$\tilde{\text{O}}$ptimal Parallel Broadcast*

Gilad Asharov
Gilad.Asharov@biu.ac.il

Anirudh Chandramouli
Anirudh.Chandramouli@biu.ac.il

Department of Computer Science
Bar-Ilan University

February 29, 2024

Abstract

We study broadcast protocols in the information-theoretic model under optimal conditions, where the number of corruptions t is at most one-third of the parties, n. While worst-case $\Omega(n)$ round broadcast protocols are known to be impossible to achieve, protocols with an expected constant number of rounds have been demonstrated since the seminal work of Feldman and Micali [STOC'88]. Communication complexity for such protocols has gradually improved over the years, reaching $O(nL)$ plus expected $O(n^4 \log n)$ for broadcasting a message of size L bits.

This paper presents a perfectly secure broadcast protocol with expected constant rounds and communication complexity of $O(nL)$ plus expected $O(n^3 \log^2 n)$ bits. In addition, we consider the problem of parallel broadcast, where n senders, each wish to broadcast a message of size L. We show a parallel broadcast protocol with expected constant rounds and communication complexity of $O(n^2 L)$ plus expected $O(n^3 \log^2 n)$ bits. Moreover, we show a lower bound for parallel broadcast, showing that our protocol is optimal up to logarithmic factors and in expectation.

Our main contribution is a framework for obtaining *perfectly* secure broadcast with an expected constant number of rounds from a *statistically* secure verifiable secret sharing. Moreover, we provide a new statistically secure verifiable secret sharing where the broadcast cost per participant is reduced from $O(n \log n)$ bits to only $O(\mathsf{poly} \log n)$ bits. All our protocols are adaptively secure.

Keywords: Perfect Secure Computation, Broadcast, Byzantine Agreement, Verifiable Secret Sharing

*This research is sponsored by the Israel Science Foundation (grant No. 2439/20). Asharov is additionally sponsored by JPM Faculty Research Award, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

Contents

1 Introduction

Broadcast is a fundamental building block in secure computation, serving as a crucial primitive. It enables a designated party (a sender) to transmit a message while ensuring that all participants receive an identical message and reach a consensus on its content. However, this task becomes particularly challenging when dealing with potentially corrupted parties, as they may deceive others about the messages they have received, or a corrupted sender may transmit inconsistent messages.

The primary focus of this paper is to address the realization of the broadcast primitive over point-to-point (ideal private) channels, in the most demanding scenario: achieving perfect security with optimal resilience. Perfect security means that the protocol cannot rely on any computational assumption and that the error probability is zero. Optimal resilience means that the number of corrupted parties is at most $t < n/3$, which is tight by the lower bounds of [LSP82, PSL80].

For broadcasting a message of size L, the best one can hope for is $O(nL)$; since each party has to receive a message of size L, then the total bits transmitted is at least nL. Broadly, broadcast protocols can be characterized into two categories:

- **Succinct protocols but with a high number of rounds:** In this family, broadcasting a message of size L takes $O(nL + n^2 \log n)$ total communication complexity with $\Theta(n)$ rounds [CW89,BGP92,Che21]. Broadcasting a single bit requires $\Omega(n^2)$ bits of communication, when the protocol is deterministic [DR82], or randomized, against strongly-adaptive adversaries[1] [ACD+23].
- **More communication, but with an expected constant number of rounds:** It has been shown that for any broadcast protocol with perfect security, there exists an execution that requires $t + 1$ rounds [FL82]. This implies that a protocol with a strict constant number of rounds is impossible to achieve, and this limitation can be overcome by using randomization [Rab83, Ben83]. This family of protocols, originated by Feldman and Micali [FM88] followed by the impressive improvement of Katz and Koo [KK06], results in $O(n^2 L)$ bits plus expected $O(n^6 \log n)$ bits in expectation for broadcasting a message of size L, with an expected constant number of rounds. This result was recently improved by Abraham et al. [AAPP22], which requires communication complexity of $O(nL)$ bits plus expected $O(n^4 \log n)$ bits, with an expected constant number of rounds.

Since broadcast is such a fundamental primitive, narrowing the gap between these two families of protocols is of high importance. This has a potential impact on the complexity of secure computation protocols.

Parallel broadcast. In secure protocols, we often witness the communication pattern in which n parties have to broadcast messages in parallel at the same round. For instance, in verifiable secret sharing, the parties often complain (in parallel) about messages sent by the dealer, or vote (in parallel) whether to accept or reject the shares of the dealer. Since each party has to receive $O(nL)$ bits in total, the best we can hope for is $O(n^2 L)$ communication complexity, with an expected constant number of rounds, and in that case, we say that the protocol is "asymptotically-free". The protocol of [KK06] shows an asymptotically-free parallel broadcast for messages of size $\approx n^4$, while [AAPP22] is asymptotically-free for messages of size $\approx n^2$. Explicitly, the former is a parallel broadcast with expected $O(n^2 L + n^6 \log n)$ communication complexity and the latter is $O(n^2 L + n^4 \log n)$.

1.1 Our Results

Our main result is an asymptotically-free parallel broadcast for messages of size $\approx n$. Additionally, we give a lower bound showing that our result is *optimal* up to logarithmic factors and expectation for messages of size $\approx n$. We show:

[1] Strongly adaptive adversary means that at the same round, the adversary can see the message sent by an honest party, corrupt it, and remove (and switch) the other messages being sent at the same round by that party.

Theorem 1.1. *There exists a perfectly secure, balanced, parallel-broadcast protocol with optimal resilience which allows n senders to distribute each a message of size L, at the communication cost of $O(n^2L)$ bits plus expected $O(n^3 \log^2 n)$ bits, and expected constant number of rounds.*

By "balanced" we mean that each party sends or receives roughly the same amount of information, and so each party sends or receives $O(nL)$ bits plus expected $O(n^2 \log^2 n)$ bits in our protocol. On the other hand, we show a simple lower bound:

Claim 1.2. *Any parallel broadcast protocol where each participant broadcasts $L \geq n$ bits requires $\Omega(n^2L + n^3)$ bits communication.*

In addition, strict constant round parallel broadcast is impossible [FL82], and so our protocol is optimal up to logarithmic factor and expectation for $L \geq n$.

Giving a coarse analysis, and just evaluating the terms in the asymptotic notation, for 1024 parties, even if each party has to broadcast just a single bit (e.g., voting whether to accept the shares of the dealer), in the protocol of [AAPP22] each party has to send or receive about 1.34GB whereas in our protocol it has to send or receive about 13MB.

Ordinary broadcast. In classical broadcast, where there is a single sender, we provide an improvement of almost a factor of n in the communication complexity of broadcast with perfect security and optimal resilience:

Theorem 1.3 (Informal)**.** *There exists a perfectly secure, balanced, broadcast protocol with optimal resilience that requires communication of $O(nL)$ plus expected $O(n^3 \log^2 n)$ bits for broadcasting a message of size L bits, with an expected constant number of rounds.*

That is, our protocol falls in the second category of protocols where the best previous result is that of Abraham et al. [AAPP22], which requires $O(nL)$ bits plus expected $O(n^4 \log n)$ bits. As such, our protocol is asymptotically-free also for much shorter messages.

The protocol of Abraham et al. [AAPP22] is asymptotically free for messages $L \in \Omega(n^3 \log n)$. Our protocol is asymptotically free for messages of size $L \in \Omega(n^2 \log^2 n)$. E.g., ignoring constant factors and just evaluating the terms in the asymptotic notation, for broadcasting 1KB and 512 parties, each party in [AAPP22] has to send or receive ≈144MB; ours is ≈2.65MB.

Main Technical Contributions

Our result is obtained from the following technical contributions.

Efficient oblivious leader election (OLE). The broadcast protocols in [FM88, KK06, AAPP22] rely on a primitive called oblivious leader election (OLE). The goal in OLE is that all parties would randomly pick one of them as a leader, and reach an agreement on the identity of that leader. The desired outcome is that the chosen leader would be one of the honest parties. The protocol might fail with some constant probability (either, a non-honest leader is chosen, or there is no agreement on who is the chosen leader). We show:

Theorem 1.4. *There exists a perfectly secure, oblivious leader election, with total communication of $O(n^3 \log^2 n)$ bits over point-to-point channels and strict constant number of rounds.*

The oblivious leader election of [KK06] requires $O(n^6 \log n)$ bits over point-to-point channels; the leader election of [AAPP22] requires $O(n^4 \log n)$ cost over point-to-point channels. Our conceptual contribution is that, at least at the intuitive level, the embedded error in the functionality of OLE allows it to be implemented from weaker primitives. Our improved OLE is achieved via two improvements:

- **OLE from Statistical VSS:** OLE uses verifiable secret sharing as a sub-protocol. Since the oblivious leader election may fail regardless with some constant probability, it becomes possible to build it using a statistically secure VSS as a sub-protocol rather than a perfectly secure one. Essentially, the negligible security error introduced to the statistical VSS is shifted into the constant failure probability of the oblivious leader election,

resulting in negligible degradation in the round complexity (which still remains constant in expectation).

- **Less VSSes:** The oblivious leader election in [KK06, AAPP22] requires all parties to run VSS in parallel, while each party has to share $O(n)$ secrets. Our OLE requires each party to share only $O(\mathsf{poly}\log n)$ secrets. This introduces an error, which is again shifted into the constant failure probability of the OLE.

Those improvements open the door to significantly reducing the communication complexity.

Broadcast efficient statistically-secure verifiable secret sharing. The protocol of [AAPP23] has low overhead per secret – but only for a relatively high number of secrets. Even if we use an ideal broadcast (that is impossible to achieve – i.e., $O(nL)$ and constant round) in the protocol of [AAPP23], we get $O(n^3 \log n)$ total communication for sharing $O(n^2)$ secrets, i.e., overhead of $O(n)$ per secret. But if one wants to share only, say $O(\log n)$ secrets, the cost, even when using an ideal broadcast, still remains $O(n^3 \log n)$, i.e., overhead of $O(n^3)$ per secret. The same is true also for other VSS protocols such as [BGW88, Fel88, AAPP22] in the perfect setting and in the statistical setting [PCR08]. We show a protocol that has low overhead also for small number of secrets.

Theorem 1.5. *There exists a verifiable secret sharing protocol that allows a dealer to distribute m secrets of $O(\log n)$ bits each, with a total communication complexity of $O(m \cdot n^2 \log n)$ bits over the point-to-point channels; the dealer broadcasts $O(n \log n)$ bits, and each party broadcasts $O((m + \log(n/\epsilon)) \cdot \log n)$ bits. The protocol has an error probability of ϵ in case of a corrupted dealer. The protocol tolerates up to $t < n/3$ malicious parties and is adaptively secure.*

Putting it differently, using an ideal broadcast, our protocol achieves a lower cost than [AAPP23] when the number of secrets $m \in o(n)$. Specifically, for our oblivious leader election, each one of the n parties broadcasts $m = \log n$ secrets. Using the VSS of [AAPP23] this results in $O(n^4 \log n)$ bits in total (for all n dealers). Using our VSS, this results in $O(n^3 \log^2 n)$, which leads to the cost of the OLE mentioned in Theorem 1.4. We achieve this improvement by letting all parties broadcast together $\tilde{O}(n)$ bits over the broadcast channel, instead of $\tilde{\Omega}(n^2)$ by previous works.

For our OLE, it suffices to have VSS that fails with probability $\epsilon = 1/\mathsf{poly}(n)$. We also use $m = \log n$, and thus we get that each participant broadcasts $O((m+\log(n/\epsilon)) \cdot \log n = O(\log^2 n)$ bits. Theorem 1.3 is reported in that light.

An important challenge that we overcome is that the protocol is *adaptively* secure (even strongly-adaptively secure). Existing statistically-secure VSS protocols [Rab94, CDD+99, PCRR09, Kum12] are not suitable for our needs due to either not being adaptively secure (see [CDD+99]) or not incurring the above costs. Furthermore, several common techniques for achieving statistical security, such as dynamically electing a *small* (for instance, of size $o(n)$) committee are also not acceptable, as an adaptive adversary can corrupt the elected committee dynamically.

Discussions

Applications. Protocols in the MPC literature are usually given in the broadcast-hybrid model, and improving broadcast automatically improves the complexity of protocols or other building blocks. For example, the VSS protocol of [AAPP23], when using the broadcast protocol of [AAPP22], results in $O(m \cdot n \log n + n^4 \log n)$ for sharing m secrets. In particular, it means that if one has to share $m = n^2 \log^2 n$ secrets, the protocol runs in $O(n^4 \log n)$, i.e., an overhead of $\approx O(n^2)$ per secret. The same protocol of [AAPP23], using our broadcast, results in $O(m \cdot n \log n + n^3 \log^2 n)$ bits for sharing m secrets, that is, for the same m as before it runs in $O(n^3 \log^2 n)$, i.e., an overhead of $O(n)$ per secret.

Worst-case number of rounds. Broadcast protocols [FM88, KK06, AAPP22] in expected constant number of rounds work by repeating some randomized process (oblivious leader election) that succeeds with a constant probability. If not succeeded, there is a repetition. This

leads to a protocol that might never terminate. Such protocols can be transformed into protocols in strict polynomial time, using the approach of Goldreich and Petrank [GP90]. E.g., after $O(\log n)$ unsuccessful iterations, the parties can run the $O(n)$ rounds protocol for broadcast with guaranteed termination. This results in a protocol that is perfectly secure and runs in an expected constant number of rounds, and $O(n)$ rounds in the worst case.

Adaptive security. It is well known that any perfectly secure protocol that is secure against a static adversary, (and in addition satisfies some natural properties) is also adaptively secure [CDD+01]. Therefore, at first sight, it seems unclear why in Theorem 1.5 we require that the statistically-secure protocol would be adaptively secure. We remark that the approach of [CDD+01] does not necessarily preserves the communication complexity. Moreover, we obtain perfect security (and thus also adaptive security) only after following the approach of Goldreich and Petrank [GP90]. For instance, assume an OLE which is not adaptively secure, and thus a corrupted adversary can make the OLE to always fail (we stress that this is not the case in our protocol, but we give it just as an example). As a result, after $O(n)$ unsuccessful iterations, the parties would always have to run the $O(n)$ rounds protocol with guaranteed termination. The resulting protocol is perfectly secure with an expected $O(1)$ rounds against a static adversary, but with $\Theta(n)$ rounds against an adaptive adversary. Using adaptively-secure VSS guarantees that we successfully terminate after (expected) $O(1)$ rounds even when the adversary is adaptive. This is why in all statistically-secure sub-protocols that we have, we explicitly require adaptive security.

Simultaneous termination. Any broadcast protocol with $o(t)$ expected rounds cannot guarantee simultaneous termination. This raises a difficulty when sequentially composing such protocols – parties are not necessarily synchronized. This issue was addressed in Lindell, Lysyanskaya and Rabin [LLR02], Katz and Koo [KK06] and Cohen, Coretti, Garay and Zikas [CCGZ19], and we refer the reader there for further details.

We use a standalone, simulation-based definition as in [Can00]. This definition does not capture rounds in the ideal functionality or that there is no simultaneous termination. The work of [CCGZ19] shows that one can compile a protocol using deterministic termination hybrids (as the ideal functionality that we provide) into a protocol that uses an expected constant number of rounds protocol for emulating those hybrids.

1.2 Related Work

Broadcast is an essential primitive and was studied extensively over the years. It is known that perfect byzantine agreement and broadcast are impossible for $t \geq n/3$ [LSP82, PSL80]. Fischer and Lynch [FL82] showed that in any deterministic broadcast, there exists an execution that takes at least $t+1$ rounds. Rabin [Rab83] and Ben-Or [Ben83] studied the effect of randomization on round complexity, and Feldman and Micali [FM88] gave the first protocol with an expected constant round protocol for byzantine agreement with optimal resilience. We report the progress of broadcast protocols over the years in Table 1.

Gradecast was introduced by Feldman and Micali [FM88], and was improved over the years (e.g., [AAPP22, AA22]). Gradecast is used as a building block for multiple consensus algorithms, e.g., multi-consensus, approximate agreement [BDH10], or for Phase-King [ALP22].

Verifiable secret sharing was introduced by Chor et al. [CGMA85]. The first perfectly secure verifiable secret sharing was presented by Ben Or, Goldwasser, and Wigderson [BGW88] and by Feldman [Fel88]. Abraham et al. [AAPP22] showed how to distribute $O(n)$ secrets at the same cost as one VSS invocation, which led to reduce the cost of Broadcast as well. In another work, Abraham et al. [AAPP23] showed how to batch many instances of VSS together while keeping the same broadcast cost among all instances as just a single instance, to reduce the cost of MPC protocols in the perfect settings. The protocol of [AAPP23] serves as our starting point for the VSS protocol, as it reduces the broadcast cost of the dealer from $O(n^2 \log n)$ to $O(n \log n)$.

Our lower bound holds for parallel broadcast of a message of size L bits, where $L \geq n$. Interestingly, the parallel broadcast protocol of [TLP22] for $L = 1$, incurs a total communication

Protocol	Total Communication	Rounds
	Broadcast	
Coan et al. [CW89], Berman et al. [BGP92]	$O(n^2L)$	$O(n)$
[CW89], [BGP92]+Chen [Che21]	$O(nL + n^2 \log n)$	
Katz and Koo [KK06]	$O(n^2L) + E(O(n^6 \log n))$	Expected Constant
[KK06] + Nayak et al. [NRS+20]	$O(nL) + E(O(n^7 \log n))$	
Abraham et al. [AAPP22]	$O(nL) + E(O(n^4 \log n))$	
Our Work	$O(nL) + E(O(n^3 \log^2 n))$	
	Parallel Broadcast	
Chen [Che21]	$O(n^2L + n^3 \log n)$	$O(n)$
Katz and Koo [KK06]	$O(n^2L) + E(O(n^6 \log n))$	Expected Constant
Abraham et al. [AAPP22]	$O(n^2L) + E(O(n^4 \log n))$	
Our Work	$O(n^2L) + E(O(n^3 \log^2 n))$	
Our lower bound:	$\Omega(n^2L + n^3)$ for $L \geq n$	

Table 1: The complexity of our protocols and comparison to previous works of broadcast and parallel broadcast. Total communication is given in bits.

$O(n^2\kappa^4)$ assuming a trusted PKI, where κ is the security parameter.

2 Technical Overview

The construction of broadcast in an expected constant number of rounds is quite complex and consists of several different building blocks. We provide a glossary for the different primitives in Appendix A, which can be used as a reference when reading other parts of the paper. In the following, we overview our contributions in two primitives, each can be studied and analyzed independently and regardless to broadcast. We overview our oblivious leader election (Section 2.1) and our statistically secure VSS (Section 2.2). We provide some conclusions in Section 2.3.

2.1 Efficient Oblivious Leader Election

Oblivious leader election is a protocol where the parties try to elect one of them as a leader, uniformly at random. Each party has no input, and each party outputs an index in $\{1, \ldots, n\}$. We might have three possible outcomes: (1) All parties output the same index $j \in \{1, \ldots, n\}$, and P_j is honest; (2) All parties agree on the same index $i \in \{1, \ldots, n\}$, but P_i is corrupted; (3) There is no agreement on the output index. The goal is to achieve outcome (1) with some constant probability (and a strict constant number of rounds). Once outcome (1) occurs, then the broadcast protocol terminates successfully. If (2) or (3) occur, then we re-run OLE, leading to an overall broadcast protocol that runs in expected constant number of rounds.

The main idea in [FM88, KK06, AAPP22] to elect a leader is to pick a random value c_j for each one of the parties P_j. The value c_j is chosen collectively by all parties. Then, the elected leader is the one for which c_j is minimal. Towards that end, in [FM88, KK06, AAPP22] each party P_i chooses n random values uniformly at random, $c_{i\to 1}, \ldots, c_{i\to n}$, where each $c_{i\to j}$ should be interpreted as the contribution of P_i to P_j. The parties define the value $c_j = \sum_{i=1}^{n} c_{i\to j}$ to be the random value associated with P_j. This guarantees that even if some corrupted parties contribute values that are not uniformly random, each c_j is still random.

To make this idea work and prevent the corrupted parties from biasing those random values, we need to implement a mechanism that achieves hiding and binding, like a commitment scheme. A mechanism that allows exactly that in the information-theoretic setting is a verifiable secret

sharing. Each party P_i verifiably secret shares $c_{i\to 1}, \ldots, c_{i\to n}$. Then, after all parties share their values, all parties reconstruct all secrets. Hiding guarantees that the shares do not provide any information about the secrets, which means that the adversary must choose its contributions independently of the contributions of the honest parties. Binding guarantees that once the sharing phase is concluded, the dealer cannot change its decision, and reconstruction is always guaranteed. Therefore, the adversary cannot bias the result by either opening different values than what it initially committed to, or selectively failing particular reconstructions.

Our OLE. Our conceptual contribution is that, at least at the intuitive level, the embedded error in the functionality of OLE allows us to use statistically secure building blocks to realize OLE. Specifically, all we care about is that outcome (1) occurs with a constant probability. Therefore, we can relax the requirements from the protocol, and achieve a more efficient construction.

Reducing the amount of secrets. When designing a perfectly secure protocol, we have to guarantee that for each party P_i, there is at least one *honest* party P_j that contributed to the value c_i. Since the number of corrupted parties might be up to $n/3$, it implies that each party must receive at least $n/3+1$ contributions, i.e., we have $O(n^2)$ secrets in total that have to be shared. This is why each party just contributes to all other parties in the protocols of [FM88, KK06, AAPP22]. However, when designing a statistically secure protocol, it suffices that *with high enough probability*, for each party P_i there is at least one *honest* party P_j that contributed $c_{j\to i}$ to the value c_i. This guarantees that c_i is uniform.

Towards that end, instead of each P_j picking n random values, we instruct it to just pick $O(\mathsf{poly}\log n)$ random values, together with $O(\mathsf{poly}\log n)$ random parties that it contributes to. The identities of which parties P_j contributes to are secret shared as well to guarantee that the adversary cannot pick which parties to contribute to after seeing the choices made by the honest parties. We show that this simple mechanism suffices to guarantee that with high probability, all parties have at least one honest party that contributed to their value, and therefore, all values $c_1, \ldots, c_n$ are random.

Moderated VSS. The above description is oversimplified. Specifically, one problem that arises is that the VSS itself uses broadcast as a primitive while we use OLE to implement broadcast. To avoid this circularity, the broadcast inside the VSS is replaced with a weaker primitive called gradecast. Gradecast (see the glossary in Appendix A) is a relaxation of broadcast where the sender sends a message M to all parties, and each party P_i outputs some message together with a grade. The guarantee is that if the sender is honest, then all honest parties output the same message (agreement) and with a high grade; but this is not necessarily true if the sender is corrupted. In that case, different parties might receive different messages (and with low grades). It is essential to note that the substitution of gradecast for broadcast within the VSS framework introduces a degree of uncertainty. This uncertainty stems from the fact that parties may not unanimously agree on whether to accept or reject the shared information, leading to potential divergence of outcomes within the VSS protocol.

The protocol of [KK06] works by having a designated party, called "a moderator", to be responsible for all gradecasts, so that if something goes wrong, we can know who is to blame (somewhat similar to the concept of identifiable abort in secure computation). Namely, whenever the VSS instructs a party to broadcast a message – that party gradecasts it; Moreover, the moderator then has to repeat the message (i.e., gradecast it), and parties proceed with the message gradecasted by the moderator. At the end of the sharing phase, each party outputs, together with the shares, a grade for the moderator in $\{0,1\}$. For instance, if a party P_j sees that the moderator repeats a different message than the one gradecsated by some party P_k (and was previously received with a high grade) – then clearly the moderator is malicious, and P_j sets the grade of the moderator to be 0.

At the end of this step, we have the guarantee that if the moderator is honest, all honest parties give grade 1 to the moderator, and we will always have an agreement on the VSSes it moderates. This is true for both the case where the dealer is honest, or the case where the dealer

is corrupted, regardless of whether the parties accept or reject the shares. If the moderator is dishonest, then we might not have an agreement, but it is enough that one honest party believes that the moderation was successful to guarantee that the underlying VSS was successful (i.e., all parties do agree whether to accept or reject the shares, but some might be uncertain of whether moderation was successful).

The protocol of [KK06] proceeds as follows:

1. For every $(i,j) \in [n^2]$ run a moderated VSS where P_i is the dealer and P_j is the moderator, and the secret is some random $c_{i\to j}$ chosen by P_i.
2. Reconstruct all secrets.
3. Each P_k sets $\mathsf{Successful}_k = \emptyset$; Add j to $\mathsf{Successful}_k$ if P_j successfully moderated all the n instances it moderated, and set $c_j = \sum_{\ell=1}^{n} c_{\ell\to j}$.
4. Each P_k chooses as a leader P_j for which c_j is minimal among all indices in $\mathsf{Successful}_k$.

All honest parties are included in all $\mathsf{Successful}_k$ for every honest k, and they see the same value c_j. If some honest party added to $\mathsf{Successful}_k$ some corrupted party P_i, then c_i must be uniform. A simple argument shows that an honest leader is chosen and agreed upon among all honest parties with some constant probability.

Reducing the amount of VSSes. In [KK06] there are n^2 independent instances of moderated VSS. As mentioned, we reduced the number of different secrets to $O(n \log n)$. Moreover, each party (a dealer) chooses randomly which $O(\log n)$ parties (moderators) it contributes to. Furthermore, we mentioned that which parties P_j contributes to must be kept secret at the sharing phase. This means that a party cannot know in advance which instances it has to moderate – those are chosen dynamically by the different dealers.

We address this challenge by implementing a novel moderation approach. Instead of assigning a single moderator to oversee each instance of VSS, all participating parties collectively assume the role of moderators for every instance. That is, we can envision each VSS execution as n parallel executions, with the same dealer, the same secret, but each instance is moderated by a different moderator. The moderation mechanism relies on the fact that at least two-thirds of the moderators are honest. As a result, we can look at the majority of the n different executions. If the dealer is honest, then all honest parties output, at the end of the reconstruction phase, the secret that the dealer shared. If the dealer is dishonest, then we might have disagreements as different moderators can make the VSS go into different directions. However, if one honest party believes that the moderation of some P_j was successful, then all honest parties unanimously output the same secret in the instance where P_j moderated.

Efficiency. The above mechanism is now problematic from an efficiency perspective. In particular, we tried to reduce the number of VSS executions from n^2 to $O(n \log n)$; instead, in each instance we have n moderators, and so we get $O(n^2 \log n)$ executions! Each VSS has $O(n^2 \log^2 n)$ bits over point-to-point and $O(n \log n)$ bits gradecasted, which results in a total of $O(n^4 \log n)$ bits over point-to-point.

To circumvent this issue, we first note that the point-to-point messages should not be repeated between the n different moderators inside a VSS execution. Moreover, following the idea of [AAPP22], we let the dealer moderate all messages except for the last message in which the execution "forks" into n different executions, corresponding to the n moderators. Each moderator echos just the last round (a vote on whether to accept or reject the shares of the dealer).

However, this idea alone does not fully address our requirements. Even echoing the last message proves to be prohibitively costly. To put it into perspective, echoing n bits of votes across n dealers, overseen by n moderators, results in the gradecast of n^3 bits. When accounting for the inherent overhead of gradecast, we are confronted once again with a total communication cost of $O(n^4)$.

To achieve the desired communication efficiency, we implement a batching mechanism wherein a single message is echoed by each moderator, applicable to the n distinct dealers they oversee. In essence, this identical message is utilized across multiple VSS executions, providing a

substantial reduction in communication complexity. In [FM88], there are n^2 independent executions, one for each $(i,j) \in [n]^2$ where P_i is the dealer, and P_j is the moderator. In [AAPP22], there are n independent executions, for every P_i, $i \in [n]$, where each execution is one dealer with n moderators at the same time. In our case, we have one big execution that contains $O(n \log n)$ secrets. That is, in our case, all the different VSSes intertwine - the same message of the moderator is used across the different dealers.

More precisely, each moderator gradecasts a single message of size $O(n)$, and this message is shared across all executions, serving as a universal indicator for which dealers' shares should be accepted or rejected. In the context of the n moderators, our approach results in a gradecast of $\tilde{O}(n^2)$ bits, which, when factoring the gradecast overhead, totals to $\tilde{O}(n^3)$. This optimization substantially improves the overall communication complexity. We refer the reader to Sections 5 and 6 for more details.

2.2 Efficient Statistical VSS

We show a statistically secure VSS with low broadcast cost. We start with a brief overview of the VSS protocol of Ben Or, Goldwasser and Wigderson [BGW88]; we then overview the VSS protocol of Abraham et al. [AAPP23] and proceed to our protocol. At a very high level, in [BGW88], each party broadcasts $O(n \log n)$ bits and the dealer might broadcast up to $O(n^2 \log n)$ bits. The work of [AAPP23] shows how to reduce the broadcast cost of the dealer to $O(n \log n)$. Our goal is to reduce the cost of all parties except the dealer to $O(\log^2 n)$ bits, albeit achieving only statistical security (or even $O(\log n)$ bits, with a one over poly error probability). Thus, in total we have $O(n \log n)$ bits broadcasted.

In this overview, we describe a statistically-secure protocol with negligible error probability (in n), while for our OLE it suffices to have a one over poly error.

Overview of the VSS of [BGW88]. To share a secret s, the dealer chooses a bivariate polynomial $S(x,y) = \sum_{k=0}^{t} \sum_{\ell=0}^{t} s_{k,\ell} x^k y^\ell$ of degree t in both x and y, where $s_{0,0} = s$. The protocol is as follows:

1. **Sharing:** The dealer gives to each P_i its shares $(f_i(x), g_i(y)) = (S(x,i), S(i,y))$.
2. **Pairwise consistency check:** P_i sends to P_j the points $(f_i(j), g_i(j)) = (S(j,i), S(i,j)) = (g_j(i), f_j(i))$. If P_i sees that the shares it received from the dealer do not agree with the points it received from P_j it publicly **broadcasts** a complaint $\mathsf{complaint}(i, j, f_i(j), g_i(j))$.
3. **The dealer resolves complaints:** Note that in each complaint, P_i is supposed to provide the values it received from the dealer, not those that it received from P_j. If the dealer notices some public complaint that is wrong, i.e., contains points that it did not provide P_i, it completely reveals the shares of that complaining party, by **broadcasting** $(i, S(x,i), S(i,y))$.
4. If a party P_j sees that (1) all polynomials that the dealer made public agree with its private share; (2) its share was not made public; (3) if there is a joint complaint - two parties P_k and P_ℓ that disagree with one another, then the dealer must publicly reveal the share of one of them. If all those conditions are met, then P_i is happy. If there are $2t+1$ parties that are happy, then the shares are accepted.

If the dealer is honest, then their shares are always consistent and honest parties never complain on honest parties; Moreover, all honest parties are happy. Furthermore, the adversary learns nothing new in the verification process – note that all possibly revealed shares are shares of corrupted parties, which the adversary has anyways.

If the dealer is corrupted, then $2t+1$ parties that are happy implies that there is a core set of at least $t+1$ honest parties that are happy. The shares of these honest parties must agree with each other; otherwise, those parties would have raised a complaint and the dealer must have publicly reveal one of them. The shares of those happy honest parties uniquely define a bivariate polynomial $S'(x,y)$ of degree t in both variables. Moreover, all other honest parties must hold shares on that polynomial – if some honest P_i initially held a polynomial that disagrees with

some P_j that is in the core set, then both P_i and P_j raised a complaint. Since P_j is in the core set, then the dealer must have resolved this complaint by revealing the share of P_i, and the entire core verified this polynomial (and therefore, it must agree with $S'(x, y)$).

Costs, and the VSS protocol of [AAPP23]. All broadcasts in the above protocol are marked in **bold**: Each party might broadcast up to $O(n)$ complaints; The dealer might broadcast $O(n^2 \log n)$ bits (e.g., revealing the shares of t parties). The protocol of [AAPP23] provides a key improvement that we borrow: it reduces the broadcast cost of the dealer from $O(n^2 \log n)$ to $O(n \log n)$. The goal was to allow batching of many parallel instances of VSS (with the same dealer) with the same broadcast cost of just one instance. Towards that end, all broadcasts made by the dealer should not be instance specific (e.g., revealing shares), but information that is useful across multiple instances. We use batching and benefit from this reduction in the broadcast cost of the dealer.

To elaborate further, instead of the dealer broadcasting the shares of each party that falsely complained, [AAPP23] let the dealer just broadcast a set CONFLICTS of the identities of parties that raised false complaints. In a sense, the dealer "revokes" the shares of all parties in CONFLICTS. Yet, the parties cannot conclude the protocol unless all parties in CONFLICTS receive correct shares. At this point, there are three possible outcomes:

1. Discard the dealer - this might occur when there is a joint complaint that was not resolved (P_i complaint against P_j but none of them is in CONFLICTS), or when the dealer broadcasts a set CONFLICTS that contains more than t parties.
2. If $|\mathsf{CONFLICTS}| > t/2$ then there are too many shares "to correct". Instead of publicly broadcasting the shares of those parties, they are all just set to be 0, and the protocol is restarted while the dealer chooses a new bivariate polynomial where all the shares of all parties in CONFLICTS are set to be 0. In that case, the dealer, in a sense, publicly reveals their shares, but without broadcasting them. In the next iteration, parties do not expect to receive shares from parties in CONFLICTS, and each party P_i verifies that $f_i(j) = g_i(j) = 0$ for $j \in \mathsf{CONFLICTS}$. Formally, the parties maintain a public set ZEROS of parties whose shares are set to be 0 and before the next iteration ZEROS is updated to $\mathsf{ZEROS} \cup \mathsf{CONFLICTS}$. Note that we can restart only once as restarting twice means that the dealer tries to revoke $> t/2 + t/2 = t$ parties ($|\mathsf{ZEROS}| > t/2$ and $|\mathsf{CONFLICTS}| > t/2$) and is being discarded.
3. Otherwise, the parties proceed to a sub-protocol where they reconstruct all shares of parties in CONFLICTS, with the help of the dealer. We elaborate on this part later below.

Our goal: reducing the broadcast cost. Note that if the parties decide to proceed, then we have binding – there is a unique bivariate polynomial $S(x, y)$ of degree at most t in x and y, such that all honest parties that are not in CONFLICTS hold shares on $S(x, y)$. This is because each pair of honest parties that disagree with each other must have raised a joint complaint, and the dealer must have included at least one of them in CONFLICTS. Therefore, all shares of honest parties not in CONFLICTS must agree, and thus define a bivariate polynomial of the appropriate degree. Moreover, note that all honest parties that are not in CONFLICTS have shares, and that $|\mathsf{CONFLICTS}| < t/2$; as such, at least $t + t/2 + 1$ honest parties have shares (as opposed to just $t + 1$ as the core set in [BGW88]) – not only that there is a well defined polynomial, but there is also some redundancy.

For reconstructing the shares of parties in CONFLICTS, it is crucial that all honest parties that are not in CONFLICTS have shares that agree with each other. However, this is achieved using the fact that each party complains against each party that they did not agree with. But this requires a high broadcast cost.

To reduce the broadcast cost, we instruct each party to limit the number of complaints they file. Specifically, each party now randomly samples $O(\log^2 n)$ complaints out of the $O(n)$ that it might have. This raises two questions – (1) why $O(\log^2 n)$ complaints suffice; (2) What if P_i chooses P_j but P_j picked other parties? In that case, we would not have a joint complaint, and the dealer can just ignore complaints without being discarded.

Addressing the second question is easy. We just add one more round of complaints – if a party P_i complains against P_j, and P_j did not choose to complain against P_i in the first round of complaints, then it complains against it at the second round of complaints. This approach effectively doubles the number of complaints, maintaining the overall count at $O(\log^2 n)$ per party. However, this additional step guarantees that if two honest parties disagree, and one picks the other in its random choices, then all parties will see a joint complaint, forcing the dealer to resolve it.

To address the first question, we claim that with high probability – $O(\log^2 n)$ complaints by each party suffice to have binding with overwhelming probability (in n). To see why it holds, consider for simplicity the case where all honest parties (that are not in CONFLICTS) agree with each other, except for some party P_i. In that case, we have a set of $t + t/2$ honest parties that agree with each other. Moreover, P_i has a polynomial of degree t – and therefore, *it must disagree with at least $t/2$ parties* in the core, and not just one. Each one of these parties picks $O(\log^2 n)$ random complaints – the probability that none of those random choices made by either i, or by those $t/2$ parties that disagree with P_i, is bounded by $((1 - 1/n)^{t/2})^{\log^2 n}$, which is negligible in n. This argument is generalized to other cases beyond this simple case of all honest parties agreeing but one. We remark again that for our purpose of broadcast, it suffices that each party chooses $O(\log n)$ complaints out of the $O(n)$ that it might have; this results in an error probability of $O(1/\mathsf{poly}n)$.

Reconstruction of shares for parties in CONFLICTS. In the second stage of the protocol, the parties face the challenge of reconstructing the shares of all parties within the CONFLICTS set on the bivariate polynomial $S(x, y)$. We cannot simply adopt the approach used in [BGW88], where the dealer broadcasts these shares, because doing so would require a broadcast of $O(n^2 \log n)$ bits by the dealer. Moreover, it's not feasible for each party P_k (for $k \notin$ CONFLICTS) to directly transmit the value $f_k(j)$ to each party P_j (for $j \in$ CONFLICTS), as in the worst-case scenario, where we have $t + t/2 + 1$ correct points and possibly t errors, party P_j might fail to decode its share. Similarly, public reconstruction (utilizing the dealer to eliminate errors) is not viable either, as it would necessitate each party ($k \notin$ CONFLICTS) to broadcast $O(n \log n)$ bits, once again resulting in a total broadcast of $O(n^2 \log n)$ bits. We remark that public reconstruction is the approach taken in [AAPP23]. Here we take a different route to reduce the broadcast complexity.

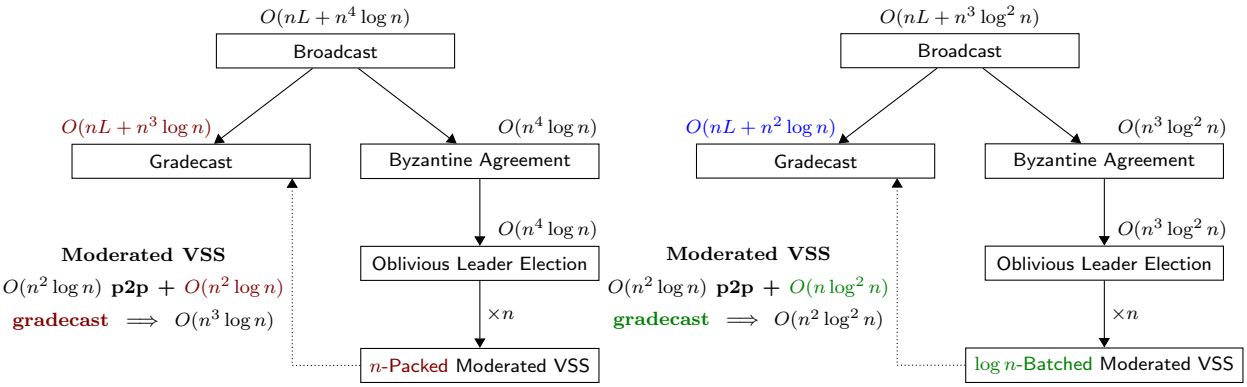
Instead of reconstructing the f and g polynomials of parties in CONFLICTS as in [AAPP23], we reconstruct just the points $g_j(0) = S(j, 0)$ for each party $j \in$ CONFLICTS. This suffices, as together with the shares of the parties that are in CONFLICTS, the parties can reconstruct the polynomial $S(x, 0)$ in the reconstruction phase, which suffices for reconstructing $S(0, 0)$. As an immediate consequence, our protocol for the reconstruction of the shares for parties in CONFLICTS requires each of the parties to broadcast only $O(\log n)$ bits.

Assume without loss of generality, and for simplicity of notation, that the set CONFLICTS $= \{1, \ldots, c\}$ where $c \leq t/2$. Consider the $c \times t$ bivariate polynomial $V(x, y) = \sum_{j=1}^{c} x^{j-1} \cdot S(j, y)$. Our goal is to publicly reconstruct the degree c univariate polynomial $f_0^V(x) := V(x, 0) := \sum_{j=1}^{c} x^{j-1} \cdot S(j, 0)$. Observe that the coefficients for this polynomial are $S(j, 0)$ for every $j \in$ CONFLICTS. The parties will reconstruct this polynomial $V(x, 0)$ publicly, and then they can recover $S(1, 0), \ldots, S(c, 0)$, as required. Moreover, note that this is a polynomial of degree at most $c \leq t/2$.

Towards that end, each party P_k for $k \notin$ CONFLICTS can locally compute

$$f_k^V(x) := V(x, k) = \sum_{j=1}^{c} x^{j-1} \cdot S(j, k) = \sum_{j=1}^{c} x^{j-1} \cdot f_k(j) \ .$$

Each P_k sends $f_k^V(i)$ to each party P_i for $i \notin$ CONFLICTS. Now each such party P_i tries to reconstruct $V(i, y)$ from all the points $(f_k^V(i))_{k \notin \mathsf{CONFLICTS}}$ that it received. Note that $V(i, y)$ is of degree t, and since $|\mathsf{CONFLICTS}| \leq t/2$, P_i receives at least $t + t/2 + 1$ correct points, but might receive up to t incorrect points. Once received more than $t/2$ incorrect points, there is no



(a) Before our work.

(b) After our work.

Figure 1: The structure of Broadcast, including costs before and after our improvements. Our improvements are in oblivious leader election and in (moderated) batched VSS. We use the gradecast protocol of [ZLC23] which we denote in blue.

unique reconstruction. In that case, P_i broadcasts $\mathsf{complaint}(i)$. If P_i successfully learns $V(i, y)$, then it can compute $f_0^V(i) = V(i, 0)$ and publish it. The dealer (which knows $V(x, y)$) listen to all published messages, and add to a set Bad all parties that complaints or that published incorrect values. The dealer broadcasts Bad.

The parties discard the dealer if Bad is too large, or restart the protocol if $|\mathsf{Bad}| > t/2$. Otherwise, $|\mathsf{Bad}| \leq t/2$, and the dealer "confirmed" $n - |\mathsf{CONFLICTS}| - |\mathsf{Bad}| \geq n - t/2 - t/2 \geq 2t + 1$ points on a univariate polynomial $f_0^V(x)$. All the points on this polynomial are public, and the dealer must take care that all points (for those it did not include in Bad) must lie on a unique univariate polynomial of degree at most $c \leq t/2$ (by including parties in Bad if some point is incorrect). If not, then the dealer is publicly discarded. Therefore, we are guaranteed that all parties have $f_0^V(x)$, and from its coefficients, all parties learn the values $g_1(0), \ldots, g_c(0)$.

We note that the above share reconstruction protocol is *perfectly* secure (as opposed to the first part, which was just statistically secure). Moreover, each party broadcasts just $O(\log n)$ bits, and the dealer broadcasts $O(n \log n)$ bits, leading to a total of $O(n \log n)$ bits broadcasted.

2.3 Putting it All Together

We now present our overall broadcast protocol, which follows the paradigm of [FM88, KK06]. We also refer the reader to Appendix A for an overview of the different primitives. To broadcast a message L:

1. The sender gradecasts the message L. Each party receives a message L_i with a grade $g_i \in \{0, 1, 2\}$.
2. The parties run Byzantine agreement on the grade g_i. If $g_i = 2$ then the input of the Byzantine agreement is 1. Otherwise, it is 0. Byzantine agreement intuitively works as follows:
   (a) The parties try to see if they all hold the same bit as input. Along the way, they send to each other the input bit. If they agree - they halt and output that bit.
   (b) If there is no agreement – they obliviously elect a leader. They run the protocol again with the leader's value that was sent in the previous step.

Gradecasting a message of size L requires $O(nL + n^2 \log n)$ communication [ZLC23]. Our OLE requires $O(n^3 \log^2 n)$, and the additional messages of Byzantine agreement require $O(n^2)$ bits per iteration. Overall we get $O(nL)$ plus expected $O(n^3 \log^2 n)$ with an expected constant number of rounds. The costs are described in Figure 1. We denote the costs that we improve in red in Figure 1a and in green in Figure 1b.

Parallel broadcast. For our parallel broadcast, we can follow the idea of Fitzi and Garay [FG03] and use a single election across all instances. That is, each party gradecasts its message, and then the parties run n instances of Byzantine agreement where they use the same leader across all n instances. Once an honest leader is chosen, all instances can reach agreement. Since we have n gradecasts, we get cost of $O(n^2L + n^3 \log n)$ for the gradecasts, and in addition $O(n^3 \log^2 n)$ per instance of OLE, leading to $O(n^2L)$ plus expected $O(n^3 \log^2 n)$ with expected constant number of rounds.

As for the lower bound, our lower bound says that there is no parallel broadcast with $o(n^2L + n^3)$ bits total communication. We show that if there exists a parlallel broadcast with the above cost, then there is a broadcast protocol (with a single sender) that requires $o(nL)$ bits transmitted. See Section 7.

Organization. The rest of the paper is organized as follows. In Section 3, we provide the preliminaries. In Section 4 we provide our statistical verifiable secret sharing, and in Section 5 we provide our batched multi-moderated VSS. We present the OLE protocol in Section 6 and the broadcast protocol, parallel broadcast and the lower bound in Section 7.

3 Preliminaries

We consider the set of parties represented by identities in $[n] := \{1, \ldots, n\}$ who are connected by pair-wise private and authenticated channels. We alternate as convenient between referring to parties by their identity $i \in [n]$ or as P_i where $i \in [n]$. In our verifiable secret sharing protocol we assume the parties have access to a broadcast channel as well. Up to $t < n/3$ of the parties are maliciously corrupted by a *computationally unbounded* active adversary $\mathcal{A}$. Our security proofs are all in the standalone model for a static adversary. The standalone security implies adaptive security with inefficient simulation [CDD+01] and universal composability [Can00] due to [KLR06].

We also assume for our protocols the existence of a finite field $\mathbb{F}$ where $|\mathbb{F}| > n + 1$ and the set of values $\{0, 1, \ldots, n\}$ is a distinct set of elements known apriori to all the parties.

We also provide additional definitions, including statistical security, hybrid model and composition, and some properties on bivariate polynomials. We also provide the ideal functionalities for our final primitives – Broadcast and Byzantine agreement.

3.1 Security Definition

As mentioned before, we state and prove security for our protocols in the standalone model [Can00, AL17]. Let $f : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality and let π be a protocol over the parties $[n]$ where the parties are connected pair-wise private and authenticated channels. The adversary $\mathcal{A}$ has auxiliary input z, and let $I \subset [n]$. The real and ideal executions are defined as follows:

- **The real execution:** In this world, the parties run the protocol π where the adversary $\mathcal{A}$ maliciously corrupts the parties in I. The adversary can see the messages of the honest parties before sending messages for the corrupted parties, that is, it is rushing. By definition, the messages between the honest parties on the point-to-point channels are hidden from the adversary. The random variable $\mathsf{Real}^{\pi}_{\mathcal{A}(z),I}(\bar{x})$ denotes the transcript of messages as seen by $\mathcal{A}$ in the execution (including the corrupted parties' inputs and internal randomness) and the honest parties outputs, where the parties start with inputs $\bar{x} = (x_1, \ldots, x_n)$.
- **The ideal execution :** The ideal model consists of honest parties, a trusted party or ideal functionality, and an ideal adversary $\mathcal{SIM}$ controlling the same subset I of parties. The honest parties send their inputs to the ideal functionality. $\mathcal{SIM}$ receives the auxiliary input z and the inputs of the corrupted parties. $\mathcal{SIM}$ may substitute the inputs of the corrupted parties (as long as the length of the inputs remains the same). The ideal functionality then receives all the inputs $x_1, \ldots, x_n$ where $\mathcal{SIM}$ may have replaced some inputs and computes the output $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and sends y_j to P_j for the

honest parties and hands the outputs of the corrupted parties to the adversary. The random variable $\mathsf{Ideal}^{f}_{\mathcal{SIM}(z),I}(\bar{x})$ denotes the output of $\mathcal{SIM}$ and the honest parties.

Adaptive security. As mentioned, we need to prove the security of our statistical primitives against an adaptive adversary (else, adaptive security of our broadcast protocol cannot be derived for free from [CDD+01]). We follow the definition of adaptive corruptions from [ACS22]. However, our statistical primitives are *reactive* and we need to make slight changes to the definition from [ACS22]. [ACS22, Section 2.4] defines adaptive corruptions for a computation at each of the following stages: (1) before inputs; and (2) after inputs and computation; and (3) post-execution. It thus suffices to allow the adaptive ideal world adversary perform adaptive corruptions (1) before a reactive interaction; and (2) after the reactive interaction; and (3) after the ideal functionality has finished executing. Note that for an ideal functionality with exactly one interaction (non-reactive) the above is identical to the adaptive definition of [ACS22].

Definition 3.1. *A protocol π perfectly realizes a functionality f, if for every adversary $\mathcal{A}$ in the real world, there exists an ideal adversary $\mathcal{SIM}$ such that for each $I \subset [n]$ with $|I| \leq t$, it holds that*

$$\{\mathsf{Ideal}^{f}_{\mathcal{SIM}(z),I}(\bar{x})\} \equiv \{\mathsf{Real}^{\pi}_{\mathcal{A}(z),I}(\bar{x})\}$$

where $\bar{x} \in (\{0,1\}^*)^n$ such that $|x_1| = \ldots = |x_n|$.

Statistical security. For certain protocols π (specifically, for the verifiable secret sharing protocol used to construct the oblivious leader election), we show that the distributions on the real and ideal executions are statistically close instead of identical. Furthermore, we give an upper bound on the distance ϵ between the real and ideal executions. We term the aforementioned notion as *statistical* security and define it as follows:

Definition 3.2. *A protocol π statistically realizes a functionality f, if for every adversary $\mathcal{A}$ in the real world, there exists an ideal adversary $\mathcal{SIM}$ such that for each $I \subset [n]$ with $|I| \leq t$, it holds that*

$$\Delta\left(\{\mathsf{Ideal}^{f}_{\mathcal{SIM}(z),I}(\bar{x})\}, \{\mathsf{Real}^{\pi}_{\mathcal{A}(z),I}(\bar{x})\}\right) \leq \epsilon$$

where $\bar{x} \in (\{0,1\}^*)^n$ such that $|x_1| = \ldots = |x_n|$,

$$\Delta(X,Y) = \frac{1}{2}\left|\sum_{v \in \mathcal{V}} \Pr[X = v] - \Pr[Y = v]\right|$$

denotes the statistical distance between the ensembles X and Y that are taken values in $\mathcal{V}$ (i.e., $\mathcal{V}$ is the union of the supports of X and Y), and $0 \leq \epsilon < 1$ is called the statistical error for protocol π.

By definition, a protocol with statistical error $\epsilon = 0$ is perfectly secure.

Hybrid model and composition. For modularity, we use the hybrid model. In a protocol in the π_g^f in the f-hybrid model, the parties have access to a trusted party that ideally computes the function f for them. The composition theorem due to [Can00] states that if a protocol π_g^f securely implements g, and a protocol π_f securely implements f, then when replacing in the protocol π_g^f all invocations of f with the protocol π_f, then the resulting protocol also securely implements g.

For statistical security, suppose that π_g^f in the f-hybrid model perfectly realizes some functionality g, and let T bounds the number of times it invokes f in all executions; moreover, suppose π_f realizes f in the plain model with statistical error ϵ. Then, there exists a protocol π that implements g in the plain model with statistical error $T \cdot \epsilon$.

3.2 Bivariate Polynomials

A degree (ℓ, m)-bivariate polynomial over $\mathbb{F}$ is of the form $S(x,y) = \sum_{i=0}^{\ell} \sum_{j=0}^{m} b_{ij} x^i y^j$ where each $b_{ij} \in \mathbb{F}$. The polynomials $f_i(x) = S(x,i)$ and $g_i(y) = S(i,y)$ are called i^{th} row and column polynomials of $S(x,y)$ respectively. In our protocol, we use (t,t)-bivariate polynomials where the i^{th} row and column polynomials are associated with the party P_i.

Lemma 3.3 (Pair-wise Consistency Lemma [CCP22]). *Let $\{f_{i_1}(x), \ldots, f_{i_q}(x)\}$ and $\{g_{j_1}(y), \ldots, g_{j_r}(y)\}$ be degree ℓ and degree m polynomials respectively where $q \geq m+1, r \geq \ell+1$ and where $i_1, \ldots, i_q, j_1, \ldots, j_r \in \{1, \ldots, n\}$. Moreover, let for every $i \in \{i_1, \ldots, i_q\}$ and every $j \in \{j_1, \ldots, j_r\}$, the condition $f_i(\alpha_j) = g_j(\alpha_i)$ holds. Then there exists a unique degree-(ℓ, m) bivariate polynomial $S^*(x,y)$, such that the polynomials $f_{i_1}(x), \ldots, f_{i_q}(x)$ and $g_{j_1}(y), \ldots, g_{j_r}(y)$ lie on $S^*(x,y)$.*

3.3 Ideal Functionalities for Broadcast and Byzantine Agreement

We provide here ideal functionalities for broadcast and byzantine agreement, as those are our final objectives. We introduce other functionalities along the way in the technical sections.

Broadcast. In broadcast, we have a sender that holds a message M, and all honest parties are supposed to agree on the message. It requires: (1) validity:

Its ideal functionality is therefore simple.

Functionality 3.4: $\mathcal{F}_{\mathsf{BC}}$

The functionality is parameterized by a value L.

1. The dealer (sender) sends to the functionality its message $M \in \{0,1\}^L$.
2. The functionality sends M to all the parties.

Byzantine agreement. In Byzantine agreement, each party holds as input a message $M_i \in \mathcal{M}$ from some message space $\mathcal{M}$ and output a message where it is guaranteed that all honest parties output the same message (consistency). Furthermore, it is also guaranteed that if all honest parties held the same input message M, then M is the output (validity).

Functionality 3.5: $\mathcal{F}_{\mathsf{BA}}$: Byzantine Agreement

The functionality is parameterized by the set of corrupted parties $I \subset [n]$ and a message space $\mathcal{M}$.

1. The functionality receives from each honest party P_j its input $M_j \in \mathcal{M}$. The functionality sends $(M_j)_{j \notin I}$ to the adversary.
2. The adversary sends a message $\hat{M}$.
3. If there exists a message M such that $M_j = M$ for each $j \notin I$, then set $y = M$. Otherwise, set $y = \hat{M}$.
4. The functionality gives y to all parties.

In case where the message space $\mathcal{M}$ is $\{0,1\}$ we call this primitive "bit-Byzantine agreement".

4 Statistical Verifiable Secret Sharing

In this section, we provide our statistically secure verifiable secret sharing, which has low broadcast cost. In Section 4.1, we give the *sharing attempt* protocol, where the dealer tries to share its secret, but it might fail. At the end of an execution of the sharing attempt, there exists a set CONFLICTS of parties of size at most $t/2$ such that each honest party $P_j \notin$ CONFLICTS holds

$f_j(x), g_j(y)$ where $f_j(x) = S(x, j)$, $g_j(y) = S(j, y)$ for some unique degree-(t, t) bivariate polynomial $S(x, y)$. Furthermore, if the dealer was honest, no honest party is included in CONFLICTS. Specifically, at the end of a sharing attempt, one of the following outcomes occurs:

1. If $|\mathsf{CONFLICTS}| > t/2$, then the protocol is restarted (after publicly fixing the shares of the parties in CONFLICTS).
2. The dealer is discarded;
3. $|\mathsf{CONFLICTS}| \leq t/2$. In this case, the protocol proceeds to reconstruct the shares of the parties in CONFLICTS. See Section 4.2.

4.1 Sharing Attempt

We realize the sharing attempt functionality $\mathcal{F}_{\mathsf{ShareAttempt}}$ (Functionality 4.1) with only statistical security and hence we need to additionally prove the adaptive security of our protocol. In the ideal execution, the adversary may adaptively corrupt parties after each interaction between the parties and the ideal functionality.

Functionality 4.1: $\mathcal{F}_{\mathsf{ShareAttempt}}$

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

1. All the parties send to $\mathcal{F}_{\mathsf{ShareAttempt}}$ a set $\mathsf{ZEROS} \subseteq [n]$. For an honest dealer, it holds that $\mathsf{ZEROS} \subseteq I$.
2. $\mathcal{F}_{\mathsf{ShareAttempt}}$ sends the set ZEROS to the adversary.
3. The dealer sends a polynomial $S(x, y)$ to $\mathcal{F}_{\mathsf{ShareAttempt}}$. When either the polynomial is not of degree at most t in x and y, or for some $i \in \mathsf{ZEROS}$ it holds that $S(x, i) \neq 0$ or $S(i, y) \neq 0$, $\mathcal{F}_{\mathsf{ShareAttempt}}$ executes Step 6c to discard the dealer.
4. For every $i \in I$, $\mathcal{F}_{\mathsf{ShareAttempt}}$ sends $(S(x, i), S(i, y))$ to the adversary.
5. Receive a set CONFLICTS from the adversary such that $\mathsf{CONFLICTS} \cap \mathsf{ZEROS} = \phi$. If the dealer is honest, then $\mathsf{CONFLICTS} \cup \mathsf{ZEROS} \subseteq I$. If $|\mathsf{CONFLICTS} \cup \mathsf{ZEROS}| > t$ for a corrupt dealer, then $\mathcal{F}_{\mathsf{ShareAttempt}}$ executes Step 6c to discard the dealer.
6. **Output:**
   (a) Detect: If $|\mathsf{CONFLICTS}| > t/2$, then send $(\mathsf{detect}, \mathsf{CONFLICTS})$ to all parties
   (b) Proceed: Otherwise, send $(\mathsf{proceed}, S(x, i), S(i, y), \mathsf{CONFLICTS})$ to every $i \notin \mathsf{CONFLICTS}$ and $(\mathsf{proceed}, \perp, \perp, \mathsf{CONFLICTS})$ to every $i \in \mathsf{CONFLICTS}$.
   (c) Discard: send discard to all the parties.

Protocol 4.2: $\Pi_{\mathsf{ShareAttempt}}$

Input: All parties input $\mathsf{ZEROS} \subset [n]$. The dealer, denoted as the party P for simplicity, inputs a polynomial $S(x, y)$ with degree t in x and y, such that for each $P_i \in \mathsf{ZEROS}$ it holds that $S(x, i) = 0$ and $S(i, y) = 0$.

The protocol:

1. **(Dealing shares)**: The dealer P sends $(f_i(x), g_i(y)) = (S(x, i), S(i, y))$ to $P_i \notin \mathsf{ZEROS}$. Each $P_i \in \mathsf{ZEROS}$ sets $(f_i(x), g_i(y)) = (0, 0)$.
2. **(Pairwise Consistency Checks):**
   (a) Each $P_i \notin \mathsf{ZEROS}$ sends $(f_i(j), g_i(j))$ to every $P_j \notin \mathsf{ZEROS}$. Let (f_{ji}, g_{ji}) be the values received by P_i from P_j.
   (b) Initialize a set $\mathsf{Complaints}_i = \emptyset$. If it holds that $f_{ji} \neq g_i(j)$ or $g_{ji} \neq f_i(j)$ then add j to $\mathsf{Complaints}_i$.
   (c) If there exists a $j \in \mathsf{ZEROS}$ for which $f_i(j) \neq 0$ or $g_i(j) \neq 0$, then broadcast $(\mathsf{complaint}, i)$.
3. **(Random Complaints):**
   (a) P_i samples a set S_i by choosing m elements from $[n]$ (with replacements).

(b) For every $j \in \mathsf{Complaints}_i \cap S_i$, the party P_i broadcasts $(\mathsf{complaint}, i, j, f_i(j), g_i(j))$.

4. **(Confirming Random Complaints):**
   (a) For each P_j that broadcast $(\mathsf{complaint}, j, i, \cdot, \cdot)$, if $j \in \mathsf{Complaints}_i$ and P_j broadcasted at most R complaints in the previous step, then P_i also broadcasts $(\mathsf{complaint}, i, j, f_i(j), g_i(j))$.
5. **(Conflict Resolution):**
   (a) P sets $\mathsf{CONFLICTS} = \phi$. Add $i \notin \mathsf{ZEROS}$ to $\mathsf{CONFLICTS}$ if one of the following occurs: (a) P_i broadcasted $(\mathsf{complaint}, i)$; (b) P_i broadcasted more than m complaints in Step 3b; (c) P_i broadcasted $(\mathsf{complaint}, i, j, u, v)$ such that $u \neq S(j, i)$ or $v \neq S(i, j)$.
   (b) P broadcasts $\mathsf{CONFLICTS}$.
6. **(Output):**
   (a) Each P_i outputs $\mathsf{discard}$ if any one of the following does not hold: (i) $\mathsf{ZEROS} \cap \mathsf{CONFLICTS} = \phi$; (ii) $|\mathsf{ZEROS} \cup \mathsf{CONFLICTS}| \leq t$; (iii) if P_i broadcasts $(\mathsf{complaint}, i)$ but $i \notin \mathsf{CONFLICTS}$; (iv) If P_i broadcasted more than m complaints in Step 3b but $i \notin \mathsf{CONFLICTS}$; (v) If P_i broadcasted $(\mathsf{complaint}, i, j, u_i, v_i)$ and P_j broadcasted $(\mathsf{complaint}, i, j, u_j, v_j)$ with $u_i \neq v_j$ or $v_i \neq u_j$, and neither i or j in $\mathsf{CONFLICTS}$.
   (b) If $|\mathsf{CONFLICTS}| > t/2$, then each P_i outputs $(\mathsf{detect}, \mathsf{CONFLICTS})$.
   (c) Else, $P_i \in \mathsf{CONFLICTS}$ outputs $(\mathsf{proceed}, \perp, \perp, \mathsf{CONFLICTS})$ and $P_i \notin \mathsf{CONFLICTS}$ outputs $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$.

The simulation for the sharing attempt protocol of [AAPP23] can be repurposed to work for our protocol; however, it is conditioned on the following event: *all conflicts between honest parties is resolved by the dealer*. Towards that end, we first provide an upper bound on the probability that the above event occurs (Claim 4.3) and then prove the security of Protocol 4.2 (Theorem 4.4).

To formalize the above event, we first define a *clique*. A set $C \subseteq [n]$ is a clique if for each ordered pair $i, j \in C$ it holds that the parties P_i and P_j agreed with each other in Step 2c, i.e., P_i did not include P_j in $\mathsf{Complaints}_i$.

Claim 4.3. *The probability that all honest parties (at the end of the protocol) output* $\mathsf{proceed}$ *but there is no clique $C \subseteq H$ among the honest parties (H is the set of parties that remain honest until the end of the protocol) that are not in* $\mathsf{CONFLICTS}$ *is at most $\epsilon = n \cdot e^{-m/6}$. Recall that m is the size of the set S_i chosen by each party P_i in Step 3b.*

Proof. Let $K = H \setminus \mathsf{CONFLICTS}$ where H is the set of honest parties at the end of the protocol. Since $|\mathsf{CONFLICTS}| \leq t/2$ and $|H| \geq 2t+1$, we have that $|K| \geq t + t/2 + 1$. Let Bad denote the following events:

> $\underline{\mathsf{Bad}}$*:* *The parties output* $\mathsf{proceed}$ *but there is no clique among K; Namely, all parties in K output* $\mathsf{proceed}$ *but the maximal clique $C \subseteq K$ is of size $< |K|$, where C is a clique if for every $i, j \in C$ it holds that $i \notin \mathsf{Complaints}_j$ and $j \notin \mathsf{Complaints}_i$.*
>
> $\underline{\mathsf{Bad}_c}$*:* *The parties output* $\mathsf{proceed}$ *but the maximal clique $C \subseteq K$ is of size c, where C is a clique if for every $i, j \in C$ it holds that $i \notin \mathsf{Complaints}_j$ and $j \notin \mathsf{Complaints}_i$.*

It is easy to see that $\mathsf{Bad} = \bigcup_{c=1}^{|K|-1} \mathsf{Bad}_c$.

Bounding the probability of Bad_c. We define different cases according to the size of c (the maximal clique). We start as a warm up with the case of $c = |K| - 1$:

The case of $c = |K| - 1$. In this case, the shares of the parties in C define a unique bivariate polynomial of degree $t \times t$. In particular, this means that there exists some P_j that can agree with at most t parties in C, that is, $|\mathsf{Complaints}_j \cap C| \geq |C| - t = |K| - 1 - t$. When P_j broadcasts its complaints, it did not pick any one of those parties, and likewise, those

$|K| - 1 - t \geq t + t/2 + 1 - 1 - t \geq t/2$ parties did not complain against P_j. Otherwise, we must have a joint complaint, and the dealer must have included one of the two parties in $\mathsf{CONFLICTS}$ or be discarded. In particular, each one of those $\geq t/2$ chose m elements in $[n]$, none of them is j. Therefore,

$$\Pr\left[\mathsf{Bad}_{|K|-1}\right] \leq \left(\left(\frac{n-1}{n}\right)^m\right)^{t/2} \leq \left(\left(1-\frac{1}{n}\right)^n\right)^{m/6} \leq e^{-m/6} .$$

The case of $c \geq t+1$. As in the previous case, the shares of the parties in C define a unique bivariate polynomial of degree $t \times t$. Moreover, there is a set J of at least $|K| - c$ honest parties that are not in the clique C. Each one of those P_j, for $j \in J$ has $|\mathsf{Complaints}_j \cap C| \geq 1$, as otherwise we would have a larger clique. In fact, since the degree of the polynomial is t, P_j can agree with at most t parties in C. As such, $|\mathsf{Complaints}_j \cap C| \geq c - t$ (≥ 1) When randomly choosing the complaints, each P_j did not pick the $c-t$ parties it disagreed with from C. Likewise, there are at least $c-t$ parties in C that did not pick a corresponding (at least one) element in J, and the parties in C did not pick those parties in J. We get:

$$\Pr\left[\mathsf{Bad}_c\right] \leq \left(\left(\frac{n-1}{n}\right)^m\right)^{|K|-c} \cdot \left(\left(\frac{n-1}{n}\right)^m\right)^{c-t} \leq \left(\left(\frac{n-1}{n}\right)^m\right)^{t/2} \leq e^{-m/6} ,$$

where we use the fact that $|K| \geq t + t/2 + 1$ and so $|K| - t \geq t/2$.

The case of $c < t+1$. In that case, there is a set of $|K| - c \geq t/2$ parties, and each one of them disagrees with at least one party in C, and does not pick a disagreeing party when complaining. We have:

$$\Pr\left[\mathsf{Bad}_c\right] \leq \left(\left(\frac{n-1}{n}\right)^m\right)^{t/2} \cdot e^{-m/6} .$$

Putting it all together. By a simple union bound, we can bound:
$\Pr\left[\mathsf{Bad}\right] \leq \sum_{c=1}^{|K|-1} \Pr\left[\mathsf{Bad}_c\right] \leq n \cdot e^{-m/6} = \epsilon.$ □

As mentioned before, the simulator constructed in the proof of security for the sharing attempt protocol of [AAPP23] can be re-purposed to work for our protocol. However, as our protocol is only statistically secure we cannot derive adaptive security directly from [CDD+01]. We provide the *adaptive* simulation security proof for our statistical sharing attempt protocol in the following theorem.

Theorem 4.4. *Protocol $\Pi_{\mathsf{ShareAttempt}}$ (Protocol 4.2), securely computes the functionality $\mathcal{F}_{\mathsf{ShareAttempt}}$ (Functionality 4.1) with statistical security, in the presence of a malicious* **adaptive** *adversary controlling at most $t < n/3$ except with probability $\epsilon = n \cdot e^{-m/6}$. The total communication complexity is $O(n^2 \log n)$ bits over point-to-point channels; The dealer broadcasts $O(n \log n)$ bits, and each other party broadcasts $O(m \log n)$ bits.*

Proof. Regarding efficiency, by inspection, each party sends or receives $O(n \log n)$ over the point-to-point channels.

Protocol $\Pi_{\mathsf{ShareAttempt}}$ (Protocol 4.2), securely computes the functionality $\mathcal{F}_{\mathsf{ShareAttempt}}$ (Functionality 4.1) with statistical security, in the presence of a malicious adversary controlling at most $t < n/3$ except with probability $\epsilon = n \cdot e^{-m/6}$. The total communication complexity is $O(n^2 \log n)$ bits over point-to-point channels; The dealer broadcasts $O(n \log n)$ bits, and each other party broadcasts $O(m \log n)$ bits.

We now show that conditioned on that Bad does not occur, then the protocol securely computes the functionality $\mathcal{F}_{\mathsf{ShareAttempt}}$ with perfect security. To that end, we construct a simulator considering two cases on the adversary strategy: the dealer is honest at the end of the protocol and the dealer is corrupted by the end of the protocol. Note that the two cases include any post-execution adaptive corruptions made by the adversary, i.e., in the first case, the adversary does not corrupt the dealer even after the execution and in the second case, the adversary may corrupt the dealer after the execution. The simulator maintains a set $\hat{I}$ of adaptively corrupted parties during the simulation and updates it upon receiving a $(\mathsf{corrupt}, i \in [n])$ request from the adaptive adversary.

The case of an honest dealer. The simulator performs the following:

1. Invoke the adversary with an auxiliary input and initialize the sets $\mathsf{CONFLICTS} = \phi$ and $\hat{I} = \phi$.
2. For each corruption $(\mathsf{corrupt}, i)$ requested by the adversary, adaptively corrupt party P_i in the ideal execution and include $i \in \hat{I}$.
3. Receive from the ideal functionality the set ZEROS. Once again, for each corruption $(\mathsf{corrupt}, i)$ requested by the adversary, adaptively corrupt party P_i in the ideal execution and include $i \in \hat{I}$.
4. Receive from the ideal functionality the leaked shares corresponding to the currently corrupted parties, $(f_i(x), g_i(y))_{i \in \hat{I}}$ where $\mathsf{ZEROS} \subseteq \hat{I}$ and $f_i(x) = g_i(y) = 0$ for each $i \in \mathsf{ZEROS}$.
5. Now simulate the steps of the protocol using the values $(f_i(x), g_i(y))_{i \in \hat{I}}$ received from the ideal functionality. Note that in each round of the protocol, the messages the adversary sees from the honest parties can be computed from just the polynomials $(f_i(x), g_i(y))_{i \in \hat{I}}$.

   Patching. After each round r of the protocol simulation, the adaptive adversary may make an adaptive corruption $(\mathsf{corrupt}, i)$. Upon receiving such a request, perform the following:
   (a) Include $i \in \hat{I}$ and adaptively corrupt i in the ideal execution.
   (b) Receive from the ideal functionality the polynomials $f_i(x), g_i(y)$.
   (c) Append to the adversary's view, the messages sent and received until round r by the simulated honest party P_i that received polynomials $f_i(x), g_i(y)$ from the dealer.
6. In the final step of the simulation, include $i \in \mathsf{CONFLICTS}$ if the adversary broadcast $\mathsf{complaint}(i, j, u_i, v_i)$ such that $u_i \neq f_i(j)$ and $v_i \neq g_i(j)$ or if the adversary broadcast $(\mathsf{complaint}, i)$ or if P_i broadcasted more than R complaints. The honest dealer's broadcast is then simulated.
7. Send $\mathsf{CONFLICTS}$ to the ideal functionality. For each adaptive corruption $(\mathsf{corrupt}, i)$ requested by the adversary, execute the same steps as in Step 5 to patch the adversary's view.
8. Receive the outputs from the ideal functionality. If the output is $(\mathsf{detect}, \mathsf{CONFLICTS})$, then send it to the adversary. Else, send $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$ for each $i \in \hat{I} \setminus \mathsf{CONFLICTS}$ and $(\mathsf{proceed}, \perp, \perp, \mathsf{CONFLICTS})$ for each $i \in \mathsf{CONFLICTS}$.
9. At this stage, the adaptive adversary may make further adaptive corruptions with a request $(\mathsf{corrupt}, i)$ for $i \notin \hat{I}$ (these are the post-execution adaptive corruptions and i cannot be the dealer). Once again, the simulator performs the steps as in Step 5 to patch the view of the adversary.

The simulator samples the random complaints from the same distribution as the honest parties in the real execution of the protocol. Additionally, after each adaptive corruption $(\mathsf{corrupt}, i)$ requested by the adversary after round r, the simulator patches (see Step 5) the view of the adversary with the messages sent and received by the simulated honest party P_i until round r. This holds for the patched view for post-execution adaptive corruptions as well. Note that the simulator can execute such a patching as it receives from the ideal functionality the shares of each adaptively corrupted party after corrupting it in the ideal execution. As the adaptive adversary may make up to t adaptive corruptions of parties excluding the dealer (including post-execution) and from Lemma 3.3, the adversary's view in the real execution and the simulated execution must be identically distributed (the rest of the simulation is deterministic). It now suffices to show that the outputs of the honest parties are identical in the real and ideal execution.

All honest parties input $\mathsf{ZEROS} \subseteq I$ and the dealer invokes the ideal functionality with a valid degree-(t, t) bivariate polynomial $S(x, y)$ with $S(x, i) = S(i, y) = 0$ for each $i \in \mathsf{ZEROS}$. Hence, the honest parties never discard the honest dealer and the output is either $(\mathsf{detect}, \mathsf{CONFLICTS})$ or $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$ where $\mathsf{CONFLICTS} \subseteq I$.

In the real execution, the honest dealer sends the valid shares on $S(x, y)$ to the honest parties who will not disagree with each other. Suppose the party P_j stays honest till Step 2c,

$|\mathsf{Complaints}_j| \leq t$ and for each $i \in \mathsf{ZEROS}$ it holds that $f_i(x) = g_i(y) = 0$. Hence, P_j does not broadcast $(\mathsf{complaint}, j)$ and then broadcasts up to $O(m)$ complaints against corrupted parties. Since the complaints are made public, all the parties (including the honest dealer) can identify whether to include the corrupted party in $\mathsf{CONFLICTS}$. The dealer broadcasts $\mathsf{CONFLICTS} \subseteq I$. Note that even after updating $\mathsf{CONFLICTS}$, $\mathsf{CONFLICTS} \subseteq I$ still holds as only corrupted parties are included in $\mathsf{CONFLICTS}$ by the honest parties (in both the simulated execution and the real execution). Furthermore, under these conditions, an honest dealer cannot be discarded as all the complaints are resolved correctly. Since, $\mathsf{CONFLICTS}$ is known to all honest parties (all the required values are broadcast), the honest parties agree on whether to output $(\mathsf{detect}, \mathsf{CONFLICTS})$ or $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$. Note that in this case, the randomness used to choose the complaints changes whether the honest parties output $\mathsf{proceed}$ or detect. Clearly, the outputs are identical to the ideal execution. For the case of an honest dealer, clearly, the event Bad does not occur as all honest parties will definitely agree with each other (also, $\mathsf{CONFLICTS} \subseteq I$).

The case of a corrupted dealer. In this case, the dealer may be corrupted at any stage of the computation, including after the execution. Critically, if the dealer is honest until some step of the simulation and then is corrupted, the prior interactions with the ideal functionality cannot be modified. The simulator performs the following:

1. Invoke the adversary with an auxiliary input and initialize the sets $\mathsf{CONFLICTS} = \phi$ and $\hat{I} = \phi$.
2. For each corruption $(\mathsf{corrupt}, i)$ requested by the adversary, adaptively corrupt party P_i in the ideal execution and include $i \in \hat{I}$.
3. Receive from the ideal functionality the set ZEROS. Once again, for each corruption $(\mathsf{corrupt}, i)$ requested by the adversary, adaptively corrupt party P_i in the ideal execution and include $i \in \hat{I}$.
4. Now simulate the steps of the protocol as the honest parties interacting with the adversary.

   Patching. After each round r of the protocol simulation, the adaptive adversary may make an adaptive corruption $(\mathsf{corrupt}, i)$. Upon receiving such a request, perform the following:
   (a) Include $i \in \hat{I}$ and adaptively corrupt i in the ideal execution.
   (b) Append to the adversary's view, the messages sent and received until round r by the simulated honest party P_i.
5. We consider three cases:
   (a) If some simulated honest party output $\mathsf{discard}$, then send $S(x, y) = y^{t+1}$ to the ideal functionality with $\mathsf{CONFLICTS} = \phi$.
   (b) If some simulated honest party output $(\mathsf{detect}, \mathsf{CONFLICTS})$, it must hold that $|\mathsf{CONFLICTS}| > t/2$ and hence send $S(x, y) = y^t$ with $\mathsf{CONFLICTS}$ to the ideal functionality.
   (c) Else, let J be an arbitrary set of $t + 1$ honest parties that are not in $\mathsf{CONFLICTS}$. Note that since $|\mathsf{CONFLICTS}| \leq t/2$ at least $n - t/2 \geq 5t/2 + 1$ parties are not in $\mathsf{CONFLICTS}$ out of which $\geq 3t/2 + 1 > t + 1$ must be honest. Interpolate a bivariate $S(x, y)$ such that $S(x, j) = f_j(x)$ for each $j \in J$ and send $S(x, y)$ with $\mathsf{CONFLICTS}$ to the ideal functionality.

The simulator only simulates the honest parties in the protocol and patches the adversaries view with the appropriate transcripts of the simulated honest parties for each adaptive corruption. Hence, the view of the adversary must be identically distributed in the real and ideal execution. Recall that we condition the security on the event that Bad does not occur. In the following analysis, we call the honest parties at the end of the execution as just honest parties for brevity. It must hold that either some honest party outputs detect or $\mathsf{discard}$ or the honest parties output $\mathsf{proceed}$ and the honest parties that are not in $\mathsf{CONFLICTS}$ agree with each other.

- **Some honest party outputs detect or $\mathsf{discard}$.** Since the choice to output is based on messages that are broadcast, all honest parties output either $(\mathsf{detect}, \mathsf{CONFLICTS})$

where $|\mathsf{CONFLICTS}| > t/2$ or $\mathsf{discard}$. If the output is $\mathsf{discard}$, the simulator sends input $S(x,y) = y^{t+1}$ to the ideal functionality which results in all honest parties receiving output $\mathsf{discard}$ which is identical to the real execution. Similarly, if one simulated honest party outputs $(\mathsf{detect}, \mathsf{CONFLICTS})$, which is given by the simulator to the ideal functionality with $S(x,y) = y^t$. The ideal functionality gives as output $(\mathsf{detect}, \mathsf{CONFLICTS})$ to all the honest parties which is identical to the real execution.

- **All honest parties output $\mathsf{proceed}$. but honest parties not in $\mathsf{CONFLICTS}$ agree with each other.** Once again, note that the honest parties must agree on the set $\mathsf{CONFLICTS}$ such that $|\mathsf{CONFLICTS}| \leq t/2$ and the honest parties not in $\mathsf{CONFLICTS}$ agree with each other. There are $n - t - t/2 \geq 3t/2 + 1 > t + 1$ honest parties not in $\mathsf{CONFLICTS}$ who agree with each other. Hence, Lemma 3.3 implies that there exists a unique bivariate $S(x,y)$ such that for each honest $i \notin \mathsf{CONFLICTS}$, it holds that $S(x,i) = f_i(x)$ and $S(i,y) = g_i(y)$. The simulator computes exactly this bivariate and gives $S(x,y)$ with $\mathsf{CONFLICTS}$ to the ideal functionality. The ideal functionality gives $i \notin \mathsf{CONFLICTS}$, the output $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$ and $(\mathsf{proceed}, \perp, \perp, \mathsf{CONFLICTS})$ to $i \in \mathsf{CONFLICTS}$. Clearly, the honest parties output the same in the real execution as well.

Hence, if Bad does not occur, the protocol perfectly computes the functionality $\mathcal{F}_{\mathsf{ShareAttempt}}$. From Claim 4.3, we have that Bad occurs with probability at most $\epsilon = n \cdot e^{-m/6}$ which gives us the required result. □

4.2 Reconstructing Shares

In the reconstruction phase, each party $j \in \mathsf{CONFLICTS}$ is able to reconstruct a share $s_j = S(j,0)$. Note that the parties not in $\mathsf{CONFLICTS}$ do not learn their entire share but just s_j. Moreover, this is sufficient to guarantee reconstruction of $S(0,0)$. Looking ahead, one of the following is possible after an execution of Protocol 4.6:

1. The parties in $\mathsf{CONFLICTS}$ successfully reconstruct their shares. In this case, the parties hold a sharing on the polynomial $q(x) = S(x,0)$ (where the secret is $q(0) = S(0,0)$ as required).
2. At most $t/2$ additional conflicts are identified. In this case, the protocol is restarted from the sharing attempt (Protocol 4.2).
3. More than $t/2$ additional conflicts are identified. In this case, the dealer is discarded.

Functionality 4.5: $\mathcal{F}_{\mathsf{rec\text{-}shares}}$: Reconstructing shares

1. **Input:** All honest parties send to the functionality $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ the sets $\mathsf{ZEROS} \subseteq [n]$ and $\mathsf{CONFLICTS} \subseteq [n]$, each honest $j \notin \mathsf{CONFLICTS}$ sends $(f_j(x), g_j(y))$. Let $S(x,y)$ be the unique bivariate polynomial of degree at most t in x and y that satisfies $f_j(x) = S(x,j)$ and $g_j(y) = S(j,y)$ for every $j \notin \mathsf{CONFLICTS}$. Moreover, it holds that $n - |\mathsf{CONFLICTS}| \geq 2t + t/2 + 1$.
2. $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ sends the $(\mathsf{ZEROS}, \mathsf{CONFLICTS}, (S(x,i), S(i,y))_{i \in I})$ to the adversary. If the dealer is corrupted, $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ sends $S(x,y)$ as well.
3. It receives back from the adversary a message M.
4. **Output:**
   (a) If $M = \mathsf{discard}$ and the dealer is corrupted, then $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ sends $\mathsf{discard}$ to all parties.
   (b) If $M = (\mathsf{detect}, \mathsf{Bad})$ with $\mathsf{Bad} \cap (\mathsf{ZEROS} \cup \mathsf{CONFLICTS}) = \phi$ and $|\mathsf{Bad}| > t/2$, and with $\mathsf{Bad} \subseteq I$ in the case of an honest dealer, then $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ sends $(\mathsf{detect}, \mathsf{Bad})$ to all the parties.
   (c) If $M = \mathsf{proceed}$, then $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ sends $(\mathsf{proceed}, S(j,0))$ to each party P_j.

Protocol 4.6: $\Pi_{\mathsf{rec\text{-}shares}}$: A Protocol for Reconstructing the shares

Input: All parties hold the same set $\mathsf{CONFLICTS}$ and ZEROS. Each honest party not in

CONFLICTS holds a pair of polynomials $(f_i(x), g_i(y))$, and it is guaranteed that all the shares of honest parties lie on the same bivariate polynomial $S(x, y)$ with degree at most t in x and y. Without loss of generality, let CONFLICTS $= \{1, \ldots, c\}$ where $c \leq t/2$.

The protocol:

1. Every party sets HAVE-SHARES $= [n] \setminus (\text{ZEROS} \cup \text{CONFLICTS})$.
2. **Computing $f_i^V(x)$:**
   (a) Each party $P_i \notin$ CONFLICTS computes the degree c polynomial:
   $$f_i^V(x) := \sum_{k=1}^{c} x^{k-1} \cdot f_i(k) \ .$$
   It sends to each party P_j for $j \notin$ CONFLICTS the value $u_{i \to j} = f_i^V(j)$.
3. **Computing $g_i^V(y)$:**
   (a) Let $u_{j \to i}$ be the value that P_j sent P_i in the previous step (and take $u_{j \to i} = 0$ for $j \in$ ZEROS). P_i attempts to find a unique polynomial $g_i^V(y)$ of degree at most t satisfying $g_i^V(j) = u_{j \to i}$ for every $j \notin$ CONFLICTS, with at most $t/2$ errors.
   (b) If there is no unique reconstruction, then P_i broadcasts complaint(i).
   (c) If there is a unique reconstruction, then P_i broadcasts reveal$(i, g_i^V(0))$.
4. **Dealer – Complaint Resolution**
   (a) The dealer sets Bad $= \phi$. For each complaint(i) broadcasted by P_i, add i to Bad.
   (b) For every reveal(i, u_i) broadcasted by P_i, the dealer checks if
   $$u_i = \sum_{k=1}^{c} i^{k-1} \cdot S(k, 0)$$
   If not, then it adds i to Bad. The dealer broadcasts Bad.
5. **Output:**
   (a) **Discard:** discard the dealer if any one of the following holds: (1) There exists P_i that broadcasted complaint(i) but $i \notin$ Bad; (2) If ZEROS $\cap$ Bad $\neq \emptyset$; (3) $|$ZEROS $\cup$ CONFLICTS$\cup$Bad$| > t$; (4) If the set of points $K = \{(j, u_j)\}_{j \notin (\text{CONFLICTS} \cup \text{Bad} \cup \text{ZEROS})} \cup \{(j, 0)\}_{j \in \text{ZEROS}}$ not all lie on a univariate polynomial $f_0^V(y)$ of degree c.
   If any one of the above holds, output discard.
   (b) **Large detection:** Otherwise, if $|$Bad$| > t/2$ then output (detect, Bad).
   (c) **Proceed:** Otherwise, if P_i for $i \notin$ CONFLICTS outputs $g_i(0)$. P_i for $i \in$ CONFLICTS reconstructs $f_0^V(x)$ from the set of points K as above, and outputs the coefficient of the x^{i-1} term.

We now prove the security of Protocol 4.6 with the following theorem:

Theorem 4.7. $\Pi_{\text{rec-shares}}$ *(Protocol 4.6) securely computes the functionality* $\mathcal{F}_{\text{rec-shares}}$ *Functionality 4.5) with perfect security, in the presence of a malicious adversary controlling at most* $t < n/3$ *parties. The total communication complexity is* $O(n^2 \log n)$ *bits over point-to-point channels; The dealer broadcasts* $O(n \log n)$ *bits and each other party broadcasts* $O(\log n)$ *bits.*

Proof. We consider two cases.

The case of an honest dealer. The simulator executes the following steps:

1. Invokes the adversary with the auxiliary input z.
2. Receives ZEROS, CONFLICTS and the shares of the corrupted parties, that is, $S(x, i), S(i, y)$ for each $i \in I$.
3. Simulate the execution of the protocol steps according to the messages sent by the adversary. Since CONFLICTS $\subseteq I$ (as the dealer is honest), the messages of the honest parties to the corrupted parties can be computed from $S(x, i), S(i, y)$ for $i \in I$. Specifically,

(a) Corresponding to each honest $j \notin \mathsf{CONFLICTS}$ and corrupted $i \notin \mathsf{CONFLICTS}$, the simulator sends the message $u_{j\to i}$ as in Step 2a, where

$$u_{j\to i} = f_j^V(i) = \sum_{k=1}^{c} x^{k-1} \cdot f_j(k) = \sum_{k=1}^{c} x^{k-1} \cdot S(k,j)$$

The simulator can compute each of the values $u_{j\to i}$, since it holds the polynomial $S(k,y)$ for every $k \in \mathsf{CONFLICTS} \subseteq I$.

(b) The simulator simulates the broadcast of $\mathsf{reveal}(j, g_j^V(0))$ in Step 3c on behalf of each honest party $j \notin \mathsf{CONFLICTS}$ where

$$g_j^V(0) = \sum_{k=1}^{c} j^{k-1} \cdot S(k,0)$$

which can be computed from $S(k,y)$ for each $k \in I \cap \mathsf{CONFLICTS}$.

(c) The simulator can compute the set Bad based on the messages of the corrupted parties: include corrupted $i \in \mathsf{Bad}$ if P_i broadcasted $\mathsf{complaint}(i)$ or if P_i broadcasted $\mathsf{reveal}(i,u_i)$ where $u_i \neq \sum_{k=1}^{c} i^{k-1} \cdot S(k,0)$ which can be computed from $S(k,y)$ for each $k \in I \cap \mathsf{CONFLICTS}$. The simulator now simulates the dealer's broadcast of Bad.

(d) If $|\mathsf{Bad}| > t/2$, then the simulator sends $(\mathsf{detect}, \mathsf{Bad})$ to the ideal functionality. Else, it sends $\mathsf{proceed}$.

Since the simulator simulates exactly the protocol and the protocol is deterministic, the adversary's view is distributed identically in the real and ideal executions. It now suffices to argue that the honest parties output identically in the real and ideal executions. For an honest dealer, all the honest parties will be able to interpolate the polynomial for the parties in $\mathsf{CONFLICTS} \subset I$. Hence, $\mathsf{Bad} \subseteq I$ and the dealer is never discarded as the simulator follows the steps of the protocol (as does an honest dealer). If the honest parties output $(\mathsf{detect}, \mathsf{Bad} \subseteq I)$ they must agree in the real execution due to the properties of broadcast. Else, the honest parties will definitely output $\mathsf{proceed}$ with the valid shares. Clearly, the outputs of the honest parties are identical in the real and ideal executions.

The case of a corrupted dealer. The simulator executes the following:

1. Invokes the adversary with the auxiliary input z.
2. Receives ZEROS, $\mathsf{CONFLICTS}$ and the bivariate $S(x,y)$.
3. Simulate the execution of the protocol steps according to $S(x,y)$ and the adversary's messages. Send M to the ideal functionality according to the following cases:
   (a) If the output of some simulated honest party is $\mathsf{discard}$ then send $\mathsf{discard}$.
   (b) If the output of some simulated honest party is $(\mathsf{detect}, \mathsf{Bad})$ then send $(\mathsf{detect}, \mathsf{Bad})$.
   (c) If the output of some simulated honest party is $\mathsf{proceed}$, then send $\mathsf{proceed}$.

Once again, the simulator executes the same steps as in the protocol. Hence, the adversary's view is identical in the real and ideal execution. It suffices to show that the honest party's outputs are identical in the two executions. Once again, from the properties of broadcast, it must hold that if one honest party outputs either detect or $\mathsf{discard}$, the honest parties agree on the output, and therefore, the outputs of the honest parties are identical in both executions. Suppose that all honest parties decide to $\mathsf{proceed}$. We claim that in the real world, each honest $i \notin \mathsf{CONFLICTS}$ outputs $g_j(0) = S(j,0)$. Each honest $i \notin \mathsf{CONFLICTS}$ sends $u_{i\to j} = \sum_{k=1}^{c} j^{k-1} \cdot f_i(k) = \sum_{k=1}^{c} j^{k-1} \cdot S(k,i)$. Hence, each $j \notin \mathsf{CONFLICTS}$ receives a codeword of size at least $n - t/2$. There are at least $n - t/2 - t \geq 3t/2 + 1$ correct values received from the honest parties. Therefore, if an honest party P_j does not broadcast $\mathsf{complaint}(j)$, then P_j must broadcast $\mathsf{reveal}(j, u_j)$ where $u_j = g_j^V(0) = \sum_{k=1}^{c} j^{k-1} \cdot S(k,0)$. Since there are at most $t/2$ parties in Bad and the set K defines a

unique polynomial $f_0^V(x) = \sum_{k=1}^{c} x^{k-1} \cdot S(k,0)$ (else, the dealer would have been discarded), each honest party $i \in \mathsf{CONFLICTS}$ outputs the coefficient of the x^{i-1} term, that is, $S(i,0)$. Clearly, the outputs of the honest parties in the real and ideal executions, completing our proof. □

4.3 Statistical VSS Protocol

In our VSS scheme, the parties can only compute their share of the dealer's secret but not the entire row and column polynomials. Hence, the VSS functionality from [AAPP23] must be modified. However, the changes are minute and our protocol also follows the same mechanism as the VSS protocol from [AAPP23].

We first present the modified functionality $\mathcal{F}_{\mathsf{VSS}}$ and then present our VSS protocol Π_{VSS} in the $(\mathcal{F}_{\mathsf{ShareAttempt}}, \mathcal{F}_{\mathsf{rec\text{-}shares}})$-hybrid model.

Functionality 4.8: $\mathcal{F}_{\mathsf{VSS}}$

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

1. **Input:** All the parties send to $\mathcal{F}_{\mathsf{VSS}}$ a set $\mathsf{ZEROS} \subseteq [n]$ such that $|\mathsf{ZEROS}| \leq t$. For an honest dealer, it holds that $\mathsf{ZEROS} \subseteq I$.
2. **Honest Dealer:** The dealer sends $s \in \mathbb{F}$ to $\mathcal{F}_{\mathsf{VSS}}$. The functionality sends ZEROS to the adversary who replies with $(f_i(x), g_i(y))_{i \in I}$ under the constraint that $f_i(x) = g_i(y) = 0$ for each $i \in \mathsf{ZEROS}$. The functionality chooses a random bivariate polynomial $S(x,y)$ of degree t in x and y under the constraints that (i) $s = S(0,0)$; (ii) $S(x,i) = f_i(x)$ and $S(i,y) = g_i(y)$ for each $i \in I$.
3. **Corrupted dealer:** The functionality sends ZEROS to the adversary, which responds with $S(x,y)$. $\mathcal{F}_{\mathsf{VSS}}$ verifies that $S(x,y)$ is of degree t in x and y, and that for every $i \in \mathsf{ZEROS}$ it holds that $S(x,i) = S(i,y) = 0$. If not, $\mathcal{F}_{\mathsf{VSS}}$ replaces $S(x,y) = \perp$.
4. **Output:** $\mathcal{F}_{\mathsf{VSS}}$ sends to each party P_j the share $s_j = S(j,0)$.

Protocol 4.9: Secret sharing in the $(\mathcal{F}_{\mathsf{ShareAttempt}}, \mathcal{F}_{\mathsf{rec\text{-}shares}})$-hybrid model – Π_{VSS}

The parties initially set $\mathsf{ZEROS} = \phi$. The dealer holds as input $s \in \mathbb{F}$.
The protocol:

1. **Dealing the shares.**
   (a) The dealer chooses a random bivariate polynomial $S(x,y)$ of degree t in x and y such that $s = S(0,0)$ and $S(x,i) = S(i,y) = 0$ for each $i \in \mathsf{ZEROS}$.
   (b) The parties invoke Functionality 4.1, $\mathcal{F}_{\mathsf{ShareAttempt}}$, where the dealer inputs $S(x,y)$ and all the parties input ZEROS:
      i. If the output is $\mathsf{discard}$, then proceed to Step 3a.
      ii. If the output is $(\mathsf{detect}, \mathsf{CONFLICTS})$ then set $\mathsf{ZEROS} = \mathsf{ZEROS} \cup \mathsf{CONFLICTS}$. If $|\mathsf{ZEROS}| > t$ then proceed to Step 3a. Else, repeat from Step 1a.
      iii. If the output is $(\mathsf{proceed}, f_i(x), g_i(y), \mathsf{CONFLICTS})$ with $|\mathsf{CONFLICTS}| \leq t/2$ and for $i \in \mathsf{CONFLICTS}$ $f_i(x) = g_i(y) = \perp$, then proceed to the next step.
2. **Reconstruct the shares:** The parties invoke Functionality 4.5, $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ with inputs $(\mathsf{ZEROS}, \mathsf{CONFLICTS}, f_i(x), g_i(y))$.
   (a) If the output is $\mathsf{discard}$, then proceed to Step 3a.
   (b) If the output is $(\mathsf{detect}, \mathsf{Bad})$ then set $\mathsf{ZEROS} = \mathsf{ZEROS} \cup \mathsf{Bad}$. If $|\mathsf{ZEROS}| > t$ then proceed to Step 3a. Else, repeat from Step 1a.
   (c) If the output is $(\mathsf{proceed}, g_i(0), \mathsf{CONFLICTS})$, then proceed to Step 3b.
3. **Output:**
   (a) **Discard:** All parties output $\perp$.

(b) **Successful:** Output $s_i = g_i(0)$.

Theorem 4.10. *Protocol Π_{VSS} (Protocol 4.9), perfectly securely computes the functionality $\mathcal{F}_{\mathsf{VSS}}$ (Functionality 4.8), in the $(\mathcal{F}_{\mathsf{ShareAttempt}}, \mathcal{F}_{\mathsf{rec\text{-}shares}})$-hybrid model (Functionality 4.1,4.5), in the presence of a malicious adaptive adversary controlling at most $t < n/3$.*

Proof. There are two differences between Protocol 4.9 and the packed secret sharing of [AAPP23]

1. In Protocol 4.9 the parties do not start with ZEROS as input and the dealer uses a degree-(t,t) bivariate polynomial instead of a degree-$(3t/2, t + t/4)$ bivariate polynomial.
2. In Protocol 4.9 the parties output only the share $s_i = g_i(0)$ computed from the polynomials received from the last call to $\mathcal{F}_{\mathsf{ShareAttempt}}$. The properties of $\mathcal{F}_{\mathsf{rec\text{-}shares}}$ guarantee that the parties output $s_i = g_i(0) = S(i, 0)$ where the dealer chooses the degree-(t,t) bivariate $S(x, y)$.

The proof from [AAPP23] can be used with minor changes to reflect the above differences. □

Statistical security after composition. Protocol Π_{VSS} in the $(\mathcal{F}_{\mathsf{ShareAttempt}}, \mathcal{F}_{\mathsf{rec\text{-}shares}})$-hybrid model perfectly realizes $\mathcal{F}_{\mathsf{VSS}}$. However, the protocol $\Pi_{\mathsf{ShareAttempt}}$ (Protocol 4.2) realizes $\mathcal{F}_{\mathsf{ShareAttempt}}$ (Functionality 4.1) with statistical security in the case of a corrupted dealer. Since we proved the **adaptive** security of $\Pi_{\mathsf{ShareAttempt}}$, protocol Π_{VSS} is also **adaptively** secure. As described in Section 3.1, we can state the following corollary:

Corollary 4.11. *There exists a protocol that statistically realizes the functionality $\mathcal{F}_{\mathsf{VSS}}$ against $t < n/3$* **adaptive** *corruptions such that the statistical error for the protocol is $\epsilon < 2n \cdot e^{-m/6}$.*

Note that in each invocation to Π_{VSS}, there is a statistical error of at most $n \cdot e^{-m/6}$ and due to the repetitions the number of invocations to $\mathcal{F}_{\mathsf{ShareAttempt}}$ is at most 2. From the definition of statistical security, it holds that the statistical error for the composed sharing protocol is at most $2n \cdot e^{-m/6}$.

4.4 Batched Verifiable Secret Sharing

The protocol Π_{VSS} (Protocol 4.9) incurs a communication complexity of $O(n^2 \log n)$ bits over the point-to-point channels. Additionally, each party broadcasts $O(m \log n)$ bits ($= \mathsf{poly} \log n$) and the dealer broadcasts $O(n \log n)$ bits. Towards reducing the cost while running multiple instances of Π_{VSS}, we batch the broadcast costs among multiple instances. We first present the batched VSS ideal functionality $\mathcal{F}_{\mathsf{bVSS}}$, before discussing our techniques for batching.

Functionality 4.12: $\mathcal{F}_{\mathsf{bVSS}}$

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

1. **Input:** All the parties send to $\mathcal{F}_{\mathsf{bVSS}}$ a set ZEROS $\subseteq [n]$ such that $|\mathsf{ZEROS}| \leq t$. For an honest dealer, it holds that ZEROS $\subseteq I$.
2. **Honest Dealer:** The dealer sends $s^{(1)}, \ldots, s^{(L)} \in \mathbb{F}$ to $\mathcal{F}_{\mathsf{bVSS}}$. The functionality sends ZEROS to the adversary who replies with $(f_i^{(\ell)}(x), g_i^{(\ell)}(y))_{i \in I}$ for $\ell = 1, \ldots, L$ under the constraint that $f_i^{(\ell)}(x) = g_i^{(\ell)}(y) = 0$ for each $i \in$ ZEROS. The functionality chooses L random bivariate polynomial $S^{(1)}(x, y), \ldots, S^{(L)}(x, y)$ of degree t in x and y under the constraints that (i) $s^{(\ell)} = S^{(\ell)}(0, 0)$; (ii) $S^{(\ell)}(x, i) = f_i^{(\ell)}(x)$ and $S^{(\ell)}(i, y) = g_i^{(\ell)}(y)$ for each $i \in I$ and each $\ell = 1, \ldots, L$.
3. **Corrupted dealer:** The functionality sends ZEROS to the adversary, which responds with $S^{(1)}(x, y), \ldots, S^{(L)}(x, y)$. $\mathcal{F}_{\mathsf{bVSS}}$ verifies that for each $\ell = 1, \ldots, L$ it holds that $S^{(\ell)}(x, y)$ is of degree t in x and y, and that for every $i \in$ ZEROS it holds that $S^{(\ell)}(x, i) = S^{(\ell)}(i, y) = 0$. If not, $\mathcal{F}_{\mathsf{bVSS}}$ replaces $S^{(\ell)}(x, y) = \perp$ for each $\ell = 1, \ldots, L$.

4. **Output:** $\mathcal{F}_{\mathsf{bVSS}}$ sends to each party P_j the share $s_j^{(\ell)} = S^{(\ell)}(j, 0)$ for each $\ell = 1, \ldots, L$.

---

1. **Batching broadcast in the sharing attempt.** The dealer inputs L bivariate polynomials, and the same set ZEROS is used across all instances, that is, each bivariate has zero shares for parties in the set ZEROS. The parties chooses R elements and broadcasts complaints with respect to one of the bivariates where there was a conflict. The complaint is now of the form $(\mathsf{complaint}, i, j, \ell, f_i^\ell(j), g_i^\ell(j))$ where $S^\ell(x, y)$ is the ℓ^{th} bivariate. The parties confirm the random complaints as before with ℓ included in the complaint. The dealer then computes a single CONFLICTS set after checking the complaints with each bivariate. As discussed in [AAPP23], a joint complaint in some instance (say the lexicographically smallest index) must be resolved by the dealer and consistency must hold across all bivariates.
2. **Batching dealer's broadcast in the share reconstruction.** We note that our share reconstruction protocol already batches the reconstruction of the shares of parties in CONFLICTS by embedding all the shares of parties in CONFLICTS in a polynomial $u(y)$ and robustly reconstructing $u(0)$. Thus, our protocol already performs a batching of the reconstruction of $|\mathsf{CONFLICTS}| = O(n)$ values and it is unclear how the same technique can be extended to the reconstruction of $O(nL)$ values. However, the dealer computes a single Bad set by checking the parties' broadcasted values with each bivariate. After removing the parties in the dealer's broadcasted Bad set, the broadcasted values must determine a unique polynomial for each instance (else, the dealer is discarded). Hence, all the reconstructed shares must be correct on each bivariate. The dealer broadcasts $O(n \log n)$ bits however each party broadcasts $O(m \log n)$ bits.

As discussed above, we perform batching primarily to allow the dealer to broadcast a common set CONFLICTS in the sharing attempt (see Protocol 4.2) and a common set Bad in the share reconstruction (see Protocol 4.6). Note that this idea is identical to the batching idea in [AAPP23] and their proof carries over directly in our case. Hence, we state the following corollary:

Corollary 4.13. *Protocol Π_{bVSS} securely computes $\mathcal{F}_{\mathsf{bVSS}}$ (Functionality 4.12), in the presence of a malicious* **adaptive** *adversary controlling at most $t < n/3$. The communication complexity of Π_{bVSS} is $O(Ln^2 \log n)$ bits on the point-to-point channels, the dealer broadcasts $O(n \log n)$ bits and each party broadcasts $O(m \log n)$ bits. The protocol is statistically secure, with statistical error $\epsilon < 2n \cdot e^{-m/6}$.*

Efficiency: The cost of the batched VSS is $O(Ln^2 \log n)$ bits on the point-to-point channels, the dealer broadcasts $O(n \log n)$ bits and each party broadcasts $O(m \log n)$ bits. Looking ahead, we will set $m = \mathsf{poly} \log n$ and $L = m$ which allows us to achieve the reduced cost for oblivious leader election.

5 Multi-Moderated Verifiable Secret Sharing

As mentioned in the overview (Section 2.1), we replace the broadcast in the VSS with a gradecast [ZLC23]. However, we need the help of moderators to make this substitution more robust. We use all parties as moderators. Corresponding to each moderator M_j, each party P_k holds a pair of flags $v_{M_j}^k$ and $\mathsf{d}_{M_j}^k$:

- The flag $v_{M_j}^k \in \{0, 1\}$ indicates if P_k believes that M_j *moderated* the broadcasts correctly, that is, if the parties agree on the outputs of the moderated broadcasts. If at least one honest party P_k holds $v_{M_j}^k = 1$, then the moderated broadcast of M_j are identical to the actual broadcast. If M_j is honest, then each honest party P_k holds $v_{M_j}^k = 1$.

- The flag $\mathsf{d}^k_{M_j} \in \{0,1\}$ indicates if P_k accepts the dealer's share based on the simulated broadcast messages of M_j.

We first present the ideal functionality formalizing the above properties.

Functionality 5.1: $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$

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

1. **Input:** The dealer holds as input a secret $s \in \mathbb{F}$.
2. **Honest Dealer:** The dealer sends s to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$. The adversary sends $(f_i(x), g_i(y))_{i\in I}$ to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$. The functionality chooses a random bivariate $S(x,y)$ of degree t in x and y such that (i) $s = S(0,0)$; (ii) $S(x,i) = f_i(x)$ and $S(i,y) = g_i(y)$ for each $i \in I$.
3. **Corrupted dealer:** The adversary sends $S(x,y)$ to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$. $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ verifies that $S(x,y)$ is of degree t in x and y. If not, the functionality sets $S(x,y) = \perp$.
4. **Moderators:** For each party $M_j \in [n]$ acting as a moderator:
   (a) If the moderator M_j is honest, then set $v^k_{M_j} = 1$ for each $k \in [n]$. Furthermore,
      i. If the dealer is honest, then set $\mathsf{d}^k_{M_j} = 1$ for each $k \in [n]$.
      ii. If the dealer is corrupted, then set $\mathsf{d}^k_{M_j} = 1$ for each $k \in [n]$ if and only if the ideal functionality sets $S(x,y) \neq \perp$ in Step 3.
   (b) If the moderator M_j is corrupted, then receive a message m_j from the adversary.
      i. If $m_j = (\mathsf{Agreement}, (\hat{v}^k_{M_j})_{k\notin I}, \mathsf{d}_{M_j})$ where $\mathsf{d}_{M_j} \in \{0,1\}$, and for some $k \notin I$ it holds that $v^k_{M_j} = 1$, then for every $k \notin I$ set $v^k_{M_j} = \hat{v}^k_{M_j}$, as received from the adversary. If the ideal functionality sets $S(x,y) \neq \perp$ in Step 3, then set $\mathsf{d}^k_{M_j} = \mathsf{d}_{M_j}$ for every $k \notin I$. Else, set $\mathsf{d}^k_{M_j} = 0$ for every $k \notin I$.
      ii. If $m_j = (\mathsf{NoAgreement}, (\hat{\mathsf{d}}^k_{M_j})_{k\notin I})$ where each $\mathsf{d}^k_{M_j} \in \{0,1\}$, then set $v^k_{M_j} = 0$ for every $k \in [n]$ and $(\mathsf{d}^1_{M_j}, \ldots, \mathsf{d}^n_{M_j}) = (\hat{\mathsf{d}}^1_{M_j}, \ldots, \hat{\mathsf{d}}^n_{M_j})$ as received from the adversary.
5. **Output:** $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ sends to each party P_j the output $S(j,0)$ and the flags $(v^j_M, \mathsf{d}^j_M)_{M\in[n]}$.

Our multi-moderated VSS protocol below is exactly the same as our VSS protocol with the broadcasts carefully replaced by simulated broadcasts. Until the output has to be computed, the dealer acts as the moderator for the broadcasts of the parties, that is, for each message m that needs to be broadcasted by a party, the party gradecasts m followed by the dealer's gradecast of the same message. Once the protocol reaches the output stage, we artificially add an additional round of public voting by the parties which is then moderated by every other party acting as a moderator. This additional round allows us to compress the size of the messages that the different n parties have to moderate.

Each party gradecasts the decision on the dealer (accept or reject). Each party acting as a moderator, then gradecasts the decision d on the dealer and a set of parties Bad such that:

1. $\mathsf{d} = 1$ if and only if $t+1$ at least accept votes were received with grade 2.
2. $k \in \mathsf{Bad}$ if and only if P_k's vote was received with a grade < 2.

The flags for each moderator are then set by the parties based on the grades in the gradecast decisions and $(\mathsf{d}, \mathsf{Bad})$. Specifically, an honest party sets a high grade for a moderator if and only if any party that sent a vote with grade 0 is included in Bad and the decision d is consistent with the votes of parties not in Bad.

Protocol 5.2: $\Pi_{\mathsf{mm\text{-}VSS}}$

Initialization: Each party P_i sets a happy bit $\mathsf{happy}_i = 1$.

The parties run Protocol 4.9 with the following modifications:

Simulating broadcast in Π_{VSS}: Simulating broadcast of a party P_j

1. **Party P_j:** When P_j wishes to broadcast a message m, it first gradecasts it.
2. **The dealer:** Let (m, g) be the message and g its associated grade. The dealer gradecasts m.
3. **Each party P_i:** Let (m', g') be the message gradecasted by the dealer. Use m' as the message broadcasted by P_j in the protocol. Moreover, if $g' \neq 2$; or if $g = 2$ but $m' \neq m$, then P_i sets $\mathsf{happy}_i = 0$.

Instead of Step 3 in Π_{VSS} — Voting:

1. **Each party P_i:** If $\mathsf{happy}_i = 1$ and the decision in Π_{VSS} is to accept the dealer (non-$\perp$ shares), then gradecast accept; else gradecast reject. Let s_i denote the share output in Π_{VSS}.
   At this point, we fork into n executions, one per moderator $M_j \in [n]$ as follows:
2. **The moderator M_j:**
   (a) Let $(a_1, \ldots, a_n)$ be the decisions of all parties as received from the n gradecasts.
   (b) Initialize $\mathsf{Bad}_{M_j} = \phi$ and include k in Bad_{M_j} if a_k is received from P_k with grade < 2.
   (c) If at least $t+1$ accepts in $(a_k)_{k \notin \mathsf{Bad}_{M_j}}$, i.e., at least $t+1$ accepts that were received with grade 2, set $\mathsf{d}_{M_j} = 1$. Else, set $\mathsf{d}_{M_j} = 0$.
   (d) Gradecast $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$.
3. **Each party P_i:**
   (a) Let $(a_1, \ldots, a_n)$ be the decisions of all parties as received from the gradecasts in Step 1.
   (b) For every moderator $M_j \in [n]$:
      i. Let $(\mathsf{d}'_{M_j}, \mathsf{Bad}'_{M_j})$ be the message gradecasted by the moderator M_j with associated grade g' from Step 2d.
      ii. Set $\mathsf{d}^i_{M_j} = \mathsf{d}'_M$ as gradecasted, and decide on $v^i_{M_j}$:
         A. Set $\mathsf{expectedD}^i_{M_j} = 1$ if and only if there are at least $t+1$ accept's in $(a_k)_{k \notin \mathsf{Bad}'_{M_j}}$ were received with grade ≥ 1.
         B. Set $v^i_{M_j} = 1$ iff all of the following hold: (i) $g' = 2$; and (ii) $\mathsf{expectedD}^i_{M_j} = \mathsf{d}'_{M_j}$; and (iii) For every k such that a_k was received with grade 0 it holds that $k \in \mathsf{Bad}'_{M_j}$; and (iv) $|\mathsf{Bad}'_{M_j}| \leq t$.
4. **Output:** P_i outputs $s_i, (\mathsf{d}^i_{M_1}, \ldots, \mathsf{d}^i_{M_n})$ and $(v^i_{M_1}, \ldots, v^i_{M_n})$.

Theorem 5.3. *Protocol $\Pi_{\mathsf{mm\text{-}VSS}}$ (Protocol 5.2), perfectly securely computes the functionality $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ (Functionality 5.1), in the presence of a malicious adversary controlling at most $t < n/3$ except with probability $\epsilon < 2n \cdot e^{-m/6}$.*

Proof. We consider two cases for the simulator:
The case of an honest dealer. Let $\mathcal{S}_{\mathsf{VSS}}$ be the simulator of Π_{VSS} (Protocol 4.9). The simulator for $\Pi_{\mathsf{mm\text{-}VSS}}$, denoted as $\mathcal{S}_{\mathsf{mm\text{-}VSS}}$ executes the same steps as $\mathcal{S}_{\mathsf{VSS}}$ with the following modifications:

1. **Replacing Broadcasts by Simulated Gradecasts (Functionality 5.8)**:
   First, note that $\mathcal{S}_{\mathsf{VSS}}$ must simulate the broadcasts by the honest parties in Π_{VSS} (including the honest dealer). Let P_j be an honest party that broadcasts a message m in Π_{VSS}. $\mathcal{S}_{\mathsf{mm\text{-}VSS}}$ executes the following instead of simulating a broadcast of m by P_j:
   (a) Simulate the messages of $\Pi_{\mathsf{Gradecast}}$ where P_j gradecasts m.
   (b) Simulate the messages of $\Pi_{\mathsf{Gradecast}}$ where the dealer gradecasts the same message m.

   Consider a corrupted party P_i that broadcasts a message m in Π_{VSS}. In $\Pi_{\mathsf{mm\text{-}VSS}}$, the corrupted party gradecasts m and the honest dealer gradecasts the message m' it received. $\mathcal{S}_{\mathsf{mm\text{-}VSS}}$ must also simulate this as follows:

(a) Simulate the messages of the honest parties in $\Pi_{\mathsf{Gradecast}}$ where P_i gradecasts m.

(b) Let (m', g') be the message and associated grade received by the simulated honest dealer. Simulate the messages of $\Pi_{\mathsf{Gradecast}}$ where the dealer gradecasts the message m'.

2. **Simulating the Moderation**:

(a) Compute the happy bits for each simulated honest party and simulate the messages of $\Pi_{\mathsf{Gradecast}}$ where each honest party P_k gradecasts accept if and only if the simulated honest party P_k held $\mathsf{happy}_k = 1$.

(b) Simulate the messages of the honest parties in $\Pi_{\mathsf{Gradecast}}$ where each corrupted party P_i gradecasts some decision a_i.

(c) For each honest moderator M_j, compute d_{M_j} and the set Bad_{M_j} as in the protocol and simulate the messages of $\Pi_{\mathsf{Gradecast}}$ where M_j sends the message $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$.

(d) For each corrupted moderator M_j, simulate the messages of the honest parties in $\Pi_{\mathsf{Gradecast}}$ where M_j gradecasts $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$. Compute the outputs of each simulated honest party P_k as per $\Pi_{\mathsf{mm\text{-}VSS}}$ denoted as $v^k_{M_j}, \mathsf{d}^k_{M_j}$.

i. If for some simulated honest party P_ℓ it holds that $v^\ell_{M_j} = 1$, then send $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k\notin I}, \mathsf{d}^\ell_{M_j})$ to the ideal functionality.

ii. Else, send $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k\notin I})$ to the ideal functionality.

By inspection, and from the fact that the changes to the Π_{VSS} protocol are deterministic (and the same are the corresponding changes in the simulation), the view generated by the simulator is identical to the real execution of $\Pi_{\mathsf{mm\text{-}VSS}}$. Hence, from the above and the security of Π_{VSS} (Theorem 4.10) we have that the view generated by $\mathcal{S}_{\mathsf{mm\text{-}VSS}}$ is identical to a real execution of $\Pi_{\mathsf{mm\text{-}VSS}}$. We now claim that the outputs are identical in the real and simulated execution. From the security of Π_{VSS}, we already know that the shares output by the honest parties must be identical in the real and simulated executions. It now suffices to show that the flags v, d for each moderator are identical.

Consider a party P_j that broadcasts m in Π_{VSS} and hence gradecasts m in $\Pi_{\mathsf{mm\text{-}VSS}}$. The honest dealer will gradecast whichever message m' was received in the gradecast of P_j by the dealer. All honest parties will take the dealer's gradecast of some m' with grade 2 as P_j's broadcasted message. When P_j is honest, it must hold that $m' = m$. Furthermore, the honest dealer gradecasts each message with grade 2 to all honest parties. Hence, even if an honest party P_k received a different message in the dealers gradecast and P_j's gradecast, it will set $\mathsf{happy}_k = 1$. Clearly, each honest party P_k must have gradecasted accept. For each party $M_j \in [n]$ acting as a moderator, we consider the following cases to show that the outputs are identical in the simulated execution and the real execution:

1. If the moderator M_j is honest, then all honest parties receive $\mathsf{d}^k_{M_j} = 1$ and $v^k_{M_j} = 1$. In the real execution, the honest moderator M_j will gradecast $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$ with grade 2 to each of the honest parties such that $\mathsf{Bad} \subseteq I$ and $\mathsf{d}_{M_j} = 1$ as all honest parties must have gradecasted accept with grade 2 (there are $n - t > t + 1$ honest parties). Hence, each honest party P_k will set $v^k_{M_j} = 1$ and $\mathsf{d}^k_{M_j} = \mathsf{d}_{M_j} = 1$.

2. If M_j is corrupted and some simulated honest party P_ℓ held $v^\ell_{M_j} = 1$, then the simulator looks at the outputs of the simulated honest parties, and extracts from them the values $(v^k_{M_j})_{k\notin I}$. Then, it sends to the trusted party $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k\notin I}, \mathsf{d}^\ell_{M_j})$. The trusted party sends to each honest party P_k the values $\mathsf{d}^\ell_{M_j}$ and $v^k_{M_j}$.
We claim that the output of all honest parties is the same as in the real execution. Specifically, the simulator sends $v^k_{M_j}$ as the outputs that were generated in the simulated execution. The only difference is that the trusted party sets $\mathsf{d}^k_{M_j} = \mathsf{d}^\ell_{M_j}$ for all honest parties. Since the simulated honest party P_ℓ outputs $v^\ell_{M_j} = 1$, it must have received $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$ from the moderator with grade 2. Since all honest parties output the value $\mathsf{d}^k_{M_j}$ as received from the gradecasted message, and all honest parties received the exact same message, we

have that $\mathsf{d}^k_{M_j} = \mathsf{d}^\ell_{M_j}$.

3. If M_j is corrupted and all simulated honest parties hold $v^k_{M_j} = 0$ at the end of the simulated execution, then the simulator sends $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k \notin I})$ to the trusted party, where $\mathsf{d}^k_{M_j}$ are the outputs of the simulated honest parties in the simulated execution. The outputs of the honest parties in the ideal execution are $v^k_{M_j} = 0$ and $\mathsf{d}^k_{M_j}$ as determined by the simulator.
   In the real, the outputs are clearly the same – all honest parties output $v^k_{M_j} = 0$ (as the simulated honest parties in the ideal execution), and the values $\mathsf{d}^k_{M_j}$ that were sent to the trusted party were also determined by the output of the simulated honest parties.

The case of a corrupted dealer. In this case, the honest parties have no input and hence, and simulation is easy – we just run the protocol with the adversary while perfectly simulating the honest parties. To determine the polynomial $S(x, y)$ to send to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$, we follow the same strategy as the simulator of $\mathcal{F}_{\mathsf{VSS}}$. Moreover, it computes the outputs of the simulated honest parties as in the protocol. For each corrupted moderator $M_i \in I$:

1. If there is a simulated honest party P_k with output $v^k_{M_i} = 1$, then send the message $m_i = (\mathsf{Agreement}, (v^j_{M_i})_{j \notin I}, \mathsf{d}^k_{M_i})$ to the trusted party.
2. If all simulated honest parties P_k have output $v^k_{M_i} = 0$, then send to the trusted party the message $m_i = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_i})_{k \notin I})$.

It is clear that the view of the adversary is identically distributed in both executions. We now show that the outputs in the real and ideal executions are identical by considering two cases on the moderators:

Honest moderator M_j.: If the moderator M_j is honest, then in the ideal execution we have that each honest party P_k holds $v^k_{M_j} = 1$ and $\mathsf{d}^k_{M_j} = \mathsf{d}_{M_j}$ for some $\mathsf{d}_{M_j} \in \{0, 1\}$. Moreover, if the dealer is not discarded then $\mathsf{d}^k_{M_j} = 1$ for all honest parties and $s_k = S(k, 0)$ for some unique degree-(t, t) bivariate polynomial $S(x, y)$.

In the real execution, we first show that all honest parties P_k hold $v^k_{M_j} = 1$ by showing that each of the conditions required for P_k to set $v^k_{M_j} = 1$ in Step 3(b)iiB hold:

1. **$g' = 2$:** The moderator is honest, and therefore when gradecasting $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$ in Step 2d of the **Voting phase**, this message is received with grade 2 by each honest party P_k.
2. **$\mathsf{expectedD}^k_{M_j} = \mathsf{d}_{M_j}$:** From the validity of gradecast, it holds that all honest parties receive the same message $(\mathsf{d}_{M_j}, \mathsf{Bad}_{M_j})$. Furthermore for each party $P_k \notin \mathsf{Bad}_{M_j}$ the moderator M_j must have received the gradecast of a_k with grade 2. From the validity and agreement of gradecast, we have that each honest party must have received the same decision a_k from P_k's gradecast with grade ≥ 1. Hence, the moderator M_j and each honest party P_k agree on the decisions in $(a_\ell)_{\ell \notin \mathsf{Bad}_{M_j}}$ (some honest parties may have received up to t decisions from parties in Bad_{M_j} with grade 1). Furthermore, the moderator M_j and each honest party P_k agree on the number of accept messages they have received with sufficient grade (2 for M_j and ≥ 1 for P_k). Hence, the decisions they each compute must also be the same, that is, for each honest party P_k it holds that $\mathsf{expectedD}^k_{M_j} = \mathsf{d}_{M_j}$.
3. **For every i such that a_i was received with grade 0 it holds that $i \in \mathsf{Bad}'_{M_j}$:** Consider a corrupted party P_i that gradecasted a decision with grade ≤ 1 to all the honest parties. M_j must have included P_i in Bad_{M_j}. If an honest party P_k received a_i from P_i's gradecast with grade 0, then M_j must have included $i \in \mathsf{Bad}_{M_j}$.
4. **$|\mathsf{Bad}'_{M_j}| \leq t$**: Finally, note that only corrupted parties are included in Bad_{M_j} as the decisions gradecasted by each honest party P_k must have been received by M_j with grade 2. Hence, $|\mathsf{Bad}_{M_j}| \leq |I| \leq t$.

Note that from the validity of gradecast, each honest party P_k sets $\mathsf{d}^k_{M_j} = \mathsf{d}_{M_j}$ where d_{M_j} is gradecasted by the dealer. We now complete the proof for this case by showing that if $\mathsf{d}_{M_j} = 1$ is

gradecasted by the moderator, the shares lie on a unique bivariate. If the value gradecasted by the moderator, d_{M_j}, is 1, there must have been at least one honest party, say P_ℓ, that gradecast accept (as at least $t+1$ accept decisions must have been received by M_j). Thus, $\mathsf{happy}_\ell = 1$ and P_ℓ must have held grade 2 in each of the dealer's gradecasts and the dealer must not have been discarded by P_ℓ (non-$\perp$ shares). Additionally, the dealer must have gradecasted the same message gradecasted by each party (else, P_ℓ must have set $\mathsf{happy}_\ell = 0$). Clearly, in this case, the moderated gradecasts emulate the functionality of broadcast (Functionality 3.4) and from the security of Π_{VSS} we know that for each share s_k output by an honest party P_k, it must hold that $s_k = S(k,0)$ on some unique degree-(t,t) bivariate $S(x,y)$.

Corrupted moderator M_i.: We consider three sub-cases based on the flags of the honest parties:

1. **Case 1:** In the ideal execution, there exists some simulated honest party P_k with $v^k_{M_i} = 1$ and $\mathsf{d}^k_{M_i} = 1$. In this case, note that the simulator sets $m_i = (\mathsf{Agreement}, (v^j_{M_i})_{j \notin I}, \mathsf{d}_{M_i} = 1)$ and sends it to the trusted party. In the ideal execution, each honest party P_j receives the same flag $\mathsf{d}^j_{M_i} = 1$ and shares on a unique degree-(t,t) bivariate polynomial $S(x,y)$ (computed by the simulator), together with the flag $v^j_{M_i}$ determined by the simulator.
   In the real execution, since $v^k_{M_i} = 1$, the following must hold: (1) P_k received the moderator's gradecast of $(\mathsf{d}_{M_i}, \mathsf{Bad}_{M_i})$ with grade 2; and (2) for each decision received from P_j with grade < 1, $j \in \mathsf{Bad}_{M_i}$; and (3) $\mathsf{expectedD}_k = \mathsf{d}_{M_i}$; and (4) $|\mathsf{Bad}_{M_i}| \leq t$. From the agreement of gradecast, all honest parties also receive the same message $(\mathsf{d}_{M_i}, \mathsf{Bad}_{M_i})$. Since $\mathsf{d}^k_{M_i} = 1$ it must hold that $\mathsf{expectedD}^k_{M_i} = \mathsf{d}_{M_i} = 1$. Hence, each honest party P_j sets $\mathsf{d}^j_{M_i} = \mathsf{d}_{M_i} = 1$, regardless of whether or not $v^j_{M_i} = 1$ or 0. Note that the simulator makes the exact same decisions as the honest parties in the real, and therefore if some simulated honest party P_j sets $v^j_{M_i} = b$ for some $b \in \{0,1\}$, the honest party P_j sets $v^j_{M_i} = b$ in the real with the exact same value.
   Once again, we complete the proof for this case by showing that the shares output by the honest parties lie on a unique degree-(t,t) bivariate $S(x,y)$. P_k must have received at least $t+1$ accept's from parties not in Bad_{M_i} with grade ≥ 1 out of which at least $t+1-t = 1$ must have been sent by an honest party, say P_ℓ. Therefore, it must hold that $\mathsf{happy}_\ell = 1$ and that P_ℓ sets non-$\perp$ shares in Π_{VSS}. P_ℓ must have received grade 2 in each of the dealer's gradecasts and from the agreement of gradecast, the moderated gradecasts are identical to actual broadcasts and the guarantees of VSS hold (Functionality 4.8). Hence, there exists a unique degree-(t,t) bivariate $S(x,y)$ (which will be computed by the simulator and sent to the trusted party) such that each honest party P_j sets $s_j = S(j,0)$.
2. **Case 2:** In the ideal execution, there exists some simulated honest party P_k with $v^k_{M_i} = 1$ and $\mathsf{d}^\ell_{M_i} = 0$. In this case, the simulator sets $m_i = (\mathsf{Agreement}, (v^j_{M_i})_{j \notin I}, \mathsf{d}_{M_i} = 0)$ and sends it to the trusted party. In the ideal execution, each honest party P_j receives the same flag $\mathsf{d}^j_{M_i} = 1$, and the flag $v^j_{M_i}$ as determined by the simulator.
   In the real execution, since $v^k_{M_i} = 1$, the following must hold: (1) P_k received the moderator's gradecast of $(\mathsf{d}_{M_i}, \mathsf{Bad}_{M_i})$ with grade 2; and (2) for each decision received from P_j with grade < 1, $j \in \mathsf{Bad}_{M_i}$; and (3) $\mathsf{expectedD}_k = \mathsf{d}_{M_i}$; and $|\mathsf{Bad}_{M_i}| \leq t$. From the agreement of gradecast, all honest parties also receive the same message $(\mathsf{d}_{M_i}, \mathsf{Bad}_{M_i})$. Since $\mathsf{d}^k_{M_i} = 0$ it must hold that $\mathsf{expectedD}_k = \mathsf{d}_{M_i} = 0$. Hence, each honest party P_j sets $\mathsf{d}^j_{M_i} = \mathsf{d}_{M_i} = 0$.
3. **Case 3:** For all simulated honest parties P_k, $v^k_{M_j} = 0$.
   In this case, there is no agreement and the simulator sets
   $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k \notin I})$ (the flags are defined by the outputs of the simulated honest parties) and the honest parties output those flags together with $v^k_{M_j} = 0$.
   In the real execution, we get exactly the same – the flags are exactly the same as the simulated honest parties in the ideal.

The error of $\Pi_{\mathsf{mm\text{-}VSS}}$ follows from the error for the compiled Π_{VSS} protocol. □

Efficiency: The protocol $\Pi_{\mathsf{mm\text{-}VSS}}$ (Protocol 5.2) incurs a communication complexity of $O(n^3 \log n)$ bits over the point-to-point channels. However, we will utilize certain batching techniques to maintain a communication complexity of $O(n^3 \log n)$ bits when running $\Pi_{\mathsf{mm\text{-}VSS}}$ in parallel for multiple dealers.

5.1 Batched Multi-Moderated VSS

Looking ahead in our oblivious leader election, each party P_ℓ is required to share m ($= \mathsf{poly} \log n$) uniformly random values. We now show how each moderator M_j can batch the gradecast in the moderation phase (**Phase II**) across multiple instances of (moderated) sharing where each party P_ℓ acts as a dealer.

1. Before the moderation phase (**Phase II**), each party P_k gradecasts a bit a_k^ℓ denoting the decision on the instance of sharing where P_ℓ acts as a dealer.
2. M_j computes a single set Bad_{M_j} across all the instances. Specifically, M_j includes $i \in \mathsf{Bad}_{M_j}$ if the gradecast of a_i^ℓ for any $\ell \in [n]$ by P_i yielded a grade < 2 for the moderator M_j.
3. M_j computes the majority decision among a_k^ℓ for each $\ell \in [n]$ and $k \notin \mathsf{Bad}_{M_j}$ and denotes it as $\mathsf{d}_{M_j}^\ell$.
4. M_j gradecasts $\left(\mathsf{d}_{M_j}^\ell\right)_{\ell=1}^n, \mathsf{Bad}_{M_j}$.

Each moderator M_j gradecasts $\Theta(n)$ bits (the n decisions $\mathsf{d}_{M_j}^\ell$ and the set Bad_{M_j}) which incurs a total communication of $O(n^2 \log n)$ bits for M_j and $O(n^3 \log n)$ bits across all moderators. Naively gradecasting n different sets Bad_{M_j} requires each moderator M_j to gradecast $\Theta(n^2)$ bits which yields a total communication $O(n^3 \log n)$ bits for M_j and $O(n^4 \log n)$ bits across all moderators (which is not suitable in our case).

As in the VSS, we present a corresponding batched multi-moderated VSS functionality $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$ and then present our protocol.

Functionality 5.4: $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$

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

1. **Input:** The dealer holds as input m secrets $s^{(1)}, \ldots, s^{(m)} \in \mathbb{F}$.
2. **Honest Dealer:** The dealer sends $s^{(1)}, \ldots, s^{(m)}$ to $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$. The adversary sends $(f_i^\ell(x), g_i^\ell(y))_{i \in I}$ to $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$ for $\ell = 1, \ldots, m$. The functionality chooses m random bivariate $S^1(x,y), \ldots, S^m(x,y)$ of degree t in x and y such that for each $\ell = 1, \ldots, m$ (i) $s^{(\ell)} = S^\ell(0,0)$; (ii) $S^\ell(x,i) = f_i^\ell(x)$ and $S^\ell(i,y) = g_i^\ell(y)$ for each $i \in I$.
3. **Corrupted dealer:** The adversary sends $S^1(x,y), \ldots, S^m(x,y)$ to $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$. $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$ verifies that each $S^\ell(x,y)$ is of degree t in x and y. If not, the functionality sets $S^\ell(x,y) = \perp$ for each $\ell = 1, \ldots, m$.
4. **Moderators:** For each party $M_j \in [n]$ acting as a moderator:
   (a) If the moderator M_j is honest, then set $v_{M_j}^k = 1$ for each $k \in [n]$. Furthermore,
      i. If the dealer is honest, then set $\mathsf{d}_{M_j}^k = 1$ for each $k \in [n]$.
      ii. If the dealer is corrupted, then set $\mathsf{d}_{M_j}^k = 1$ for each $k \in [n]$ if and only if the ideal functionality sets each $S^\ell(x,y) \neq \perp$ in Step 3 for each $\ell = 1, \ldots, m$.
   (b) If the moderator M_j is corrupted, then receive a message m_j from the adversary.
      i. If $m_j = (\mathsf{Agreement}, (\hat{v}_{M_j}^k)_{k \notin I}, \mathsf{d}_{M_j})$ where $\mathsf{d}_{M_j} \in \{0,1\}$, and for some $k \notin I$ it holds that $v_{M_j}^k = 1$, then for every $k \notin I$ set $v_{M_j}^k = \hat{v}_{M_j}^k$, as received from the adversary. If the ideal functionality sets $S^\ell(x,y) \neq \perp$ in Step 3 for each $\ell = 1, \ldots, m$, then set $\mathsf{d}_{M_j}^k = \mathsf{d}_{M_j}$ for every $k \notin I$. Else, set $\mathsf{d}_{M_j}^k = 0$ for every $k \notin I$.
      ii. If $m_j = (\mathsf{NoAgreement}, (\hat{\mathsf{d}}_{M_j}^k)_{k \notin I})$ where each $\mathsf{d}_{M_j}^k \in \{0,1\}$, then set $v_{M_j}^k = 0$ for every $k \in [n]$ and $(\mathsf{d}_{M_j}^1, \ldots, \mathsf{d}_{M_j}^n) = (\hat{\mathsf{d}}_{M_j}^1, \ldots, \hat{\mathsf{d}}_{M_j}^n)$ as received from the adversary.

5. **Output:** $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ sends to each party P_j the outputs $S^1(j,0),\ldots,S^\ell(j,0)$ and the flags $(v^j_M, \mathsf{d}^j_M)_{M\in[n]}$.

As discussed in Section 4.4, the batching modifications for the VSS do not affect the security due to the argument in [AAPP23]. We now show that the same holds true for the batching of the moderator's gradecasts.

Recall the definition of the Bad_{M_j} set for a moderator M_j. If $i \in \mathsf{Bad}_{M_j}$, then P_i gradecasted some decision a_i^ℓ before the moderation phase, with grade < 2 received by M_j. As a consequence, it holds that $\mathsf{Bad}_{M_j} \subseteq I$ for an honest moderator M_j. If a corrupted party P_i is not included in Bad_{M_j} by an honest moderator M_j, then M_j must have received a grade 2 in all of the gradecasts initiated by P_i. From the properties of gradecast, all honest parties must have received the same decision from P_i with grade ≥ 1. Hence, even across instances, it holds that there is agreement for the honest parties on the decisions gradecasted by each party $k \notin M_j$ (which yields the required properties for multi-moderated VSS). We denote by $\Pi_{\mathsf{mm\text{-}bVSS}}$ the corresponding batched multi-moderated VSS protocol.

Corollary 5.5. *Protocol $\Pi_{\mathsf{mm\text{-}bVSS}}$ securely computes $\mathcal{F}_{\mathsf{mm\text{-}bVSS}}$ (Functionality 5.4), in the presence of a malicious* **adaptive** *adversary controlling at most $t < n/3$. The communication complexity of $\Pi_{\mathsf{mm\text{-}bVSS}}$ is $O(n^3 \log^2 n)$ bits on the point-to-point channels. The protocol is statistically secure, with statistical error $\epsilon < 2n \cdot e^{-m/6}$.*

5.2 Reconstruction with Moderators

Protocol 5.6: $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}VSS}}$

Input: Each party P_i holds $s_i, (\mathsf{d}^i_M)_{M\in[n]}$ and $(v^i_M)_{M\in[n]}$.

1. Each party sends s_i to all. Let $(s'_1,\ldots,s'_n)$ be the shares received.
2. If $\mathsf{d}^i_{M_j} = 1$ for at least $n-t$ parties M_j acting as the moderator, then use Reed-Solomon decoding to reconstruct the unique degree t polynomial $g(y)$ that agrees with at least $2t+1$ values $s'_1,\ldots,s'_n$ and set $s^i = g(0)$. If no unique decoding exists or if less than $n-t$ moderators rejected, then set $s^i = 0$.
3. **Output:** Output s^i.

We show the properties of $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}VSS}}$ with the following theorem:

Theorem 5.7. *If the dealer is honest, then all honest parties output the dealer's secret s. Else, if for each party $M_j \in [n]$ acting as a moderator, there exists an honest party P_k with $v^k_{M_j} = 1$, then each honest party P_k outputs the same secret $s^k = s'$.*

Proof. By the properties of Functionality 5.1, if the dealer is honest, each honest party P_k holds $\mathsf{d}^k_{M_j} = 1$ corresponding to each honest moderator M_j. Furthermore, each honest party P_k holds $s_k = S(k,0)$ where $S(x,y)$ is the bivariate chosen by the honest dealer. As there are $\geq n-t \geq 2t+1$ honest parties, there are sufficient shares from the honest parties to reconstruct $s = S(0,0)$.

For a corrupted dealer, by the properties of Functionality 5.1, if there exists an honest party P_k that holds $v^k_{M_j} = 1$ for a party M_j acting as a moderator, then all honest parties hold the same flag d_{M_j} (we omit the superscript in the ensuing discussion). Furthermore, for an honest party M_j acting as a moderator each honest party P_k holds $v^k_{M_j} = 1$. Suppose that the above holds even for each corrupted party M_i acting as a moderator (note that it may not be the same honest party P_k corresponding to each corrupted party M_i acting as a moderator). We consider two cases:

1. If $\mathsf{d}_{M_j} = 0$ for more than t moderators M_j, then each honest party P_k sets $s^k = 0 = s'$.

2. Else, each honest party P_k must hold $\mathsf{d}_{M_j} = 1$ for at least $n - t$ moderators M_j and all honest parties hold shares on a unique bivariate $S'(x, y)$ of degree t in x and y, that is, for each honest party P_k, it holds that $s_k = S'(k, 0)$. Furthermore, $g(y) = S(y, 0)$ is of degree t. Since there are $\geq n - t \geq 2t + 1$ correct points, the honest parties will be able to reconstruct the unique $g(y)$ where $s_k = g(k)$ for each honest party P_k. Hence, they will all output the same $s^k = g(0) = s'$.

The choice of s' is clearly fixed at the end of $\Pi_{\mathsf{mm\text{-}VSS}}$ based on the flags and shares output by the honest parties. □

Efficiency: The protocol $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}VSS}}$ (Protocol 5.6) incurs a communication complexity of $O(n^2 \log n)$ bits over the point-to-point channels.

5.3 Gradecast

We now present the gradecast functionality that we use in our broadcast. Moreover, we also re-state the protocol of [ZLC23], which we use. The protocol in [ZLC23] is proven with respect to a property-based definition. We use those properties to show that it also works with our gradecast functionality.

Functionality 5.8: $\mathcal{F}_{\mathsf{Gradecast}}$

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

1. If the dealer is honest: the dealer sends m to the functionality, and all parties receive $(m, 2)$ as output.
2. If the dealer is corrupted then it sends some message M to the functionality.
   (a) If $M = (\mathsf{ExistsGrade2}, m, (g_j)_{j \notin I})$ for some $m \in \{0,1\}^*$ and each $g_j \in \{1, 2\}$, then verify that $g_j \geq 1$ and that at least one honest party receives grade 2. Send (m, g_j) to each party P_j.
   (b) If $M = (\mathsf{NoGrade2}, (m_j, g_j)_{j \notin I})$ where each $m_j \in \{0,1\}^*$ and $g_j \in \{0, 1\}$, then verify that for every $j, k \notin I$ with $g_j = g_k = 1$ it holds that $m_j = m_k$. Then, send (m_j, g_j) to each party P_j.

Balancing the protocol. Additionally, we note that the protocol of [ZLC23] is not balanced, i.e., the sender sends $O(nL + n \log n)$ bits, whereas each party sends $O(n \log n)$ bits. To balance the communication in the protocol (such that each party sends/receives $O(L + n \log n)$), we perform the following:

1. At the start of the protocol, the sender interprets its input as a polynomial of degree $t/5$ $f(x)$ and sends $f(i)$ to each party. Note that in the gradecast protocol of [ZLC23], the sender sends $f(x)$ to all the parties.
2. Each party P_j upon receiving $f(i)$, sends $f(i)$ to everyone.
3. Each party denotes by $f'(1), \ldots, f'(n)$ the points received in the previous step and executes Reed Solomon decoding to reconstruct the unique polynomial of degree $t/5$ that agrees with at least $2t + 1$ of the points received.
4. At this point, each party either receives some unique polynomial or is unable to interpolate a unique polynomial. The latter case can be interpreted as a corrupted sender not delivering the polynomial to some party in Round 1 of the protocol in [ZLC23]. Hence, the parties proceed with the protocol in [ZLC23] with the values they have received.

Protocol 5.9: Balanced $\Pi_{\mathsf{Gradecast}}$

Input: The dealer P holds a degree d polynomial $f(x)$ over a field $\mathbb{F}$ as input that it wishes to distribute. All other parties have no input.
The protocol:

1. **Dealer's distribution:** P sends $f(i)$ to every P_i. Let $f(i)$ denote the value received from the dealer by P_i.
2. **Interpolating the input polynomial:**
   (a) P_i sends $f(i)$ to each P_j.
   (b) Let $f'(1), \ldots, f'(n)$ denote the points received in the previous step.
   (c) Run Reed Solomon decoding to reconstruct the unique polynomial of degree $t/5$ that agrees with at least $2t+1$ of the points received. Let $f_i(x)$ denote the unique polynomial. If such a polynomial does not exist, then set $f_i(x) = \perp$.
3. **Exchange:**
   (a) P_i sends $(f_i(j), f_i(i))$ to each P_j.
4. **Set Agree_i^1:**
   (a) Let $(u_{j,i}, v_{j,i})$ be the values P_i received from P_j. If $(u_{j,i}, v_{j,i}) = (f_i(i), f_i(j))$ $(= f_j(i), f_j(j))$, then add j to Agree_i^1.
   (b) If $|\mathsf{Agree}_i^1| \geq n - t$ then send OK_1 to everyone.
5. **Set Agree_i^2:**
   (a) Initialize $\mathsf{Agree}_i^2 = \mathsf{Agree}_i^1$. If OK_1 was not received from $P_j \in \mathsf{Agree}_i^1$ then remove j from Agree_i^2. If $|\mathsf{Agree}_i^2| \geq n - t$ then send OK_2 to everyone.
6. **Update Agree_i^3:**
   (a) Initialize $\mathsf{Agree}_i^3 = \mathsf{Agree}_i^2$. If OK_2 was not received from $P_j \in \mathsf{Agree}_i^2$ then remove j from Agree_i^3. If $|\mathsf{Agree}_i^3| \geq n - t$ then send OK_3 to everyone.
7. **Set CORE_i:**
   (a) Define CORE_i as the set of all parties P_j from which P_i received a message OK_3.
   (b) If $i \in \mathsf{CORE}_i$ then set $y_i = f_i(i)$. If $i \notin \mathsf{CORE}_i$ then define y_i as the majority value among the set $\{u_{j,i} \mid j \in \mathsf{CORE}_i\}$, where $u_{j,i}$ is the value P_i received from P_j in Step 4a. If no majority value exists, then define $y_i = \perp$.
   (c) Send y_i to every party P_j.

Output: Run Reed Solomon decoding of a polynomial of degree d on the values $(y_1, \ldots, y_n)$ received in the previous step. Let $f_i'(x)$ be the resulting polynomial.

1. If $i \in \mathsf{CORE}_i$ and $|\mathsf{CORE}_i| \geq n - t$, then output $(f_i'(x), 2)$.
2. If $i \notin \mathsf{CORE}_i$ and $|\mathsf{CORE}_i| \geq n - t$, then output $(f_i'(x), 1)$.
3. Otherwise, output $(\perp, 0)$.

We show that the protocol realizes the $\mathcal{F}_{\mathsf{Gradecast}}$-functionality, given the property-based proof of [ZLC23]:

1. **An honest dealer.** In this case, the honest dealer sends m to the functionality $\mathcal{F}_{\mathsf{Gradecast}}$ and all parties receive output $(m, 2)$. The simulator, therefore, receives the message $(m, 2)$ from the trusted party. Since no other party has any input to the protocol, and since the simulator has m, it simulates the protocol by just executing it with the input m of the dealer, and it generates the view of the corrupted parties. Since the protocol is deterministic, the view of the adversary in the simulated execution and the real execution must be *identical*. The output of the ideal and real are the same as follows from the validity of the protocol [ZLC23, Lemma 3].
2. **A corrupt dealer.** In this case, the honest parties have no input, and the simulator can again perfectly simulate the honest parties interacting with the corrupt parties (adversary). At the end of the simulated execution, the simulator holds the outputs corresponding to the honest parties, that is, (m_j, g_j) for each $j \notin I$. Clearly, the adversary's view is identical in the simulated and real execution as the protocol is deterministic. By showing that the non-equivocation [ZLC23, Lemma 4] and agreement [ZLC23, Lemma 5] properties hold in

the protocol, then the output of the simulated execution also satisfies those properties, and the simulator can generate the appropriate message M to send to the trusted party. I.e., if there is one honest party that outputs grade 2 in the simulated execution, then it sends the message ExistsGrade2, and if not then it sends NoGrade2, all are together with the outputs of the honest parties. Since agreement and non-equivocation hold, the trusted party accepts the message that the simulator sends and the output of the honest parties in the ideal execution is exactly the same as in the simulated honest parties in the ideal execution (which is the same as the real execution).

6 Oblivious Leader Election

As mentioned before, we realize the same oblivious leader election functionality as in [AAPP22]. Formally, oblivious leader election defines a secure sampling from some family $\mathcal{D}$ of distributions. The adversary is required to choose one of the distributions $D \in \mathcal{D}$ and the ideal functionality samples randomness r and delivers to each party P_i the output $\ell_i \in \{1, \ldots, n\}$ where $(\ell_1, \ldots, \ell_n) = D(r)$. Following the notation in [KK06, AAPP22], we say that D is valid if the following holds:

$$\mathsf{Pr}_r[D(r) = (j, \ldots, j) \mid j \notin I] \geq \delta$$

for some constant $\delta > 0$.

Functionality 6.1: $\mathcal{F}_{\mathsf{OLE}}$

The functionality is parameterized by the set of corrupted parties $I \subset [n]$ and the family of distributions $\mathcal{D}$.

1. The functionality receives from the adversary a sampler D and verifies that $D \in \mathcal{D}$. If not, then it takes some default sampler in $D \in \mathcal{D}$.
2. The functionality chooses a random $r \leftarrow \{0,1\}^{\mathsf{poly}(n)}$ and samples $(\ell_1, \ldots, \ell_n) = D(r)$.
3. The functionality gives r to the adversary and it gives ℓ_i to each P_i.

Protocol 6.2: Π_{OLE}

1. **Choose random hidden moderators:** Each party P_i samples a set of hidden moderators $\mathcal{M}_i$ by choosing $m < n$ elements from $[n]$ (with replacements).
2. **Choose and commit weights:** Each party P_i acts as a dealer and chooses $c_{i \to j}$ as random values in $\{1, \ldots, n^4\}$, for every $j \in \mathcal{M}_i$. P_i then runs the following m times in parallel, for each $\ell \in [m]$, where P_i acts as a dealer in each instance:
   (a) The dealer distributes the values $(j, c_{i \to j})$ for each $j \in \mathcal{M}_i$ using $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ where P_i is the dealer.
   (b) Each party P_k gets as output a share $s_k^{i,\ell}$, outputs $\mathsf{d}_{i,j}^k$ and a flag $v_{i,j}^k$ for each party P_j acting as moderator.

   Note that the above is run for all dealers $P_1, \ldots, P_n$ in parallel, where each dealer has m parallel instances (in total $m \cdot n$ invocations). Upon completion, let $\mathsf{Successful}_i$ be the set of moderators for which P_i holds a flag $v = 1$ in all executions, that is, $\mathsf{Successful}_i = \{j \mid v_{d,j}^i = 1 \text{ for all dealers } d \in [n]\}$.
3. **Reconstruct the weights and recipients and pick a leader:** The reconstruction phase, $\Pi_{\mathsf{mm\text{-}VSS}}^{\mathsf{Rec}}$ (Protocol 5.6) of each of the above mn instances of multi-moderated secret sharing is run in parallel to reconstruct the secrets previously shared.
   (a) Let $\mathcal{M}_i^k$ and the values $c_{i \to j}$ for each $i \in [n]$, $j \in \mathcal{M}_i^k$ denote P_k's view of the set R_i and $c_{i \to j}$ for each $i \in [n]$, $j \in \mathcal{M}_i$, that is, the reconstructed value for the instance where P_i was the dealer and chose $j \in \mathcal{M}_i$ as a hidden moderator (included in the reconstructed value for P_i's instance).

(b) P_k sets $c_j^k = \sum_{i,j \in \mathcal{M}_i^k} c_{i \to j} \mod n^4$ and outputs j that minimizes c_j^k among all $j \in \mathsf{Successful}_k$ (break ties arbitrarily).

The proof from [KK06, AAPP22] carries over almost verbatim in our case, conditioned on the following event: *For each party P_j which succeeds in the moderation, some honest party P_k included P_j in its hidden moderator set $\mathcal{M}_i$.* We first present only the bound on the above event. To that end, we define: $\mathsf{Successful} = \bigcup_{k \in H} \mathsf{Successful}_k$, where H is the set of all parties that were honest at the end of phase 2. We now define Bad_j for $j \in \mathsf{Successful}$ which defines the event that the party j was not chosen as a hidden moderator by any honest dealer.

$\underline{\mathsf{Bad}_j}$: *No $P_k \in H$ chose $j \in \mathcal{M}_k$.*

We then define the event Bad as the disjunction of the events Bad_j over all $j \in \mathsf{Successful}$. If Bad does not occur, then for each $j \in \mathsf{Successful}$ there exists some honest party P_k that chose P_j as a hidden moderator, that is, $j \in \mathcal{M}_i$. Note that this is exactly the informal event we describe above.

$\underline{\mathsf{Bad}}$: $\bigcup_{j \in \mathsf{Successful}} \mathsf{Bad}_j$.

We now give a bound on the probability of Bad occurring.

Claim 6.3. *The probability that Bad occurs is at most $n \cdot e^{-2m/3}$ where each party chooses m hidden moderators in Protocol 6.2.*

Proof. We first upper bound $\mathsf{Pr}[\mathsf{Bad}_j]$ and use a union bound over all parties in $\mathsf{Successful}$ to give an upper bound on $\mathsf{Pr}[\mathsf{Bad}]$. The probability that no $k \in H$ chooses $j \in \mathcal{M}_k$ is $1 - 1/n$ for each independent sample. For $t < n/3$, it holds that $|H| \geq 2n/3$ and hence,

$$\mathsf{Pr}[\mathsf{Bad}_j] \leq \left(\left(1 - \frac{1}{n}\right)^m\right)^{|H|} \leq \left(\left(1 - \frac{1}{n}\right)^n\right)^{2m/3} \leq e^{-2m/3}.$$

From the definition of Bad and the fact that $|H| \leq n$, it must hold that $\mathsf{Pr}[\mathsf{Bad}] \leq n \cdot \mathsf{Pr}[\mathsf{Bad}_j] \leq n \cdot e^{-2m/3}$. □

We now prove the security of Π_{OLE} in the following theorem:

Theorem 6.4. *Protocol Π_{OLE} (Protocol 6.2), perfectly securely computes the functionality $\mathcal{F}_{\mathsf{OLE}}$, (Functionality 6.1), in the presence of a malicious adversary controlling at most $t < n/3$ parties.*

Proof. The simulator first simulates the call to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ (Functionality 5.1) for all the secret sharings of the honest dealers. The simulator receives the adversary's shares $(f_i(x), g_i(y))_{i \in I}$ and gives them to the ideal functionality. For each honest moderator P_j, the simulator sets $\mathsf{d}_{i,j}^k = \mathsf{accept}$ and flag $v_{i,j}^k = 1$ for each P_k. For a corrupted moderator P_j, the adversary's message can be used to simulate the appropriate outputs. For an honest moderator the parties reconstruct the honest dealer's secret and for a corrupt moderator they may reconstruct either the dealer's secret or 0. Now the simulator simulates the calls to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ (Functionality 5.1) for all the secret sharings of the corrupted dealers. In this case, the simulation is trivial as the simulator can just run the steps of the ideal functionality as per the messages sent by the adversary. The simulator is now required to give to $\mathcal{F}_{\mathsf{OLE}}$ a sampling algorithm $D \in \mathcal{D}$. The sampler D is constructed from the adversary's message denoting the flags for the corrupted moderators and the simulated honest party's outputs for the honest moderators. Note that the sampler holds $\mathcal{M}_i^k$ and $c_{i \to j}^k$ for each corrupted dealer P_i, each hidden moderator $j \in \mathcal{M}_i^k$ and corresponding to the view of the simulated honest party P_k. Given these values and the flags, D is defined as follows:

1. Receive input r which corresponds to a uniformly random value $c_{i \to j} \in \{1, \ldots, n^4\}$ and a uniformly random set of hidden moderators $\mathcal{M}_i$ (sampled from $[n]$ with replacements) for each honest dealer P_i and each $j \in \mathcal{M}_i$.
2. Based on the flags, simulated moderator messages and honest moderator definition, compute $c_{i \to j}^k, v_{i,j}^k, \mathsf{d}_{i,j}^k$ for each honest dealer P_i, each hidden moderator P_j and each party P_k.

3. Compute the output ℓ_k for each party P_k as per the protocol description.

The simulator can now simulate the reconstruction phase using the randomness r and the adversary's chosen shares in the simulated multi-moderated secret sharing. Since the sampler is constructed from the simulated protocol execution, it must hold that the outputs of the honest parties after the reconstruction are identical to the outputs of the simulated honest parties which is in turn identical to the output of the sampler. As in [KK06, AAPP22], it now suffices to show that the sampler D^{hybrid} defined above is valid.

Proving that D^{hybrid} is valid. Recall that we have to prove that

$$\mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I] \geq \delta$$

for some constant $\delta > 0$. In the hybrid model, the properties $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ hold perfectly (with probability 1). We first define:

$$\mathsf{Successful} = \bigcup_{k \in H} \mathsf{Successful}_k \ ,$$

where H is the set of all parties that were honest at the end of phase 2. From the properties of $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$, even if a single honest party P_k holds $v^k_{d,j} = 1$ for some dealer P_d and some party P_j acting as the moderator, then the honest parties agree on all values reconstructed from P_d's sharing. Hence, if reconstruction is successful and $k \in \mathsf{Successful}$, then for any honest P_i, P_j and any dealer d, it holds that $\mathcal{M}^i_d = \mathcal{M}^j_d$ and $c^i_{d\to k} = c^j_{d\to k}$, that is, the honest parties agree on the dealers' hidden moderators and chosen weights. We can thus omit the superscripts i and j.
We now utilize the bound derived in Claim 6.3 to compute the success probability of the sampler in the hybrid model. We denote by Bad the event defined in Claim 6.3. Recall that Bad is the event that no honest party chooses any party $j \in \mathsf{Successful}$ as a hidden moderator.

$$\begin{aligned}
&\mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I] \\
&= \mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I, \overline{\mathsf{Bad}}] \cdot \mathsf{Pr}[\overline{\mathsf{Bad}}] \\
&= \mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I, \overline{\mathsf{Bad}}] \cdot (1 - \mathsf{Pr}[\mathsf{Bad}]) \\
&\geq \mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I, \overline{\mathsf{Bad}}] \cdot (1 - ne^{-2m/3})
\end{aligned}$$

Conditioned on the event that Bad does not occur, each $j \in \mathsf{Successful}$ was chosen as a hidden moderator by some honest dealer P_k. Hence, $c_{k\to j}$ must be uniformly random and secret until phase 2. Furthermore, for every corrupted dealer P_k that also chose $j \in \mathcal{M}_k$, the value $c_{k\to j}$ must have been chosen independently of the other weights. Hence, the value c_j for each $j \in \mathsf{Successful}$ must be uniformly distributed in $\{1, \ldots, n^4\}$ and following similar analysis as in [KK06, AAPP22] for the probability that an honest party is chosen from uniformly random weights c_j for parties in $\mathsf{Successful}$. For completeness, we repeat the same and first define the event $\mathsf{HonestChosen}$. Let $\mathsf{HonestChosen}$ be the event where the index k for which c_k is minimal among all parties in $\mathsf{Successful}$ is an index of an honest party.

$$\begin{aligned}
\mathsf{Pr}[\mathsf{HonestChosen}] &\geq \mathsf{Pr}[\mathsf{HonestChosen} \mid \forall i, j \in \mathsf{Successful}\ c_i \neq c_j] \\
&\quad \cdot \ \mathsf{Pr}[\forall i, j \in \mathsf{Successful}\ c_i \neq c_j] \\
&\geq \frac{n-t}{n} \cdot (1 - \mathsf{Pr}[\exists i, j \in \mathsf{Successful}\ c_i \neq c_j) \\
&\geq \frac{n-t}{n} \cdot \left(1 - n^2 \cdot \frac{1}{n^4}\right) \geq \frac{n-t}{n} - \frac{1}{n^2} \geq \frac{1}{2}
\end{aligned}$$

In our setting we have that,

$$\mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I, \overline{\mathsf{Bad}}] = \mathsf{Pr}[\mathsf{HonestChosen}]$$

Applying the same in our bound, we have

$$\mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I] \geq \frac{1}{2} \cdot (1 - ne^{-2m/3})$$

It is sufficient if the success probability is constant, say $1/4$. We can compute an appropriate setting for the parameter m to allow for the constant success probability.

$$\begin{aligned}
\frac{1}{2} \cdot (1 - ne^{-2m/3}) &\geq \frac{1}{4} \\
1 - ne^{-2m/3} &\geq \frac{1}{2} \\
ne^{-2m/3} &\leq \frac{1}{2} \\
-\frac{2m}{3} &\leq -\log 2n \\
m &\geq \frac{3}{2}\log 2n
\end{aligned}$$

For instance, $m = 2\log 2n$ yields constant success probability and completes our proof in the hybrid model.

Note that the above analysis applies for the distribution generated by the sampler D^{hybrid} where the parties have access to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$. We denote by D the sampler where the calls to $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ are replaced by calls to the protocol $\Pi_{\mathsf{mm\text{-}VSS}}$. **Proving that D is valid after composition.** After realizing $\mathcal{F}_{\mathsf{mm\text{-}VSS}}$ with statistical security where the statistical error of the protocol is ϵ, it must hold that the sampler D succeeds in the event that D^{hybrid} succeeds **and** there is no error in any of the n instances of $\Pi_{\mathsf{mm\text{-}VSS}}$. As the two events are disjoint, the success probability of D is the product of the success probability of D^{hybrid} and the probability that there is no error in any of the n instances $\Pi_{\mathsf{mm\text{-}VSS}}$. From a union bound, it holds that the probability that there is no error in any of the n instances of $\Pi_{\mathsf{mm\text{-}VSS}}$ is at least $1 - n \cdot \epsilon$. Hence,

$$\begin{aligned}
\mathsf{Pr}_r[D(r) = (j, \ldots, j) \mid j \notin I] &\geq (1 - n \cdot \epsilon) \cdot \mathsf{Pr}_r[D^{\mathsf{hybrid}}(r) = (j, \ldots, j) \mid j \notin I] \\
&\geq (1 - n \cdot \epsilon) \cdot \frac{1}{4}
\end{aligned}$$

Note that if we have $1 - n \cdot \epsilon \geq 1/2$, that is, $n \cdot \epsilon \leq 1/2$, we have that even after composition D still defines a valid sampling algorithm. Critically, inverse polynomial error $(1/\mathsf{poly}(n))$ was sufficient for the multi-moderated secret sharing to construct oblivious leader election. From Theorem 5.3 we have that the statistical error of the protocol $\Pi_{\mathsf{mm\text{-}VSS}}$ is $\epsilon = 2n \cdot e^{-m/6}$. We can thus provide a different lower bound on the parameter m

$$\begin{aligned}
n \cdot \epsilon &\leq 1/2 \\
2n^2 \cdot e^{-m/6} &\leq 1/2 \\
\frac{-m}{6} &\leq -\log 4n^2 \\
m &\geq 12\log 2n
\end{aligned}$$

Once again, $m = 12\log 2n$ $(> 2\log 2n)$ yields inverse polynomial statistical error and completes our proof.

□

Efficiency: Each instance of $\Pi_{\mathsf{mm\text{-}VSS}}$ incurs a cost of $O(n^2 \log n)$ bits on the point-to-point channels and gradecast of $O(m \log n)$ bits by each party. Since there are $O(mn)$ instances (which have the required batching modifications), the total cost is thus $O(mn^3 \log n)$ bits. Fixing $m = 6\log 4n$ (suffices to realize oblivious leader election) yields a total communication complexity of $O(n^3 \log^2 n)$.

7 Broadcast, and Parallel Broadcast

The protocols for Byzantine agreement, broadcast and parallel broadcast are identical to the protocols of [FG03,KK06,AAPP22] and we present them here for the completeness of our result.

Protocol 7.1: Π_{BA}

Input: Each party P_i holds a bit b_i.
Initialization: Each party initializes $\mathsf{decided}_i = \mathsf{false}$ and $\mathsf{openToAcceptRandom} = \mathsf{false}$. Run the following iteratively until termination:

1. **Round I – each party P_i:**
   (a) Send b_i to all parties.
   (b) Let $b_{j,i}$ be the bit receive from P_j (if no value was received, use the value from the previous iteration; at the start of the protocol, use a default value).
2. **Round II – each party P_i:**
   (a) Set $\mathcal{S}_i^0 := \{j \mid b_{j,i} = 0\}$ and $\mathcal{S}_i^1 := \{j \mid b_{j,i} = 1\}$.
   (b) If $|\mathcal{S}_i^0| \geq t+1$ then set $b_i = 0$. If $|\mathcal{S}_i^0| \geq n - t$ then set $\mathsf{decided} = \mathsf{true}$.
   (c) Send b_i to all parties. If a value was received from party P_j, then store it as $b_{j,i}$.
3. **Round III – each party P_i:**
   (a) Update $\mathcal{S}_i^0$ and $\mathcal{S}_i^1$ according to the new values $b_{1,i}, \ldots, b_{n,i}$.
   (b) If $|\mathcal{S}_i^1| \geq t+1$ then set $b_i = 1$. If $|\mathcal{S}_i^1| \geq n - t$ then set $\mathsf{decided} = \mathsf{true}$.
   (c) Send b_i to all parties. If a value was received from party P_j, then store it as $b_{j,i}$.
4. **Round IV – each party P_i:**
   (a) If $\mathsf{decided}_i = \mathsf{false}$ then set $\mathsf{openToAcceptRandom} = \mathsf{true}$.
   (b) Update $\mathcal{S}_i^0$ and $\mathcal{S}_i^1$ according to the new values $b_{1,i}, \ldots, b_{n,i}$.
   (c) If $|\mathcal{S}_i^0| \geq t+1$ then set $b_i = 0$. If $|\mathcal{S}_i^0| \geq n - t$ then set $\mathsf{openToAcceptRandom} = \mathsf{false}$.
   (d) Send b_i to all parties. If a value was received from party P_j, then store it as $b_{j,i}$.
5. **Round V – each party P_i:**
   (a) Update $\mathcal{S}_i^0$ and $\mathcal{S}_i^1$ according to the new values $b_{1,i}, \ldots, b_{n,i}$.
   (b) If $|\mathcal{S}_i^1| \geq t+1$ then set $b_i = 1$. If $|\mathcal{S}_i^1| \geq n - t$ then set $\mathsf{openToAcceptRandom} = \mathsf{false}$.
   (c) Send b_i to all parties. If a value was received from party P_j, then store it as $b_{j,i}$.
6. **Round VI – each party P_i:**
   (a) All parties execute Π_{OLE} (Protocol 6.2) and let ℓ_i be the output of P_i.
   (b) If $\mathsf{openToAcceptRandom}_i = \mathsf{true}$, then set $b_i = b_{\ell_i}$
   (c) If $\mathsf{decided}_i = \mathsf{true}$, then output b_i and terminate. Else, proceed to the next iteration.

Theorem 7.2. *Protocol Π_{BA} (Protocol 7.1), perfectly securely computes the functionality $\mathcal{F}_{\mathsf{BA}}$ (Functionality 3.5), in the presence of a malicious adversary controlling at most $t < n/3$ parties and incurs a communication complexity of $O(n^3 \log^2 n)$ bits in expectation.*

7.1 Broadcast

To realize this protocol, the dealer first gradecasts the message. Note that if any honest party held grade 2, then the honest parties agree on some non-$\perp$ message m' such that $m' = m$ when the dealer is honest. It thus, suffices to have the parties agree on whether there was a grade 2 or not which can be performed at the cost of $O(n^3 \mathsf{poly} \log n)$ bits in expectation.

Protocol 7.3: Π_{BC}

- **Input:** The dealer holds a message $M \in \{0,1\}^L$.
- **Common input:** A parameter L.
  1. **The dealer:** Gradecast M.
  2. **Each party P_i:** Let M' be the resultant message and let g be the associated grade. All parties run Byzantine agreement where input of P_i is 1 if and only if $g = 2$.

- **Output:** If the output of the Byzantine agreement is 1, then output M'. Else, output $\perp$.

Theorem 7.4. *Protocol Π_{BC} (Protocol 7.3), perfectly securely computes the functionality $\mathcal{F}_{\mathsf{BC}}$ (Functionality 3.4), in the presence of a malicious adversary controlling at most $t < n/3$ parties. For an input message M of length L bits, the protocol incurs a communication complexity of $O(nL)$ bits and an additional $O(n^3 \log^2 n)$ bits in expectation with $O(1)$ rounds in expectation.*

7.2 Parallel broadcast

Consider the case when each party wishes to broadcast some message of size L bits each. As in [FG03, AAPP22], a single election can be used across the instances. Hence, we have the following corollary:

Corollary 7.5. *There exists a perfectly secure parallel-broadcast in the presence of a malicious adversary controlling at most $t < n/3$ parties. For n parties to each broadcast L bits, the protocol incurs a communication complexity of $O(n^2 L)$ bits and an additional $O(n^3 \log^2 n)$ bits in expectation with $O(1)$ rounds in expectation.*

7.3 Lower Bound

We claim that the broadcast of L ($\geq n$) bits by each of the n parties in parallel is $\Omega(n^2 L + n^3)$ bits. Intuitively, our proof relies on the fact that if a parallel broadcast protocol beats the above lower bound, i.e, if a parallel broadcast protocol incurs a total communication of $o(n^2 L + n^3)$ bits, then the same protocol could have been used to design a single sender broadcast protocol where the total communication is $o(nL)$ bits which is impossible.

Theorem 7.6. *There does not exist any perfectly-secure parallel broadcast with communication complexity $o(n^2 L + n^3)$ bits that is secure against $t \in \Theta(n)$ corruptions for $L \geq n$.*

Proof. We build a broadcast protocol from the parallel broadcast protocol. Assume that the sender has a message $M = (M_1, \ldots, M_{t+1})$ of size $t+1$ over some field $\mathbb{F}$. The broadcast protocol is as follows:

1. The sender interprets its message as a polynomial of degree t over $\mathbb{F}$. It sends to each party P_i its point $M(i)$ privately.
2. The parties run the parallel broadcast protocol where each party P_i broadcasts $M(i)$.
3. Let $M'(1), \ldots, M'(n)$ be the points received in the parallel broadcast. Use Reed Solomon decoding to reconstruct the unique polynomial of degree t that agrees with at least $2t+1$ of the points received. Output that polynomial.

The communication complexity of this broadcast protocol is exactly the parallel broadcast where each party has to broadcast one field element plus the $n \cdot ||\mathbb{F}||$ bits over the point-to-point channels.

Now, when having a field $\mathbb{F}$ of size at least n bits, the communication complexity of the broadcast protocol becomes $n^2 + n \times \mathcal{BC}(n)$, where $n \times \mathcal{BC}(n)$ denotes the cost of a parallel broadcast of n bits. If $n \times \mathcal{BC}(L) \in o(n^2 L + n^3)$ then the cost of our broadcast protocol is $n^2 + o(n^3) \in o(n^3)$. However, any broadcast protocol for $L = n \cdot (t+1)$ bits must incur a communication of at least $nL = n^2(t+1) \in \Omega(n^3)$ whenever $t \in \Omega(n)$. This gives a contradiction. Finally, note that the contradiction arises for $\|\mathbb{F}\| \geq n$, and hence we implicitly require that $L \geq n$. □

References

[AA22] Ittai Abraham and Gilad Asharov. Gradecast in synchrony and reliable broadcast in asynchrony with optimal resilience, efficiency, and unconditional security. In *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 392–398. ACM, 2022. 6

[AAPP22] Ittai Abraham, Gilad Asharov, Shravani Patil, and Arpita Patra. Asymptotically free broadcast in constant expected time via packed VSS. In *Theory of Cryptography - 20th International Conference, TCC 2022*, volume 13747 of *Lecture Notes in Computer Science*, pages 384–414. Springer, 2022. 3, 4, 5, 6, 7, 8, 9, 10, 37, 38, 39, 40, 42

[AAPP23] Ittai Abraham, Gilad Asharov, Shravani Patil, and Arpita Patra. Detect, pack and batch: Perfectly-secure MPC with linear communication and constant expected time. In *Advances in Cryptology - EUROCRYPT 2023*, volume 14005 of *Lecture Notes in Computer Science*, pages 251–281. Springer, 2023. 5, 6, 10, 11, 12, 18, 19, 25, 26, 27, 34

[ACD+23] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. *Distributed Comput.*, 36(1):3–28, 2023. 3

[ACS22] Gilad Asharov, Ran Cohen, and Oren Shochat. Static vs. adaptive security in perfect mpc: A separation and the adaptive security of bgw. In *Conference on Information-Theoretic Cryptography - ITC 2022*. (To Appear), 2022. 15

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the bgw protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 2017. 14

[ALP22] Ittai Abraham and Andrew Lewis-Pye. Phase-king through the lens of gradecast: A simple unauthenticated synchronous byzantine agreement protocol. *Decentralized Thoughts, Blog Post*, 2022. https://decentralizedthoughts.github.io/2022-06-09-phase-king-via-gradecast/. 6

[BDH10] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Brief announcement: Simple gradecast based algorithms. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 194–197. Springer, 2010. 6

[Ben83] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proc. of the Annual Symposium on Principles of Distributed Computing (PODC)*, 1983. 3, 6

[BGP92] Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science*. 1992. 3, 7

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of Annual ACM Symposium on Theory of Computing*, 1988. 5, 6, 10, 11, 12

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000. 6, 14, 15

[CCGZ19] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. *J. Cryptol.*, 32(3):690–741, 2019. 6

[CCP22] Anirudh Chandramouli, Ashish Choudhury, and Arpita Patra. A survey on perfectly secure verifiable secret-sharing. *ACM Comput. Surv.*, 54(11s), sep 2022. 16

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *International conference on the Theory and Applications of Cryptographic Techniques*, pages 311–326. Springer, 1999. 5

[CDD+01] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, 2001. 6, 14, 15, 19

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395. IEEE Computer Society, 1985. 6

[Che21] Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *DISC 2021*, volume 209 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 3, 7

[CW89] Brian A Coan and Jennifer L Welch. Modular construction of nearly optimal byzantine agreement protocols. In *ACM Symposium on Principles of distributed computing*, 1989. 3, 7

[DR82] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In Robert L. Probert, Michael J. Fischer, and Nicola Santoro, editors, *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Ottawa, CanadaAugust 18-20, 1982*, pages 132–140. ACM, 1982. 3

[Fel88] Paul Neil Feldman. *Optimal Algorithms for Byzantine Agreement.* PhD thesis, Massachusetts Institute of Technology, 1988. 5, 6

[FG03] Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In *PODC*, 2003. 14, 40, 42

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 1982. 3, 4, 6

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988. 3, 4, 5, 6, 7, 8, 10, 13

[GP90] Oded Goldreich and Erez Petrank. The best of both worlds: Guaranteeing termination in fast randomized byzantine agreement protocols. *Inf. Process. Lett.*, 36(1):45–49, 1990. 6

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference*, 2006. 3, 4, 5, 6, 7, 8, 9, 13, 37, 38, 39, 40

[KLR06] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006. 14

[Kum12] Ranjit Kumaresan. *Broadcast and verifiable secret sharing: New security models and round optimal constructions.* University of Maryland, College Park, 2012. 5

[LLR02] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In Aleta Ricciardi, editor, *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*, pages 203–212. ACM, 2002. 6

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 1982. 3, 6

[NRS+20] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. *arXiv preprint arXiv:2002.11321*, 2020. 7

[PCR08] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Round efficient unconditionally secure multiparty computation protocol. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2008. 5

[PCRR09] Arpita Patra, Ashish Choudhary, Tal Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing revisited. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 487–504, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 5

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 1980. 3, 6

[Rab83] M. O. Rabin. Randomized byzantine generals. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 1983. 3, 6

[Rab94] Tal Rabin. Robust sharing of secrets when the dealer is honest or cheating. *Journal of the ACM (JACM)*, 41(6):1089–1109, 1994. 5

[TLP22] Georgios Tsimos, Julian Loss, and Charalampos Papamanthou. Gossiping for communication-efficient broadcast. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 439–469. Springer, 2022. 6

[ZLC23] Jianjun Zhu, Fan Li, and Jinyuan Chen. Communication-efficient and error-free gradecast with optimal resilience. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 108–113, 2023. 13, 27, 35, 36

A Glossary: Broadcast in Expected Constant Rounds

Implementation of broadcast in constant expected number of rounds consists of several underlying building blocks. To understand the large picture, we provide an overview of the different primitives and their interplay. This overview can be viewed as a glossary of the various primitives, and therefore we also repeat several (informal) definitions of primitives we already discussed.

Broadcast – See Section 3.3 Broadcast involves a distinguished party, a sender, that holds a message M. The primary objective is to ensure that all n parties receive an identical message M' (reaching agreement), while maintaining the condition that if the sender is honest, $M = M'$ (ensuring validity). To implement broadcast successfully, two fundamental primitives are required: Gradecast and Byzantine agreement.

Gradecast – See Section 5.3 As we mention, Gradecast is a relaxation of Broadcast in which there is a distinguished party with input message M, and all parties output some (M', g) – some message together with a grade $g \in \{0, 1, 2\}$. If the sender is honest, then for all honest parties the output is $(M, 2)$. If the sender is corrupted but there exists an honest party with $g = 2$, then all honest parties are guaranteed to have the same output message (but perhaps with a grade 1). In the absence of an honest party with a grade 2, all honest parties with a grade 1 must receive an identical message, while other honest parties (with a grade 0) might have varying output messages.

Byzantine agreement – See Section 3.3 In Byzantine agreement (of a single bit), the input of each party P_i is some bit $b_i \in \{0, 1\}$, and the output is also a bit. The desired properties of Byzantine agreement are validity and consistency. Validity ensures that if all honest parties have the same input bit, their output must be that input bit. Consistency guarantees that even if honest parties have different input bits, they must still reach an agreement on a common output bit.

Broadcast + Gradecast $\Longrightarrow$ Byzantine agreement. To implement Broadcast, the sender first gradecasts its message M, and then the parties run Byzantine agreement to find out whether they all received as output grade 2. If this condition is met, they collectively output the message received during gradecast. Otherwise, they output $\perp$. Notably, if the sender is honest and possesses input message M, all honest parties receive M with a grade 2 during gradecast. Consequently, since they all have the same input for Byzantine agreement, they unanimously output M. In the event that the sender is corrupted and no honest party receives a grade 2, they uniformly input 0 for Byzantine agreement and unanimously agree on the output $\perp$. When at least one honest party has a grade 2, while others do not, the outcome of Byzantine agreement becomes uncertain. However, gradecast guarantees that all honest parties possess the same message (since there is an honest party with grade 2), resulting in the output potentially being either $\perp$ or M – but there is always an agreement.

Oblivious leader election (OLE) – See Section 2.1. Oblivious leader election (OLE) is a randomized protocol designed to elect a leader among the set of parties, ensuring the process remains oblivious to the adversary's manipulation. The primary objective is for all parties to reach a consensus on the identity of the elected leader. There are two possible outcomes: either all parties agree on the chosen leader or they do not. However, it is important to note that even if they do agree on the elected leader, there remains a possibility that the chosen leader is a corrupted party. The goal of OLE is to develop a protocol that elects an honest leader with a constant probability.

OLE $\Longrightarrow$ Byzantine agreement. Without getting into too many details, in the implementation of Byzantine agreement (see Protocol 7.1), the parties can essentially identify whether their current value agrees. If they do not have an agreement, then they try to agree on *some* value. Towards that end, they obliviously elect a leader, and verify whether they agree on the value the leader had provided (i.e., since the elected leader might be corrupted, it might report different messages as to what value it holds). Recall that the parties might not agree on the identity of

the leader, (in that case they would receive different values), or maybe the leader is corrupted and provided different values to different parties. In that case, the protocol repeats. Once an honest leader is elected then all parties reach an agreement in the next iteration. This random process leads to "expected constant number of rounds".

Verifiable secret sharing (VSS) – See Section 4. Verifiable secret sharing is a two-phase protocol, that allows a dealer to distribute some secret such that each party receives a share. In the second stage (the reconstruction phase), the parties can reconstruct the secret from their shares. For our discussion, VSS can be viewed as the information-theoretic equivalence of a commitment scheme. It provides hiding – the shares do not provide any information about the secret, and binding – once the sharing phase is concluded, the dealer cannot change its decision, and reconstruction is always guaranteed.

VSS $\Longrightarrow$ OLE. To implement an oblivious leader election, the idea is to run a "coin-tossing" protocol. Specifically, each party is associated with a random value, and the party with the largest party is the one elected. We cannot let each party choose its own value, as the adversary would then always pick to itself the largest number. Therefore, we let each party contribute to the value of some *hidden* and *random* subset of parties $\mathcal{M} \subset [n]$ of size $O(\mathsf{poly} \log n)$. That is, each party P_i provides the contribution $c_{i \to j}$ to each $j \in \mathcal{M}_i$. Then, the value of P_j is $c_j = \sum_{j \in \mathcal{M}_i} c_{i \to j}$. We choose the size of the set $\mathcal{M}_i$ to ensure that with sufficiently high probability each party P_j receives a contribution from some honest party, where the probability is taken over the randomness of all the parties that remain honest throughout the protocol. To prevent the adversary from picking its own values as a function of the honest parties' values, we use VSS as a commitment; The parties first commit to their values, and later, after all parties committed to their values, they reconstruct them and find who the leader is.

Multi-moderated VSS – See Section 5. A complication arises from the fact that VSS uses broadcast to reach an agreement on whether the shares of the dealer should be accepted. However, we cannot use broadcast here, as this would lead to a circular construction. This was addressed in previous works by substituting all broadcasts with gradecasts. This does not work directly and requires some more work since gradecast is too weak, and might lead to cases in which the parties do not necessarily agree on the leader. We do not get into the details in this overview, and just mention that the construction does not directly use VSS but rather a primitive called "multi-moderated VSS" (with no broadcast).