We move on to the second case: assume some honest $P_i \in \mathcal{L}_4$ outputs $(b, 1)$. Since $P_i$ received a degree-2-certificate for $b$, there are at least $n - t$ parties in $\mathcal{L}_2$ (at least one of which is honest) who signed a degree-1-certificate for $b$. Thus, no honest party in $\mathcal{L}_1$ signed a degree-0-certificate for $(1 - b)$, which means it is impossible for any party in $\mathcal{L}_3$ to form a degree-2-certificate for $(1-b)$. Therefore, no honest party in $\mathcal{L}_4$ outputs $(1 - b, g)$ with $g \geq 1$.

□

**Proposition 2 (Proposition 1).** *Assume $t < n/2$. Let $\pi$ be an $(n, t, d)$-layered gradecast protocol and $d' > d$. Then there exists an $(n, t, d')$-layered protocol $\pi'$ which achieves gradecast towards $\mathcal{L}_d \cup \mathcal{L}_{d'}$, with graded consistency holding for any pair of honest parties $P, P' \in \mathcal{L}_d \cup \mathcal{L}_{d'}$.*

*Proof.* We construct such $\pi'$ as follows: First, parties in $\mathcal{L}_0, \ldots, \mathcal{L}_d$ execute $\pi$. Then, every party in $\mathcal{L}_d$ sends their output from $\pi$ to all parties in $\mathcal{L}_{d'}$. Parties in $\mathcal{L}_{d'}$ sort all received values using the ordering $(0,2) < (0,1) < (\bot, 0) < (1,1) < (1,2)$. Finally, parties in $\mathcal{L}_{d'}$ output the median of the resulting list.

This preserves validity: If $P_{\text{sdr}}$ is honest with input $x$, all honest parties in $\mathcal{L}_d$ will output $(x, 2)$ by validity of $\pi$. Parties in $\mathcal{L}_{d'}$ will receive $(x, 2)$ from at least $n - t > n/2$ parties in $\mathcal{L}_d$, so the median value they compute is also $(x, 2)$. Next, we show graded consistency, i.e., that the outputs of all honest parties are contained in a set of two consecutive values (w.r.t. the aforementioned ordering). Again, it follows using $t < n/2$ that if the outputs of all honest parties in $\mathcal{L}_d$ fulfil this property, then the outputs of all honest parties in $\mathcal{L}_{d'}$ are contained in the same set of two consecutive values. □

# B Proofs regarding SMT

## B.1 Extended Preliminaries

We first recap some building blocks used within $\mathsf{SMT}$. In the following, $\mathbb{F}_p$ denotes some prime field.

**Definition 12 (Shamir Sharing).** *Given any $s \in \mathbb{F}_p$, we write $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$ to denote a Shamir sharing of $s$ with degree $t$, i.e. $(s_1, \ldots, s_n)$ is obtained by sampling uniformly random $a_1, \ldots, a_t \in \mathbb{F}_p$ and computing $s_i := f(i)$, where $f(x) := s + a_1 x + \ldots + a_t x^t$.*

Note that Shamir sharing satisfies 1) $t$-privacy: Any $t$ shares reveal nothing about the secret, 2) $(t+1)$-reconstruction: From any $t+1$ shares, one can recover the secret, and 3) linearity: For any $s, s', \mu \in \mathbb{F}_p$, we have that $\mathsf{Share}(s) + \mu \cdot \mathsf{Share}(s')$ is a degree-$t$ sharing of $s + \mu s'$.

**Definition 13 (Commitments).** *We define a **non-interactive commitment scheme** with message space $\mathcal{M}$, randomness space $\mathcal{R}$, and commitment space $\mathcal{C}$ to be a tripleof PPT algorithms $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Open})$such that:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter $\lambda$ and produces a set of public parameters* $\mathsf{pp}$
- $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{pp}, m, r)$*: The commit algorithm takes as input the public parameters* $\mathsf{pp}$*, a message $m \in \mathcal{M}$, and randomness $r \in \mathcal{R}$ and produces a commitment* $\mathsf{com} \in \mathcal{C}$*;*
- $\mathsf{b} \leftarrow \mathsf{Open}(\mathsf{pp}, \mathsf{com}, m, r)$*: The open algorithm takes as input the public parameters* $\mathsf{pp}$*, a commitment* $\mathsf{com} \in \mathcal{C}$*, a message $m \in \mathcal{M}$, and randomness $r \in \mathcal{R}$ and outputs a bit* $\mathsf{b} \in \{0, 1\}$*;*

*We say that such a commitment scheme is:*

- ***Hiding*** *if for all* $m' \neq m$ *and for* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, any PPT adversary* $\mathcal{A}$ *has a negligible advantage (in the security parameter* $\lambda$*) in distinguishing between the distributions* $\mathsf{Com}(\mathsf{pp}, m, r)$ *and* $\mathsf{Com}(\mathsf{pp}, m', r)$*, where the distribution is over the randomness* $r$*;*
- ***Binding*** *if any PPT adversary* $\mathcal{A}$ *has a negligible probability (in the security parameter* $\lambda$*) of outputting* $(m, r)$ *and* $(m', r')$ *such that* $m \neq m'$ *and* $\mathsf{Com}(\mathsf{pp}, m, r) = \mathsf{Com}(\mathsf{pp}, m', r')$*;*
- ***Linearly Homomorphic*** *if there exist a binary function* $+ : \mathcal{C}^2 \to \mathcal{C}$ *and an element* $0 \in \mathcal{C}$ *such that* $(\mathcal{C}, +, 0)$ *is an abelian group, a binary function* $+ : \mathcal{M}^2 \to \mathcal{M}$ *and an element* $0 \in \mathcal{M}$ *such that* $(\mathcal{M}, +, 0)$ *is an abelian group, and a binary function* $+ : \mathcal{R}^2 \to \mathcal{R}$ *and an element* $0 \in \mathcal{R}$ *such* $(\mathcal{R}, +, 0)$ *is an abelian group, and for all* $m, m' \in \mathcal{M}$ *and all* $r, r' \in \mathcal{R}$ *it holds that* $\mathsf{Com}(\mathsf{pp}, m + m', r + r') = \mathsf{Com}(\mathsf{pp}, m, r) + \mathsf{Com}(\mathsf{pp}, m', r')$ *for all* $(\mathsf{pp}, \tau) \leftarrow \mathsf{Setup}(1^\lambda)$*;*

*In our setting we will consider* $\mathcal{M} = \mathcal{R} = \mathbb{F}_p$ *and commitment space* $\mathcal{C}$ *an* $\mathbb{F}_p$*-module* $(\mathbb{G}, +, 0)$*.*

### B.2 Proof of Theorem 13

**Proposition 3 (Correctness).** *The value output by the receiver* $P_{\text{rcv}}$*, if any, is equal to the secret* $s$ *of the sender* $P_{\text{sdr}}$*.*

**Proof of Proposition 3 (correctness)** Proposition 3 follows from applying recursively the following Lemma 1 in all the pairs of consecutive layers $(\mathcal{L}_{\mathsf{old}}, \mathcal{L}_{\mathsf{new}})$ from the sender to the receiver.

**Lemma 1.** *Consider* $C \in \mathbb{G}$ *such that the invariant* $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{old}}, C)$ *(presented in Section 9.1) holds. Then, after an execution of* Reshare *between* $\mathcal{L}_{\mathsf{old}}$ *and* $\mathcal{L}_{\mathsf{new}}$*,* $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{new}}, C)$ *also holds if at most* $t < n/2$ *parties in* $\mathcal{L}_{\mathsf{old}}$ *are corrupt.*

*Proof.* Assume $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{old}}, C)$ holds. Since a majority of parties in $\mathcal{L}_{\mathsf{old}}$ are honest, all honest parties in $\mathcal{L}_{\mathsf{new}}$ find $C$ to be the most received commitment. Additionally, the check performed by honest parties in $\mathcal{L}_{\mathsf{new}}$ during steps 2(a)iii and 2(a)iv in protocol Reshare ensures (by linearity of Shamir sharings) that for all tuples $(\ell, \mathbf{C}^\ell, \cdot, \cdot) \neq \perp$ stored by honest parties in $\mathcal{L}_{\mathsf{new}}$ it holds that $C, \mathbf{C}$ lie on a degree-$t$ polynomial.

Thus, $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{new}}, C)$ holds. □

**Proposition 4 (Liveness).** *The receiver* $P_{\text{rcv}} \in \mathcal{L}_d$ *outputs a value at the end of the* $d$*-th round.*

**Proof of Proposition 4 (Liveness).** Proposition 4 follows from applying the following Lemma 2 in all the pairs of consecutive layers $(\mathcal{L}_{\mathsf{old}}, \mathcal{L}_{\mathsf{new}})$ from the sender to the receiver. In the proofs of Proposition 4 below, we consider as a unique public fixed data the commitment $C$ to the secret which is passed unchanged from layer to layer (see Lemma 1).

**Lemma 2.** *Assume that* $\mathsf{validSharing}(\mathcal{L}_{\mathsf{old}})$ *(presented in Section 9.1) holds, that there exists a commitment* $C$ *such that* $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{old}}, C)$ *holds and that at most* $t < n/2$ *parties in* $\mathcal{L}_{\mathsf{old}}$ *are corrupt. Then, after executing* $\mathsf{Reshare}$ *between* $\mathcal{L}_{\mathsf{old}}$ *and* $\mathcal{L}_{\mathsf{new}}$*, we have that* $\mathsf{validSharing}(\mathcal{L}_{\mathsf{new}})$ *holds.*

*Proof.* Fix $I \in \mathcal{Q}$ such that all $P_i \in I \subseteq \mathcal{L}_{\mathsf{old}}$ are honest. By $\mathsf{validSharing}(\mathcal{L}_{\mathsf{old}})$, there exists a label $\ell$ and $\mathbf{C} \in \mathbb{G}^n$ such that all $P_i \in I$ have stored a tuple starting with $(\ell, \mathbf{C})$ in their list. By $\mathsf{consistentCommitments}(\mathcal{L}_{\mathsf{old}}, C)$ we have that $C, \mathbf{C}$ lie on a degree-$t$ polynomial. This implies that, when iterating over the set $I$, every honest $P_j \in \mathcal{L}_{\mathsf{new}}$ will store a tuple, as all the checks in step a will be fulfilled for $\ell$.

Moreover, as honest parties in $\mathcal{L}_{\mathsf{old}}$ send a subsharing for any given $\ell$ to either all or no parties in $\mathcal{L}_{\mathsf{new}}$, and in the former case send the same vector of commitments to subshares (with one correct opening), it follows that any $\ell$ either passes the check for all or for no honest $P_j \in \mathcal{L}_{\mathsf{new}}$ iterating over $I$.

Thus, all honest $P_j \in \mathcal{L}_{\mathsf{new}}$ pick the same maximum label $\ell^*$ for $I$, and they compute the same new commitment vector $\mathbf{C}^{\mathsf{new},I}$. It then follows from the fact that checks 2(a)iii and 2(a)iv passed that $C, \mathbf{C}^{\mathsf{new},I}$ lie on a degree-$t$ polynomial (using linearity of Shamir sharings); and it follows from check 2(a)ii (using linearity of commitments) that any honest $P_j \in \mathcal{L}_{\mathsf{new}}$ will obtain an opening to $\mathbf{C}^{\mathsf{new},I}[j]$.

Therefore, $\mathsf{validSharing}(\mathcal{L}_{\mathsf{new}})$ holds. □

**Proof of Theorem 13.** We now show the claim of Theorem 13, which is that for any PPT adversary $\mathcal{A}$ in the real world corrupting $t < n/2$ parties, there exists a simulator Sim in an ideal world (where parties have access to an ideal functionality $\mathcal{F}_{\mathrm{smt}}$) such that the joint distribution of the adversary's view and the honest parties' inputs and outputs is statistically indistinguishable between the two worlds.

*Proof.* We describe the simulator Sim for an execution with an honest sender and an honest receiver (we make no guarantees if the sender or receiver is corrupted). The simulator Sim will simulate the view of the adversary $\mathcal{A}$ in a real protocol execution by interacting solely with the adversary (including the corrupt parties).

*Simulating the sender* $P_{\mathrm{sdr}}$. Denote $\mathrm{F} \subset [n]$ and $\mathrm{H} \subset [n]$ the subsets of indices of the $t$ corrupt parties and of the $n-t$ honest parties in $\mathcal{L}_1$. Generate a random commitment $C$. For each index $i \in \mathrm{F}$ of a corrupt party $P_i \in \mathcal{L}_1$, generate uniformly at random $(s_i, r_i)$, set $C_i = \mathsf{Com}(s_i, r_i)$. Interpolate the remaining $(C_i)_{i \in \mathrm{H}}$ through the $t+1$ points $\{0\} \cup \mathrm{F}$ and corresponding values $C, (C_i)_{i \in \mathrm{F}}$,

so that the full vector $(C_i)_{i \in \mathrm{H}}$ is a vector of evaluations of a degree-$t$ polynomial with coefficients in $\mathbb{G}$, i.e., in $\mathbb{G}[X]$, which evaluates in particular to $C$ at 0. Make $P_{\mathrm{sdr}}$ send to each corrupted party $P_i \in \mathcal{L}_1$ $(i \in \mathrm{F})$ the quintuple

$$(C, (C_i)_{i \in [n]}, s_i, r_i)$$

and to each simulated honest party $P_i \in \mathcal{L}_1$ $(i \in \mathrm{H})$ the quintuple

$$\big(C, (C_i)_{i \in [n]}, \perp, \perp\big)$$

*Simulating the honest parties of an old layer $\mathcal{L}_{\mathsf{old}}$ during a* Reshare *to the next layer $\mathcal{L}_{\mathsf{new}}$.* Denote $\mathrm{F}' \subset [n]$ and $\mathrm{H}' \subset [n]$ the subsets of indices of the $t$ corrupt parties and of the $n-t$ honest parties in $\mathcal{L}_{\mathsf{new}}$. For each (simulated) old honest party $P_i$ and for each $\ell \in \mathcal{Q}$ for which $P_i$ stored a tuple $(\ell, \mathbf{C}_i^\ell, s_i^\ell, r_i^\ell) \neq \perp$, generate uniformly at random $(s_{i,j}^\ell, r_{i,j}^\ell)$ for every $j \in \mathrm{F}'$ and set $C_{ij}^\ell = \mathsf{Com}(s_{i,j}^\ell, r_{i,j}^\ell)$. Interpolate the remaining $(C_{ij}^\ell)_{j \in \mathrm{H}'}$ through the $t+1$ points $\{0\} \cup \mathrm{F}'$ and corresponding values $\mathbf{C}_i^\ell[j], (C_{ij}^\ell)_{j \in \mathrm{F}'}$, so that the full vector $(C_{ij}^\ell)_{j \in [n]}$ is a vector of evaluations of a degree-$t$ polynomial with coefficients in $\mathbb{G}$, i.e., in $\mathbb{G}[X]$, which evaluates in particular to $\mathbf{C}_i^\ell[j]$ at 0. Make $P_i$ send $C$ to each corrupted party $P'_j \in \mathcal{L}_{\mathsf{new}}$ $(j \in \mathrm{F}')$ as well as the tuples

$$\big(\ell, (C_{ij}^\ell)_{j \in [n]}, s_{i,j}^\ell, r_{i,j}^\ell\big)$$

for every $\ell \in \mathcal{Q}$. Make $P_i$ send to each simulated honest party $P'_j \in \mathcal{L}_{\mathsf{new}}$ $(i \in \mathrm{H}')$ the quintuples

$$\big(\ell, (C_{ij}^\ell)_{j \in [n]}, \perp, \perp\big)$$

for every $\ell \in \mathcal{Q}$.

*Simulating (silently) the honest parties of a "new" layer $\mathcal{L}_{\mathsf{new}}$ during a* Reshare *from the "old" previous layer $\mathcal{L}_{\mathsf{old}}$.* Each $P'_j \in \mathcal{L}_{\mathsf{new}}$ follows the protocol up to two changes: The first change is that for each $\mathrm{I} \in \mathcal{Q}$, the check in step 2(a)ii (fig. 13) is only performed for corrupt $P_i \in \mathcal{L}_{\mathsf{old}}$, as simulated honest parties don't send any subshares, and the second change is that instead of computing new shares in step c, simulated honest parties set $(s_j^{\mathsf{new},I}, r_j^{\mathsf{new},I}) \leftarrow (\perp, \perp)$.

***Proof of indistinguishability with a real execution.*** We consider a series of hybrid games going from the view in the real execution, to the view in the simulated one.

We first define the games $\mathbf{Game}^{\mathrm{strip},\mathbf{r}}$ for $r \in [d]$ as follows: $\mathrm{Game}^{\mathrm{strip},\mathbf{d}}$ is the real execution. Then, for $r \in [d-1]$: $\mathrm{Game}^{\mathrm{strip},\mathbf{r}}$ is defined recursively as $\mathrm{Game}^{\mathrm{strip},\mathbf{r+1}}$, with the change that in the Reshare between $\mathcal{L}_{r-1}$[11] and $\mathcal{L}_r$ the new parties $P'_j \in \mathcal{L}_{\mathsf{new}}$ behave (silently) as in the first change (1) of the simulated execution. Namely, for each I, they only apply the check in step 2(a)ii to tuples sent by corrupt parties. Since the honest parties only send correct data, it follows

[11] by $\mathcal{L}_0$ we mean simply $P_{\mathrm{sdr}}$

that removing the check produces the same outcome. Hence, the view of the adversary is unchanged.

We then define the game $\mathbf{Game^{rcv}}$ as follows. $\text{Game}^{\mathbf{rcv}}$ is defined as the real execution, with the difference that the opening is simulated as done by Sim. Since the number of 5-uples sent by each simulated $P_i \in \mathcal{L}_{d-1}$ to $P_{\text{rcv}}$, as well as their formats, are the same as if it had been playing honestly the protocol, the view of the adversary is unchanged.

We then define the cascade of $(n+1)(d-2)$ games:

$$(\mathbf{Game^{r,i}})_{r \in [d-2], i \in [n]} \tag{1}$$

for $i = 0, \ldots, n$, then for $r = d-2, \ldots, 1$ in backwards order. We set $\text{Game}^{\mathbf{d-1,0}} = \text{Game}^{\mathbf{rcv}}$ and $\text{Game}^{\mathbf{r,0}} = \text{Game}^{\mathbf{r+1,n}}$ $\forall r = 2, \ldots, d-1$. We then define recursively each $\text{Game}^{\mathbf{r,i}}$, for $i \geqslant 1$, as equal to $\text{Game}^{\mathbf{r,i-1}}$, with the change that $P_i$ now behaves, as an old committee member, in the same way as simulated by Sim. First, since the modified checks pass if and only if the actual checks pass, the simulated honest parties send messages with the same volume as the ones sent in $\text{Game}^{\mathbf{r,i}}$. As regards the contents of the messages received by the corrupt parties, it follows from the $t$-privacy property of the Shamir Secret Sharing that they are equally distributed in $\text{Game}^{\mathbf{r,i}}$ as in $\text{Game}^{\mathbf{r,i-1}}$. In conclusion, the view of the adversary in $\text{Game}^{\mathbf{r,i}}$ is the same as in $\text{Game}^{\mathbf{r,i-1}}$.

We then define the last game $\mathbf{Game^{sdr}}$, which is obtained from $\text{Game}^{\mathbf{1,n}}$ by making the simulated sender $P_{\text{sdr}}$ follow the same behavior as the one simulated by Sim. Hence, we are now arrived at the simulated execution. Indistinguishability from $\text{Game}^{\mathbf{1,n}}$ again follows from the $t$-privacy property of the Shamir Secret Sharing.

□

## C Broadcast Functionality and Construction

We define the functionality of $\mathcal{F}_{\mathsf{bc}}$ below, which is also used in prior papers for MPC with dynamic committees.

**Broadcast Channel $\mathcal{F}_{\mathsf{bc}}$**

**Public Parameters.** Senders $P_{\text{sdr}} \in \mathcal{L}_0$, receiver layer $\mathcal{L}_k$ where $k \geq 1$. The domain $M$ of the message. Set $m = \perp$.

**Layer $\mathcal{L}_0$:**

- The functionality receives a message $x \in M$ from $P_{\text{sdr}} \in \mathcal{L}_0$ (where a corrupted sender can input the message in a rushing fashion, after the honest parties sent their messages of this round), towards all parties in $\mathcal{L}_k$. Set $m = x$.
- Output $m$ to the adversary.

**Layer $\mathcal{L}_k$:**

– If the sender $P_{\text{sdr}}$ is corrupted, receive $m'$ from the (ideal) adversary. Set $m = m'$.
– Output $m$ to all parties in $\mathcal{L}_k$.

### C.1 Protocol Description.

Consider the straightforward protocol $\Pi_{\mathsf{bc}}$ that simply lets the $P_{\text{sdr}} \in \mathcal{L}_0$ send its message $x$ to all parties in $\mathcal{L}_1$. Then, each $P_1 \in \mathcal{L}_1$ inputs $x$ to a layered consensus protocol $\Pi_{\mathsf{consensus}}$ taking $d = O(\lambda)$ layers (we choose the one in corollary 12). Finally, parties in $\mathcal{L}_{d+1}$ output what they receive from the consensus protocol.

**Theorem 18.** *There exist layered broadcast protocols with (1) statistical security for $t < n/3$ and no setup assumptions and (2) cryptographic security for $t < n/2$ assuming a plain PKI.*

*Proof.* (Sketch)

First note that the outputs of honest parties are the same in both real and ideal worlds. If the sender $P_{\text{sdr}} \in \mathcal{L}_0$ is honest with input $x$, the broadcast functionality outputs $x$ in the ideal world. In the real world, all parties in layer $\mathcal{L}_1$ obtain the same input $x$, and by validity of $\Pi_{\mathsf{consensus}}$, all honest parties output $x$. In case the sender $P_{\text{sdr}} \in \mathcal{L}_0$ is corrupted, $\Pi_{\mathsf{consensus}}$ guarantees that all honest parties output the same value $x'$ at the end of the protocol. The simulator simply emulates all the steps of the protocol $\Pi_{\mathsf{bc}}$ on behalf of the honest parties, obtains the output $x'$ of $\Pi_{\mathsf{consensus}}$ and inputs it to $\mathcal{F}_{\mathsf{bc}}$. The functionality then guarantees that honest parties also obtain the output $x'$ in the ideal world.

The simulation of the adversary's view is straightforward. Consider an honest sender $P_{\text{sdr}} \in \mathcal{L}_0$. The simulator obtains the message $x$ and simulates the whole protocol $\Pi_{\mathsf{bc}}$ on behalf of the honest parties. For the case of a corrupted sender, the simulator again simulates the whole protocol $\Pi_{\mathsf{bc}}$ on behalf of the honest parties. Note that since there is no secret information, the simulator can exactly emulate the adversary's view in the real world.