Asymptotically Free Broadcast in Constant Expected Time via Packed VSS

Ittai Abraham* Gilad Asharov† Shravani Patil‡ Arpita Patra§

Abstract

Broadcast is an essential primitive for secure computation. We focus in this paper on optimal resilience (i.e., when the number of corrupted parties t is less than a third of the computing parties n), and with no setup or cryptographic assumptions.

While broadcast with worst case t rounds is impossible, it has been shown [Feldman and Micali STOC'88, Katz and Koo CRYPTO'06] how to construct protocols with expected constant number of rounds in the private channel model. However, those constructions have large communication complexity, specifically $\mathcal{O}(n^2L + n^6 \log n)$ expected number of bits transmitted for broadcasting a message of length L. This leads to a significant communication blowup in secure computation protocols in this setting.

In this paper, we substantially improve the communication complexity of broadcast in constant expected time. Specifically, the expected communication complexity of our protocol is $\mathcal{O}(nL + n^4 \log n)$. For messages of length $L = \Omega(n^3 \log n)$, our broadcast has no asymptotic overhead (up to expectation), as each party has to send or receive $\mathcal{O}(n^3 \log n)$ bits. We also consider parallel broadcast, where n parties wish to broadcast L bit messages in parallel. Our protocol has no asymptotic overhead for $L = \Omega(n^2 \log n)$, which is a common communication pattern in perfectly secure MPC protocols. For instance, it is common that all parties share their inputs simultaneously at the same round, and verifiable secret sharing protocols require the dealer to broadcast a total of $\mathcal{O}(n^2 \log n)$ bits.

As an independent interest, our broadcast is achieved by a *packed verifiable secret sharing*, a new notion that we introduce. We show a protocol that verifies $\mathcal{O}(n)$ secrets simultaneously with the same cost of verifying just a single secret. This improves by a factor of n the state-of-the-art.

*VMWare Research. iabraham@vmware.com

†Department of Computer Science, Bar-Ilan University, Israel. Gilad.Asharov@biu.ac.il. Sponsored by the Israel Science Foundation (grant No. 2439/20), by JPM Faculty Research Award, by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.

‡Indian Institute of Science, Bangalore, India. shravanip@iisc.ac.in. Supported by DST National Mission on Interdisciplinary Cyber-Physical Systems (NM-ICPS) 2020-2025.

§Indian Institute of Science, Bangalore, India. arpita@iisc.ac.in. Supported by DST National Mission on Interdisciplinary Cyber-Physical Systems (NM-ICPS) 2020-2025, Google India Faculty Award 2020, and SERB MATRICS (Theoretical Sciences) Grant 2020-2023.

Contents

1 Introduction

A common practice in designing secure protocols is to describe the protocol in the broadcast-hybrid model, i.e., to assume the availability of a *broadcast channel*. Such a channel allows a distinguished party to send a message while guaranteeing that all parties receive and agree on the same message. Assuming the availability of a broadcast channel is reasonable only in a restricted setting, for instance, when the parties are geographically close and can use radio waves. In most settings, particularly when executing the protocol over the Internet, parties have to implement this broadcast channel over point-to-point channels.

The cost associated with the implementation of the broadcast channel is often neglected when designing secure protocols. In some settings, the implementation overhead is a real obstacle in practice. In this paper, we focus on the most demanding setting: **perfect security with optimal resilience**.

Perfect security means that the protocol cannot rely on any computational assumptions, and the error probability of the protocol is zero. Optimal resilience means that the number of parties that the adversary controls is bounded by $t < n/3$, where n is the total number of parties. This bound is known to be tight, as a perfectly-secure broadcast protocol tolerating $n/3$ corrupted parties or more is impossible to construct [LSP82, PSL80], even when a constant error probability is allowed [AN21].

Asymptotically-free broadcast. What is the best implementation of broadcast that we can hope for? For broadcasting an L bit message, consider the ideal trusted party that implements an "ideal broadcast". Since each party has to receive L bits, the total communication is $\mathcal{O}(nL)$. To avoid bottlenecks, we would also prefer *balanced* protocols where all parties have to communicate roughly the same number of bits, i.e., $\mathcal{O}(L)$, including the sender.

Regarding the 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]. Therefore, a protocol that runs in strict constant number of rounds is impossible to achieve. The seminal works of Rabin and Ben-Or [Rab83, Ben83] demonstrated that those limitations can be overcome by using randomization. We define *asymptotically-free broadcast* as a balanced broadcast protocol that runs in *expected* constant number of rounds and with (expected) communication complexity of $\mathcal{O}(nL)$.

There are, in general, two approaches for implementing broadcast in our setting. These approaches provide an intriguing tradeoff between communication and round complexity:

- **Low communication complexity, high number of rounds:** For broadcasting a single bit, the first approach [CW89, BGP92] requires $\mathcal{O}(n^2)$ bits of communication complexity, which is asymptotically optimal for any deterministic broadcast protocols [DR82], or in general, $\mathcal{O}(nL + n^2 \log n)$ bits for broadcasting a message of size L bits via a perfect broadcast extension protocol [Che21].[1] This comes at the expense of having $\Theta(n)$ rounds.
- **High communication complexity, constant expected number of rounds:** The second approach, originated by the seminal work of Feldman and Micali [FM88], followed by substantial improvements and simplifications by Katz and Koo [KK06], requires significant communication complexity of $\mathcal{O}(n^6 \log n)$ bits in expectation for broadcasting just a single bit,

[1] Broadcast extension protocols handle long messages efficiently at the cost of a small number of single-bit broadcasts.

or $\mathcal{O}(n^2L + n^6 \log n)$ bits for a message of L bits.[2] However, they work in *expected constant number of rounds.*

To get a sense of how the above translates to practice, consider a network with 200ms delay per round-trip (such a delay is relatively high, but not unusual, see [lat]), and $n = 300$. Using the first type of protocol, ≈ 300 rounds are translated to a delay of 1 minute. Then, consider for instance computing the celebrated protocol of Ben-Or, Goldwasser and Wigderson [BGW88] on an arithmetic circuit with depth 30. In each layer of the circuit the parties have to use broadcast, and thus the execution would take at least 30 minutes. The second type of protocols require at least $\Omega(n^6 \log n)$ bits of communication. The protocol is balanced and each party sends or receives $n^5 \log n$ bits ≈ 2.4 terabytes. Using 1Gbps channel, this is a delay of 5.4 hours. Clearly, both approaches are not ideal.

This current state of the affairs calls for the design of faster broadcast protocols and in particular, understanding better the tradeoff between round complexity and communication complexity.

Why perfect security? Our main motivation for studying broadcast is for perfectly secure multiparty computation. Perfect security provides the strongest possible security guarantee. It does not rely on any intractable assumptions and provides unconditional, quantum, and everlasting security. Protocols with perfect security remain adaptively secure (with some caveats [CDD+01, ACS22]) and secure under universal composition [KLR06]. Perfect broadcast is an essential primitive in generic perfectly secure protocols.

Even if we relax our goals and aim for statistical security only, the situation is not much better. Specifically, the best upper bounds that we have are in fact already perfectly secure [CW89, BGP92, KK06, Pat11, NRS+20, Che21]. That is, current statistically secure results do not help in achieving a better communication complexity vs round complexity tradeoff relative to the current perfect security results. We remark that in the computational setting, in contrast, the situation is much better. Asymptotically-free broadcast with $f < n/2$ can be achieved assuming threshold signatures and setup assumption in constant expected rounds and with $\mathcal{O}(n^2 + nL)$ communication [KK06, ADD+19, SBKN21].

1.1 Our Results

We provide a significant improvement in the communication complexity of broadcast with perfect security and optimal resilience in the presence of a *static adversary*. Towards that end, we also improve a pivotal building block in secure computation, namely, verifiable secret sharing (VSS). Our new VSS has an $O(n)$ complexity improvement that may be of independent interest. We present our results in a top-down fashion. Our main result is:

Theorem 1.1. *There exists a perfectly secure, balanced, broadcast protocol with optimal resilience, which allows a dealer to send L bits at the communication cost of $\mathcal{O}(nL)$ bits, plus $\mathcal{O}(n^4 \log n)$ expected bits. The protocol runs in constant expected number of rounds and assumes private channels.*

Previously, Katz and Koo [KK06] achieved $\mathcal{O}(n^2L)$ bits plus $\mathcal{O}(n^6 \log n)$ expected number of bits. For messages of size $L = \Omega(n^3 \log n)$ bits, the total communication of our protocol is $\mathcal{O}(nL)$

[2] Using broadcast extension of [NRS+20] we can bring the asymptotic cost to $\mathcal{O}(nL)+E(\mathcal{O}(n^7 \log n))$ bits. However, the minimum message size to achieve this $L = \Omega(n^6 \log n)$. This is prohibitively high even for $n = 100$.

bits. Thus, we say that our protocol is asymptotically free for messages of size $L = \Omega(n^3 \log n)$ bits. We recall that [KK06] together with [NRS+20] are also asymptotically free albeit only for prohibitively large value of L ($= \Omega(n^6 \log n)$). Table 1 compares our work to the state of the art in broadcast protocols.

To get a sense from a practical perspective, for broadcasting a single bit with $n = 300$, our protocol requires each party to send/receive roughly $n^3 \log n \approx 27$ MB (as opposed to ≈ 2.4 terabytes by [KK06]). Using a 1Gbps channel, this is 200ms. For broadcasting a message of size $\approx$ 27 MB, each party still has to send/receive roughly the same size of this message, and the broadcast is asymptotically free in that case.

Parallel composition of broadcast. In MPC, protocols often instruct the n parties to broadcast messages of the same length L in parallel at the same round. For instance, in the protocol of [BGW88], all parties share their input at the same round, and for verifying the secret, each party needs to broadcast $L = \mathcal{O}(n^2 \log n)$ bits.[3] In fact, the notion of parallel-broadcast goes back to the work of Pease et al. [PSL80]. We have the following extension to our main result:

Corollary 1.2. *There exists a perfectly-secure, balanced, parallel-broadcast protocol with optimal resilience, which allows n dealers to send messages of size L bits each, at the communication cost of $\mathcal{O}(n^2 L)$ bits, plus $\mathcal{O}(n^4 \log n)$ expected bits. The protocol runs in constant expected number of rounds.*

For message of size $L = \mathcal{O}(n^2 \log n)$ bits, which is common in MPC, our broadcast is asymptotically optimal. We obtain a cost of $\mathcal{O}(n^4 \log n)$ bits in expectation, with expected constant rounds. Note that each party receives $\mathcal{O}(nL)$ bits, and therefore $\mathcal{O}(n^2 L) = \mathcal{O}(n^4 \log n)$ bits is the best that one can hope for. Again, the protocol is balanced, which means that each party sends or receives only $\mathcal{O}(nL)$ bits.

For comparison, the other approach for broadcast based on [CW89, BGP92, Che21] requires total $\mathcal{O}(n^4 \log n)$ bits for this task, but with $\Theta(n)$ rounds. We refer again to Table 1 for comparison.

To get a practical sense of those complexities, when $n = 300$ and parties have to broadcast simultaneously messages of size L, our protocol is asymptotically optimal for $L = n^2 \log n \approx 90$KB.

Packed verifiable secret sharing. A pivotal building block in our construction, as well as perfectly secure multiparty protocols is *verifiable secret sharing* (VSS), originally introduced by Chor et al. [CGMA85]. It allows a dealer to distribute a secret to n parties such that no share reveal any information about the secret, and the parties can verify, already at the sharing phase, that the reconstruction phase would be successful.

To share a secret in the semi-honest setting, the dealer embeds its secret in a degree-t univariate polynomial, and it has to communicate $\mathcal{O}(n)$ field elements. In the malicious setting, the dealer embeds its secret in a bivariate polynomial of degree-t in both variables [BGW88, Fel88]. The dealer then has to communicate $\mathcal{O}(n^2)$ field elements to share its secret. An intriguing question is

[3] In fact, in each round of the protocol, each party performs $\mathcal{O}(n)$ verifiable secret sharings (VSSs), i.e., it has to broadcast $\mathcal{O}(n^3 \log n)$ bits. In [AAY21] it has been shown how to reduce it to $\mathcal{O}(1)$ VSSs per party, i.e., each party might have to broadcast $\mathcal{O}(n^2 \log n)$.

[4] Since the broadcast extension protocol of [Che21] requires $\mathcal{O}(n)$ rounds, combining [KK06] with [Che21] results in linear-round complexity and a worse communication complexity than what the second row ([CW89, BGP92] + [Che21]) provides.

Task	Reference	Total P2P (in bits)	Rounds
$1 \times \mathcal{BC}(L)$	[CW89, BGP92]	$\mathcal{O}(n^2 L)$	$\mathcal{O}(n)$
	[CW89, BGP92] + [Che21]	$\mathcal{O}(nL + n^2 \log n)$	$\mathcal{O}(n)$
	[KK06]	$\mathcal{O}(n^2 L) + E(\mathcal{O}(n^6 \log n))$	$E(\mathcal{O}(1))$
	[KK06] + [NRS+20]	$\mathcal{O}(nL) + E(\mathcal{O}(n^7 \log n))$	$E(\mathcal{O}(1))$
	Our work	$\boldsymbol{\mathcal{O}(nL) + E(\mathcal{O}(n^4 \log n))}$	$\boldsymbol{E(\mathcal{O}(1))}$
$n \times \mathcal{BC}(L)$	[CW89, BGP92]	$\mathcal{O}(n^3 L)$	$\mathcal{O}(n)$
	[KK06]	$\mathcal{O}(n^3 L) + E(\mathcal{O}(n^6 \log n))$	$E(\mathcal{O}(1))$
	[KK06] + [NRS+20][4]	$\mathcal{O}(n^2 L) + E(\mathcal{O}(n^7 \log n))$	$E(\mathcal{O}(1))$
	Our work	$\boldsymbol{\mathcal{O}(n^2 L) + E(\mathcal{O}(n^4 \log n))}$	$\boldsymbol{E(\mathcal{O}(1))}$

Table 1: Comparison of communication complexity of our work with the state-of-the-art broadcast.
$1 \times \mathcal{BC}(L)$ refers to the task of a single dealer broadcasting a L-element message.
$n \times \mathcal{BC}(L)$ refers to the task of n dealers broadcasting a L-element message in parallel.

whether this gap between the semi-honest (where the dealer has to encode its secret in a structure of size $\mathcal{O}(n)$) and the malicious setting (where the dealer has to encode its secret in a structure of size $\mathcal{O}(n^2)$) is necessary. While we do not answer this question, we show that the dealer can pack $\mathcal{O}(n)$ secrets, simultaneously in one bivariate polynomial. Then, it can share it at the same cost as sharing a single VSS, achieving an overhead of $\mathcal{O}(n)$ per secret. We show:

Theorem 1.3. *Given a synchronous network with pairwise private channels and a broadcast channel, there exists a perfectly secure packed VSS protocol with optimal resilience, which has a communication complexity of $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits broadcast for sharing $\mathcal{O}(n)$ secret field elements (i.e., $\mathcal{O}(n \log n)$ bits) in strict $\mathcal{O}(1)$ rounds. The optimistic case (where all the parties behave honestly) does not use the broadcast channel in the protocol.*

The best previous results achieve $\mathcal{O}(n^3 \log n)$ (point-to-point and broadcast) for sharing $\mathcal{O}(n)$ secret elements [BGW88, Fel88, KKK08, AKP20], this is an improvement by a factor of n in communication complexity.

Packing k secrets into one polynomial is a known technique, proposed by Franklin and Yung [FY92]. It was previously used in Shamir's secret sharing scheme. However, it comes with the following price: While Shamir's secret sharing allows protecting against even $n-1$ corrupted parties, packing k secrets in one polynomial achieves privacy against only $n-k-1$ parties. In the malicious case, VSS of a single secret is possible only when the number of corruption satisfies $t < n/3$. The idea of packing many secrets without trading off the allowed threshold of corruption has been explored by Damgård et al. [DDGN14]. However, it is achieved at the expense of having $\mathcal{O}(n)$ rounds. In contrast, our packed verifiable secret sharing enables packing $\mathcal{O}(n)$ secrets while keeping the threshold exactly the same and ensuring $\mathcal{O}(1)$ round complexity. Compared to a constant round VSS of a single secret, we obtain packed secret sharing completely for free (up to small hidden constants in the $\mathcal{O}$ notation of the above theorem). Lastly, the same result as ours is achieved in the asynchronous setting with the optimal resilience of $t < n/4$ in [CP16, CHP13].

Optimal gradecast for $\Omega(n^2)$ messages. Another building block that we improve along the way is gradecast. Gradecast is a relaxation of broadcast introduced by Feldman and Micali [FM88] ("graded-broadcast"). It allows a distinguished dealer to transmit a message, and each party outputs the message it receives together with a grade $g \in \{0, 1, 2\}$. If the dealer is honest, all

honest parties receive the same message and grade 2. If the dealer is corrupted, but some honest party outputs grade 2, it is guaranteed that all honest parties output the same message (though some might have grade 1 only). We show that:

Theorem 1.4. *There exists a perfectly secure gradecast protocol with optimal resilience, which allows a party to send a message of size L bits with a communication cost of $\mathcal{O}(nL + n^3 \log n)$ bits and in $\mathcal{O}(1)$ rounds. The protocol is balanced.*

Note that this result is optimal when $L = \Omega(n^2 \log n)$ bits as each party has to receive L bits even in an ideal implementation. Previously, the best gradecast protocol in the perfect security setting [FM88] required $\mathcal{O}(n^2 L)$ bits of communication.

1.2 Applications and Discussions

Applications: Perfect secure computation. We demonstrate the potential speed up of protocols in perfect secure computation using our broadcast. There are, in general, two lines of works in perfectly secure MPC, resulting again in an intriguing tradeoff between round complexity and communication complexity.

The line of work [BGW88, CCD88, GRR98, CDM00, ALR11, AAY21] achieves constant round per multiplication and round complexity of $\mathcal{O}(\mathsf{depth}(C))$, where C is the arithmetic circuit that the parties jointly compute. The communication complexity of those protocols results in $\mathcal{O}(n^3|C|\log n)$ bits over point-to-point channels in the optimistic case, and an additional $\mathcal{O}(n^3|C|\log n)$ bits over the broadcast channel in the pessimistic case (recall that this means that each party has to send or receive a total of $\mathcal{O}(n^4|C|\log n)$ bits). In a nutshell, the protocol requires each party to perform $\mathcal{O}(1)$ VSSs in parallel for each multiplication gate in the circuit, and recall that in each VSS the dealer broadcasts $\mathcal{O}(n^2 \log n)$ bits. This is exactly the setting in which our parallel broadcast gives asymptotically free broadcast (Corollary 1.2). Thus, we get a protocol with a total of $\mathcal{O}(n^4|C|\log n)$ bits (expected) and expected $\mathcal{O}(\mathsf{depth}(C))$ rounds over point-to-point channels. Previously, using [KK06], this would have been resulted in expected $\mathcal{O}(n^6|C|\log n)$ communication complexity with $\mathcal{O}(\mathsf{depth}(C))$ rounds.

Another line of work [HMP00, BTH08, GLS19] in perfectly-secure MPC is based on the *player elimination* framework (introduced by Hirt and Maurer and Przydatek [HMP00]). Those protocol identify parties that may misbehave and exclude them from the execution. Those protocols result in a total of $\mathcal{O}((n|C| + n^3)\log n)$ bits over point-to-point channels, and $\mathcal{O}(n \log n)$ bits over the broadcast channel. However, this comes at the expense of $\mathcal{O}(\mathsf{depth}(C) + n)$ rounds. This can be compiled to $\mathcal{O}((n|C| + n^3)\log n)$ communication complexity with $\mathcal{O}(n^2 + \mathsf{depth}(C))$ rounds using [CW89, BGP92], or to $\mathcal{O}((n|C| + n^7)\log n)$ communication complexity with $\mathcal{O}(n + \mathsf{depth}(C))$ rounds (expected) using [KK06]. Using our broadcast, the communication complexity is $\mathcal{O}((n|C| + n^5)\log n)$ with $\mathcal{O}(n + \mathsf{depth}(C))$ rounds (expected). We remark that in many setting, a factor n in round complexity should not be treated the same as communication complexity. Roundtrips are slow (e.g., 200ms delay for each roundtrip), whereas communication channels can send relatively large messages fast (1 or even 10Gbps).

On sequential and parallel composition of our broadcast. Like Feldman and Micali [FM88] and Katz and Koo [KK06] (and any $o(t)$-round expected broadcast protocol), our protocol cannot provide simultaneous termination. Sequentially composing such protocols is discussed in Lindell, Lysyanskaya and Rabin [LLR02], Katz and Koo [KK06] and Cohen et al. [CCGZ19]. Regarding

parallel composition, unlike the black-box parallel composition of broadcasts studied by Ben-Or and El-Yaniv [BE03], we rely on the idea of Fitzi and Garay [FG03] that applies to OLE-based protocols. The idea is that multiple broadcast sub-routines are run in parallel when only a single election per iteration is required for all these sub-routines. This reduces the overall cost and also guarantees that parallel broadcast is also constant expected number of rounds.

Modeling broadcast functionalities. We use standalone, simulation-based definition as in [Can00]. The standalone definition does not capture rounds in the ideal functionalities, or the fact that there is no simultaneous termination. The work of Cohen et al. [CCGZ19] shows that one can simply treat the broadcast without simultaneous termination as an ideal broadcast as we provide (which, in particular, has simultaneous and deterministic termination). Moreover, it allows compiling a protocol using deterministic-termination hybrids (i.e., like our ideal functionalities) into a protocol that uses expected-constant-round protocols for emulating those hybrids (i.e,. as our protocols) while preserving the expected round complexity of the protocol. We remark that in order to apply the compiler of [CCGZ19], the functionalities need to follow a structure of (1) input from all parties; (2) leakage to the adversary; (3) output. For simplicity, we did not write our functionalities using this specific format, but it is clear that our functionalities can be written in this style.

Our broadcast with strict-polynomial run time. Protocols in constant expected number of rounds might never terminate (although, with extremely small probability). Our protocols can be transformed into a protocol that runs in strict polynomial time using the approach of Goldreich and Petrank [GP90]: Specifically, after $\mathcal{O}(n)$ attempts to terminate, the parties can run the $\mathcal{O}(n)$ rounds protocol with guaranteed termination. See also [CCGZ19].

1.3 Related Work

We review the related works below. Error-free byzantine agreement and broadcast are known to be possible only if $t < n/3$ holds [LSP82, PSL80]. Moreover, Fischer and Lynch [FL82] showed a lower bound of $t + 1$ rounds for any deterministic byzantine agreement protocol or broadcast protocol. Faced with this barrier, Rabin [Rab83] and Ben-Or [Ben83] independently studied the effect of randomization on round complexity, which eventually culminated into the work of Feldman and Micali [FM97] who gave an expected constant round protocol for byzantine agreement with optimal resilience. Improving over this work, the protocol of [KK06] requires a communication of $\mathcal{O}(n^2L + n^6 \log n)$ for a message of size L bits, while achieving the advantage of expected constant rounds. In regards to the communication complexity, Dolev and Reischuk [DR85] established a lower bound of n^2 bits for deterministic broadcast or agreement on a single bit. With a round complexity of $\mathcal{O}(n)$, [CW89, BGP92] achieve a broadcast protocol with a communication complexity of $\mathcal{O}(n^2)$ bits.

We quickly recall the state of the art perfectly-secure broadcast extension protocols. Recall that these protocols aim to achieve the optimal complexity of $\mathcal{O}(nL)$ bits for sufficiently large message size L and utilize a protocol for bit broadcast. The protocol of [Pat11, GP21] communicates $\mathcal{O}(nL)$ bits over point-to-point channels and $\mathcal{O}(n^2)$ bits through a bit-broadcast protocol. The work of [NRS+20] improves the number of bits sent through a bit-broadcast protocol to $\mathcal{O}(n)$ bits. Both these extension protocols are constant round. The recent work of [Che21] presents a protocol

that communicates $\mathcal{O}(nL + n^2 \log n)$ bits over point-to-point channels and a single bit through a bit-broadcast protocol. However, the round complexity of this protocol is $\mathcal{O}(n)$.

A few other works in different settings are given below. The notion of parallel broadcast was recently explored by Tsimos et al. [TLP20] in the dishonest majority setting under cryptographic assumptions. Hirt and Zikas [HZ10] studied the adaptive security of broadcast in the UC model, and improved the resilience of the ideal functionality to adaptive corruptions.

2 Technical Overview

We describe the high-level overview of our techniques. We start with our improved broadcast in Section 2.1, and then describe packed VSS in Section 2.2, followed by the gradecast protocol in Section 2.3. To aid readability, we summarize our different primitives and the relationship between them in Figure 1. In each one of the those primitives we improve over the previous works.

Primitive	P2P	Broadcast	Reference	Remarks
Broadcast	$\mathcal{O}(nL) + E(\mathcal{O}(n^4 \log n))$	–	Section 8.2	L bit message
Byzantine Agreement	$\mathcal{O}(n^2) + E(\mathcal{O}(n^4 \log n))$	–	Section 8.1	–
Gradecast	$\mathcal{O}(nL + n^3 \log n)$	–	Section 5	L bit message
Oblivious Leader Election	$\mathcal{O}(n^4 \log n)$	–	Section 7	–
Multi-moderated VSS	$\mathcal{O}(n^4 \log n)$	–	Section 6	Sharing $\mathcal{O}(n)$ values
Packed VSS (w. Gradecast)	$\mathcal{O}(n^3 \log n)$	–	Section 4	Sharing $\mathcal{O}(n)$ values
Packed VSS (w. Broadcast)	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n^2 \log n)$	Section 4	Sharing $\mathcal{O}(n)$ values

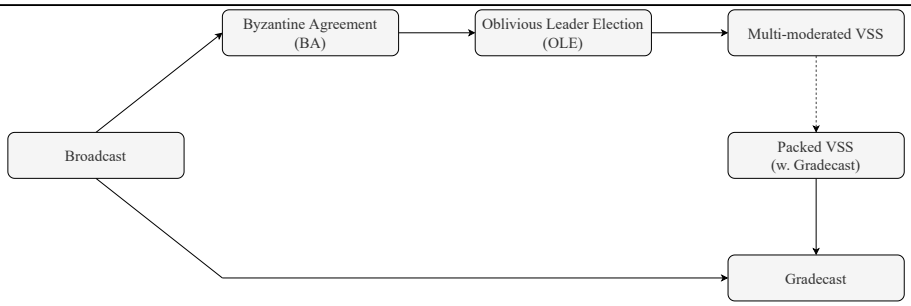

Figure 1: Roadmap of our building blocks. All lines are compositions, except for the line from Multi-moderated VSS to Packed VSS, which is a white-box modification.

2.1 Improved Broadcast in Constant Expected Rounds

Our starting point is a high-level overview of the broadcast protocol of Katz and Koo [KK06], which simplifies and improves the construction of Feldman and Micali [FM88]. Following the approach of Turpin and Coan [TC84] for broadcast extension closely, broadcast can be reduced to two primitives: Gradecast and Byzantine agreement.

1. **Gradecast:** A gradecast is a relaxation of broadcast, where a distinguished dealer transmits a message, and parties output the message together with a grade. If the dealer is honest, all honest parties are guaranteed to output the dealer's message together with a grade 2. Moreover, if the dealer is corrupted and one honest party outputs grade 2, then it is guaranteed that all other honest parties also output the same message, though maybe with a grade 1. Looking ahead, we show how to improve gradecast of message of length L bits from $\mathcal{O}(n^2 L)$

bits to $\mathcal{O}(nL + n^3 \log n)$ bits, which is optimal for messages of $L = \Omega(n^2 \log n)$ bits. We overview our construction in Section 2.3.

2. **Byzantine agreement:** In Byzantine agreement all parties hold some bit as input, and all of them output a bit at the end of the protocol. If all honest parties hold the same value, then it is guaranteed that the output of all parties would be that value. Otherwise, it is guaranteed that the honest parties would agree and output the same (arbitrary) bit.

To implement broadcast, the dealer gradecasts its message M and then the parties run Byzantine agreement (BA) on the grade they received (using 1 as input when the grade of the gradecast is 2, and 0 otherwise). Then, if the output of the BA is 1, each party outputs the message it received from the gradecast, and otherwise it outputs $\perp$.

If the dealer is honest, then all honest parties receive grade 2 in the gradecast, and all would agree in the BA that the grade is 2. In that case, they all output M. If the dealer is corrupted, and all honest parties received grade 0 or 1 in the gradecast, they would all use 0 in the Byzantine agreement, and all would output $\perp$. The remaining case is when some honest parties receives grade 2 in the gradecast, and some receive 1. However, once there is a single honest party that received grade 2 in the gradecast, it is guaranteed that all honest parties hold the same message M. The Byzantine agreement can then go either way (causing all to output M or $\perp$), but agreement is guaranteed.

Oblivious leader election. It has been shown that to implement a Byzantine agreement (on a single bit), it suffices to obliviously elect a leader, i.e., a random party among the parties. In a nutshell, a Byzantine agreement proceeds in iterations, where parties exchange the bits they believe that the output should be and try to see if there is an agreement on the output. When there is no clear indication of which bit should be the output, the parties try to see if there is an agreement on the output bit suggested by the elected leader. A corrupted leader might send different bits to different parties. However, once an honest leader is elected, it must have sent the same bit to all parties. In that case the protocol guarantees that all honest parties will agree in the next iteration on the output bit suggested by the leader, and halt.

Oblivious leader election is a protocol where the parties have no input, and the goal is to agree on a random value in $\{1, \ldots, n\}$. It might have three different outcomes: (1) All parties agree on the same random index $j \in \{1, \ldots, n\}$, and it also holds that P_j is honest; this is the preferable outcome; (2) All parties agree on the same index $i \in \{1, \ldots, n\}$, but P_i is corrupted; (3) The parties do not agree on the index of the party elected. The goal is to achieve the outcome (1) with constant probability, say $\geq 1/2$. Recall that once outcome (1) occurs then the Byzantine agreement succeeds. Achieving outcome (1) with constant number of rounds and with constant probability implies Byzantine agreement with constant expected number of rounds.

The key idea to elect a leader is to randomly choose, for each party, some random value c_i. Then, the parties choose an index j of the party for which c_j is minimal. To do that, we cannot let each party P_j choose its random value c_j, as corrupted parties would always choose small numbers to be elected. Thus, all parties contribute to the random value associated with each party. That is, each party P_k chooses $c_{k \to j} \in \{1, \ldots, n^4\}$ and the parties define $c_j = \sum_{k=1}^{n} c_{k \to j} \bmod n^4$ as the random value associated with P_j. This guarantees that each value c_j is uniform.

However, just as in coin-tossing protocols, a party cannot publicly announce its random choices, since then it would allow a rushing adversary to choose its random values as a function of the announced values. This is prevented by using *verifiable secret sharing*. Verifiable secret sharing

provides *hiding* – given t shares, it is impossible to determine what is the secret, and *binding* – at the end of the sharing phase, the dealer cannot change the secret, and reconstruction is guaranteed. The parties verifiably share their random values $c_{k\to j}$ for every k, j. After all parties share their values, it is safe to reconstruct the secret, reveal the random values, and elect the leader based on those values.

A problem: VSS uses a broadcast channel. A problem with the above solution is that protocols for VSS use a broadcast channel to reach an agreement on whether or not to accept the dealer's shares. Yet, the good news is that broadcast is used only during the sharing phase. Replacing each broadcast with a gradecast does not suffice since honest parties do not necessarily agree on the transmitted messages when corrupted senders gradecast messages. This leads to the notion of "moderated VSS", where the idea is to have a party that is responsible for all broadcasted message. Specifically, now there are two distinguished parties: a dealer P_k and a moderator P_j. The parties run the VSS where P_k is the dealer; whenever a participant has to broadcast a message m, it first gradecasts it, and then the moderator P_j has to gradecast the message it received. Each party can then compare between the two gradecasted messages; however, the parties proceed the execution while using the message that the moderator had gradecasted as the message that was broadcasted. At the end of the execution, each party outputs together with the shares, a grade for the moderator in $\{0, 1\}$. For instance, if the moderator ever gradecasted some message and the message was received by some party P_i with grade ≤ 1, then the grade that P_i gives the moderator is 0 — P_i cannot know whether other parties received the same message at all. The idea is that honest parties might not necessarily output the same grade, but if there is one honest party that outputs grade 1, it is guaranteed that the VSS was successful, and we have binding. Moreover, if the moderator is honest, then all honest parties would give it grade 1.

Going back to leader election, the value $c_{k\to j}$ is distributed as follows: the parties run a VSS where P_k is the dealer and P_j is the moderator. After all values of all parties were shared (i.e., all parties committed to the values $c_{k\to j}$), each party defines for each moderator P_j the value $c_j = \sum_{k=1}^{n} c_{k\to j}$. If the grade of P_j was not 1 in all its executions as a moderator, then replace $c_j = \infty$. Each party elects the party P_ℓ for which c_ℓ is minimal.

If the moderator P_j is honest, then for both honest and corrupted dealer P_k, the VSS would end up with agreement, and all honest parties would give P_j grade 1 as a moderator. The value $c_j = \sum_{k=1}^{n} c_{k\to j} \bmod n^4$ would be the same for all honest parties, and it must distribute uniformly as honest dealers contributed random values in this sum. Likewise, if a moderator P_j is corrupted but some honest party outputs grade 1 in all executions where P_j served as a moderator, then the value $c_j = \sum_{k=1}^{n} c_{k\to j} \bmod n^4$ must be the same for all honest parties, and it also must be random, as honest dealers contributed random values. There might be no agreement if some honest parties gave grade 1 for that moderator, while others did not and defined $c_j = \infty$. In that case, we might not have an agreement on the elected leader. However, it is guaranteed that the value c_j is distributed uniformly. Thus, the inconsistency is bounded with constant probability (roughly $t/n \leq 1/3$).

Our improvements. As noticed above, each party participates as the dealer in n executions, and as the role of the moderator in n executions. Thus, we have a total of n^2 executions of VSS. First, we show a new protocol that enables a dealer to pack $\mathcal{O}(n)$ secrets at the cost of just one

VSS (assuming broadcast), called packed VSS (see an overview in Section 2.2). For leader election, we have to replace the broadcast in the packed VSS with a gradecast (with a moderator).

However, we cannot just pack all the $\mathcal{O}(n)$ values $c_{k\to j}$ where P_k is the dealer in one instance of a VSS with a moderator since each one of the secrets corresponds to a different moderator. We, therefore, introduce a new primitive which is called "Multi-moderated packed secret sharing": The dealer distributes $\mathcal{O}(n)$ values, where each corresponds to a different moderator, and have all parties serve as moderator in one shared execution of a VSS.

More precisely, the packed VSS uses several invocations of broadcasts in the sharing phase, just as a regular VSS. Until the very last round, the dealer also serves as the moderator within each of those broadcasts. In the last round, there is a vote among the parties whether accept or reject the dealer, where the vote is supposed to be performed over the broadcast channel. At this point, the execution is forked to $\mathcal{O}(n)$ executions. Each corresponds to a different moderator, where the moderator moderates just the last round's broadcasts. The idea is that the vast majority of the computation is shared between all $\mathcal{O}(n)$ executions, thus the additional cost introduced for each moderator is small. This allows us to replace all n executions where P_i serves as a dealer with just one execution where P_i is the dealer and other $\mathcal{O}(n)$ parties are moderators at the same time.

Another obstacle worth mentioning is that within multi-moderated packed VSS, the dealer broadcasts $\mathcal{O}(n^2 \log n)$ bits, whereas other participant broadcasts at most $\mathcal{O}(n \log n)$ bits. Our gradecast is not optimal for this message size, and thus when replacing those broadcasts with gradecasts, the overall cost would be $\mathcal{O}(n^5 \log n)$. We can do better by considering all the multi-moderated VSSs in parallel. Each party then participates in $\mathcal{O}(1)$ executions as a dealer and in $\mathcal{O}(n)$ executions as a participant. Therefore, each party has to broadcast $\mathcal{O}(n^2 \log n)$ bits in all invocations of multi-moderated packed VSS combined ($\mathcal{O}(n^2 \log n)$ bits when it serves as a dealer, and $(n-1) \times \mathcal{O}(n \log n)$ when it serves as a participant). For that size of messages, our gradecast is optimal.

To conclude, to obtain our broadcast, we build upon [FM88, KK06] and introduce: (1) an optimal gradecast protocol for $\Omega(n^2 \log n)$ messages which is used twice – for gradecasting the message before running the Byzantine agreement and within the Byzantine agreement as part of the VSSs; (2) a novel multi-moderated packed secret sharing, which is based on a novel packed VSS protocol; (3) carefully combine all the $\mathcal{O}(n)$ invocations of multi-moderated packed secret sharing to amortize the costs of the gradecasts.

When comparing to the starting point of $\mathcal{O}(n^2 L)$ plus $E(\mathcal{O}(n^6 \log n))$ of [KK06], the improved gradecast allows us to reduce the first term to $\mathcal{O}(nL)$, for large enough messages. Regarding the second term, packing $\mathcal{O}(n)$ values in the VSS reduces one n factor, and the improved gradecast within the VSS reduces another n factor. Overall this brings us to $\mathcal{O}(nL)$ plus $E(\mathcal{O}(n^4 \log n))$.

2.2 Packed Verifiable Secret Sharing

Our packed verifiable secret sharing protocol is the basis of the multi-moderated VSS. We believe that it will find applications in future constructions of MPC protocols, and is of independent interest. Communication cost wise, the best-known constant-round perfect VSS sharing one secret is $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels in the optimistic case, and additional $\mathcal{O}(n^2 \log n)$ bits over the broadcast channel in the pessimistic case [BGW88, GIKR01, AL17]. Here, we retain the same cost, yet "pack" $t+1$ secrets in one bivariate polynomial and generate $t+1$ independent Shamir-sharings at one go. We remark that in asynchronous setting, with the optimal threshold of $t < n/4$, this goal has been achieved in [CP16, CHP13].

Sharing more secrets at one go. Our goal is to generate Shamir-sharing of $t+1$ secrets, $s_{-t}, \ldots, s_0$, at once. Denoting Shamir-sharing of a secret s by $[s]$, our goal is to produce $[s_{-t}], \ldots, [s_0]$ using a single instance of a VSS. For this, the dealer chooses a degree-$(2t,t)$ bivariate polynomial[5] $S(x,y)$ such that $S(l,0) = s_l$ for each $l \in \{-t, \ldots, 0\}$. We set $f_i(x) = S(x,i)$ of degree $2t$ and $g_i(y) = S(i,y)$ of degree-t and observe that for every i,j it holds that $f_i(j) = S(j,i) = g_j(i)$. The goal of the verification part is that each P_i will hold $f_i(x)$ and $g_i(y)$ on the same bivariate polynomial $S(x,y)$. Then, each degree-t univariate polynomial $g_l(y)$ for $l \in \{-t, \ldots, 0\}$ is the standard Shamir-sharing of s_l amongst the parties. Once the shares of the parties are consistent, each party P_i can locally compute its share on $g_l(y)$ as $g_l(i) = f_i(l)$.

Our protocol is a strict improvement of [AAY21]. Specifically, the work of [AAY21] considers the VSS protocol of [BGW88] when the dealer uses a $(2t,t)$-polynomial instead of a degree-(t,t) polynomial. It observes that by minor modifications, the protocol still provides weak verifiability even though the sharing is done on a higher degree polynomial. By "weak", we mean that the reconstruction phase of the polynomial might fail in the case of a corrupted dealer. Nevertheless, the guarantee is that the reconstruction phase would either end up successfully reconstructing $S(x,y)$, or $\perp$, and whether it would succeed or not depends on the adversary's behavior. In contrast, in a regular ("strong") VSS, reconstruction is always guaranteed.

The work of [AAY21] utilizes this primitive to improve the efficiency of the degree-reduction step of the BGW protocol. However, this primitive is weak and does not suffice for most applications of VSS. For instance, it cannot be used as a part of our leader election protocol: The adversary can decide whether the polynomial would be reconstructed or not. Thus there is no "binding", and it can choose, adaptively and based on the revealed secrets of the honest parties, whether the reconstruction would be to the secret values or some default values. As such, it can increase its chance of being elected.

Our work: achieving strong binding. In our work, we show how to achieve strong binding. We omit the details in this high-level overview of achieving weak verifiability of [AAY21] secret sharing while pointing out that the protocol is a variant of the VSS protocol of [BGW88]. For our discussion, the protocol reaches the following stage: If the dealer is not discarded, then there is a CORE of $2t+1$ parties that hold shares of a unique bivariate polynomial $S(x,y)$, and this set of parties is public and known to all (it is determined based on votes performed over the broadcast channel). Each party P_i in CORE holds two univariate shares $f_i(x) = S(x,i)$ of degree-$2t$ and $g_i(y) = S(i,y)$ of degree-t. Each party P_j for $j \notin$ CORE holds a polynomial $g_j(y) = S(j,y)$, where some of those polynomials are also public and were broadcasted by the dealer. In case the dealer is honest, then all honest parties are part of CORE, whereas if the dealer is corrupted, then it might be that only $t+1$ honest parties are part of CORE. To achieve strong binding, the dealer has to provide shares for parties outside CORE, publicly, and in a constant number of rounds.

The first step is to make all the polynomials $g_j(y)$ for each $j \notin$ CORE public. This is easy, since each such polynomial is of degree t. The dealer can broadcast it, and the parties in CORE vote whether to accept. If there are no $2t+1$ votes to accept, then the dealer is discarded. Since the shares of the honest parties in CORE are consistent and define a unique $(2t,t)$-bivariate polynomial $S(x,y)$, the dealer cannot publish any polynomial $g_j(y)$ which is not $S(j,y)$. Any polynomial $g'_j(y) \neq S(j,y)$ can agree with at most t points with $S(j,y)$ and thus it would receive at most t

[5] We call a bivariate polynomial where the degree in x is $2t$ and in y is t, i.e., $S(x,y) = \sum_{i=0}^{2t} \sum_{j=0}^{t} a_{i,j} x^i y^j$ as a $(2t,t)$-bivariate polynomial.

votes of honest parties in CORE, i.e., it cannot reach $2t+1$ votes.

The next step is to make the dealer also publicize the shares $f_j(x)$ for each $j \notin \mathsf{CORE}$. This is more challenging since each $f_j(x)$ is of degree-$2t$, and therefore achieving $2t+1$ votes is not enough, as t votes might be false. Therefore, the verification is more delicate:

1. First, the parties in CORE have to vote OK on the f-polynomials that the dealer publishes. If there are less than $2t+1$ votes, the dealer is discarded.
2. Second, for each party P_j in CORE that did not vote OK, the dealer is required to publish its $g_j(y)$ polynomial. The parties in CORE then vote on the revealed polynomials as in the first step of boosting from weak to strong verification.

To see why this works, assume that the dealer tries to distribute a polynomial $f'_j(x) \neq S(x,j)$. Then, there must exist an honest party such that its share does not agree with $f'_j(x)$. If $f'_j(x)$ does not agree with shares that are public, then it would be immediately discarded. If $f'_j(x)$ does not agree with a share of an honest party P_k that is part of CORE, then $g_k(y)$ would become public in the next round, and the dealer would be publicly accused. The dealer cannot provide a share $g_k(y) \neq S(k,y)$ for the same reason as the first step of boosting from weak to strong VSS. At the end of this step we have that all honest parties are either part of CORE and their shares are private, or they are not in CORE and their shares are public. Overall, all honest parties hold shares on the bivariate polynomial $S(x,y)$. We refer to section 4 for the formal protocol description.

2.3 Optimal Gradecast

A crucial building block in our construction is gradecast. We show how to implement gradecast of a message of length L bits using total communication of $\mathcal{O}(n^3 \log n + nL)$ bits. For this overview, we just deal with the case where the dealer is honest and show that all honest parties output the message that the dealer gradecasted with grade 2. We leave the case of a corrupted dealer to the relevant section (Section 5).

Data dissemination. Our construction is inspired in part by the data dissemination protocol of [DXR21], while we focus here on the synchronous settings. In the task of data dissemination, $t+1$ honest parties hold as input the same input M, while other honest parties hold the input $\perp$, and the goal is that all honest parties receive the same output M in the presence of t corrupted parties. In our protocol, assume for simplicity messages of size $(t+1)^2$ field elements (i.e., a degree-(t,t) bivariate polynomial). Data dissemination can be achieved quite easily: (1) Each honest party sends to each party P_j the univariate polynomials $S(x,j), S(j,y)$. (2) Once a party receives $t+1$ messages with the same pair of univariate polynomials, it forwards those polynomials to all others. An adversary might send different polynomials, but it can never reach plurality $t+1$. (3) After all the honest parties forwarded their polynomials to the others, we are guaranteed that each party holds $2t+1$ correct shares of S and at most t incorrect shares. Each party can reconstruct S efficiently using Reed Solomon decoding. Note that this procedure requires the transmission of $\mathcal{O}(n^3 \log n)$ bits overall. Therefore, our goal in the gradecast protocol is to reach a state where $t+1$ honest parties hold shares of the same bivariate polynomial.

Gradecast. For the sake of exposition, we first describe a simpler protocol where the dealer is computationally unbounded, and then describe how to make the dealer efficient. Again, assume

that the input message of the dealer is encoded as a bivariate polynomial $S(x, y)$. The dealer sends the entire bivariate polynomial to each party. Then, every pair P_i and P_j exchange the polynomials $S(x, i), S(i, y), S(x, j), S(j, y)$. The two parties check whether they agree on those polynomials or not. If P_i sees that the polynomials it received from P_j are the same as it received from the dealer, then it adds j to a set Agreed_i. The parties then send their sets Agreed_i to the dealer, who defines an undirected graph where the nodes are the set $\{1, \ldots, n\}$ and an edge $\{i, j\}$ exists if and only if $i \in \mathsf{Agreed}_j$ and $j \in \mathsf{Agreed}_i$. The dealer then (inefficiently) finds a maximal clique $K \subseteq \{1, \ldots, n\}$ of at least $2t + 1$ parties and gradecasts K to all parties using a naïve gradecast protocol of [FM88, KK06] (note that this is a gradecast of case $\mathcal{O}(n^2 L)$ with $L = \mathcal{O}(n \log n)$). A party P_i is happy if: (1) $i \in K$; (2) it received the gradecast message of the dealer with grade 2; and (3) $K \subseteq \mathsf{Agreed}_i$. The parties then proceed to data dissemination protocol.

The claim is that if the dealer is honest, then at least $t + 1$ honest parties are happy, and they all hold the same bivariate polynomial. This is because the set of honest parties defines a clique of size $2t + 1$, and any clique that the honest dealer finds of cardinality $2t + 1$ must include at least $t + 1$ honest parties. The result of the data dissemination protocol is that all honest parties output S. If the dealer is corrupted, we first claim that all honest parties that are happy must hold the same bivariate polynomial. Any two honest parties that are happy must be part of the same clique K that contains at least $t + 1$ honest parties, and all honest parties in that clique must agree with each other (all see the same clique K defined by the dealer, and verified that they agreed with each other). The univariate polynomials exchanged between those $t + 1$ honest parties define a unique bivariate polynomial. Again, data dissemination would guarantee that all honest parties would output that bivariate polynomial.

On making the dealer efficient. To make the dealer efficient, we rely on a procedure that finds an approximation of a clique, known as the STAR technique, introduced by [Can93]. In the technical section, we show how we can use this approximation of a clique, initially introduced for the case of $t < n/4$, to the much more challenging scenario of $t < n/3$. We refer to Section 5 for the technical details.

Organization. The rest of the paper is organized as follows. In Section 3 we provide preliminaries and notations. In Section 4 we describe our packed verifiable secret sharing, followed by our gradecast in Section 5. We then proceed to multi-moderated packed secret sharing (Section 6), oblivious leader election (Section 7) and we conclude with our broadcast protocol in Section 8.

3 Preliminaries

We consider a synchronous network model where the parties in $\mathcal{P} = \{P_1, \ldots, P_n\}$ are connected via pairwise private and authenticated channels. Additionally, for some of our protocols we assume the availability of a broadcast channel, which allows a party to send an identical message to all the parties. One of the goals of this paper is to implement such a broadcast channel over the pairwise private channels, and we mention explicitly for each protocol whether a broadcast channel is available or not. The distrust in the network is modelled as a *computationally unbounded* active adversary $\mathcal{A}$ which can maliciously corrupt up to t out of the n parties during the protocol execution and make them behave in an arbitrary manner. We prove security in the stand-alone model for

a static adversary. Owing to the results of [CDD+01], this guarantees adaptive security with inefficient simulation. We derive universal composability [Can01] for free using [KLR06].

Our protocols are defined over a finite field $\mathbb{F}$ where $|\mathbb{F}| > n + t + 1$. We consider two sets of n and $t+1$ distinct elements from $\mathbb{F}$ publicly known to all the parties, which we denote by $\{1, \ldots, n\}$ and $\{-t, \ldots, 0\}$ respectively. We use $[v]$ to denote the degree-t Shamir-sharing of a value v among parties in $\mathcal{P}$.

3.1 Security Definition

We prove the security of our protocols in the standard, standalone simulation-based security model of multiparty computation in the perfect settings [Can00, AL17]. Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let π be an n-party protocol over private and authenticated point-to-point channels and an authenticated broadcast channel. Let $\mathcal{A}$ be the adversary with auxiliary input z, and let $\mathcal{C} \subset \mathcal{P}$ be the set of corrupted parties controlled by it. We define the real and ideal executions:

- **The real execution:** In the real model, the parties run the protocol π where the adversary $\mathcal{A}$ controls the parties in $\mathcal{C}$. The adversary is assumed to be rushing, meaning that in every round it can see the messages sent by the honest parties to the corrupted parties before it determines the message sent by the corrupted parties. The adversary cannot see the messages sent between honest parties on the point-to-point channels. We denote by $\mathsf{Real}^{\pi}_{\mathcal{A}(z),\mathcal{C}}(\vec{x})$ the random variable consisting of the view of the adversary $\mathcal{A}$ in the execution (consisting of all the initial inputs of the corrupted parties, their randomness and all messages they received), together with the output of all honest parties.
- **The ideal execution:** The ideal model consists of all honest parties, a trusted party and an ideal adversary $\mathcal{SIM}$, controlling the same set of corrupted parties $\mathcal{C}$. The honest parties send their inputs to the trusted party. The ideal adversary $\mathcal{SIM}$ receives the auxiliary input z and sees the inputs of the corrupted parties. $\mathcal{SIM}$ can substitute any x_i with any x'_i of its choice (for the corrupted parties) under the condition that $|x'_i| = |x_i|$. Once the trusted party receives (possibly modified) inputs $(x'_1, ..., x'_n)$ from all parties, it computes $(y_1, ..., y_n) = f(x'_1, ..., x'_n)$ and sends y_i to P_i. The output of the ideal execution, denoted as $\mathsf{Ideal}^{f}_{\mathcal{SIM}(z),\mathcal{C}}(\vec{x})$ is the output of all honest parties and the output of the ideal adversary $\mathcal{SIM}$.

Definition 3.1. *We say that a protocol π is t-secure for a functionality f, if for every adversary $\mathcal{A}$ in the real world, there exists an adversary $\mathcal{SIM}$ in the ideal world such that for every $\mathcal{C} \subset \mathcal{P}$ of cardinality at most t, it must hold that*

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

where $\vec{x}$ is chosen from $(\{0,1\}^*)^n$ such that $|x_1| = \ldots = |x_n|$.

3.2 Bivariate Polynomials

A degree (l, m)-bivariate polynomial over $\mathbb{F}$ is of the form $S(x, y) = \sum_{i=0}^{l} \sum_{j=0}^{m} b_{ij} x^i y^j$ where $b_{ij} \in \mathbb{F}$. The polynomials $f_i(x) = S(x, i)$ and $g_i(y) = S(i, y)$ are called i^{th} f and g univariate polynomials of $S(x, y)$ respectively. In our protocol, we use $(2t, t)$-bivariate polynomials where the i^{th} f and g univariate polynomials are associated with party P_i for every $P_i \in \mathcal{P}$.

Claim 3.2 ([AAY21, Claim 3.4]). *Let t be a non-negative integer, $\{1,\ldots,t+1\}$ be distinct elements in $\mathbb{F}$ and $f_1(x),\ldots,f_{t+1}(x)$ be $t+1$ univariate polynomials of degree at most $2t$. Then, there exists a unique $(2t,t)$-bivariate polynomial $S(x,y)$ such that $S(x,i)=f_i(x)$ holds for every $i\in[t+1]$.*

Claim 3.3 ([ACP21, Lemma 2.7]). *Let $C\subseteq D\subseteq\{1,\ldots,n\}$ be two sets such that $|C|\geq t+1$ and $|D|\geq 2t+1$. Let $\{f_i(x)\}_{i\in C}$ be a set of degree-$2t$ polynomials and $\{g_j(y)\}_{j\in D}$ be a set of degree-t polynomials over $\mathbb{F}$. If for every $i\in C$ and every $j\in D$ it holds that $f_i(j)=g_j(i)$, then there exists a unique $(2t,t)$-bivariate polynomial $S(x,y)$ such that $S(x,i)=f_i(x)$ holds for every $i\in C$ and $S(j,y)=g_j(y)$ holds for every $j\in D$.*

Claim 3.4 ([AAY21, Claim 3.6]). *Let $\mathcal{C}\subset\{1,\ldots,n\}$ be a set such that $|\mathcal{C}|\leq t$ and let $p(x)$ and $q(x)$ be two degree-$2t$ polynomials such that $p(i)=q(i)$ holds for every $i\in\mathcal{C}$. Then, the probability distributions $\{(i,S_p(x,i),S_p(i,y))\}_{i\in\mathcal{C}}$ and $\{(i,S_q(x,i),S_q(i,y))\}_{i\in\mathcal{C}}$ are identical, where $S_p(x,y)$ and $S_q(x,y)$ are $(2t,t)$-bivariate polynomials chosen under the constraint that $S_p(x,0)=p(x)$ and $S_q(x,0)=q(x)$ respectively.*

3.3 Finding (n,t)-STAR

Definition 3.5. *Let G be a graph over the nodes $\{1,\ldots,n\}$. We say that a pair (C,D) of sets such that $C\subseteq D\subseteq\{1,\ldots,n\}$ is an (n,t)-*star *in G if the following hold:*

- $|C|\geq n-2t$,
- $|D|\geq n-t$,
- *For every $j\in C$ and every $k\in D$, the edge (j,k) exists in G.*

Canetti [Can93] showed that if a graph has a clique of size $n-t$, then there exists an efficient algorithm which always finds an (n,t)-star. For completeness, we describe the algorithm for finding an (n,t)-star in Algorithm 3.6 which is taken verbatim from [BCG93, Can96]. We describe the (n,t)-star finding algorithm [BCG93, Can96] below.

Algorithm 3.6: STAR – **efficiently finding a star**

Input: an undirected graph G (over the nodes $\{1,\ldots,n\}$), a parameter t.

1. Find a maximum matching M in $\overline{G}$. Let N be the set of matched nodes (namely, the endpoints of the edges in M) and let $\overline{N}:=\{1,\ldots,n\}\setminus N$.
2. Let T be the set of triangle-heads, i.e., all vertices that are not endpoints of the matching but they have two neighbors in the matching.

$$T \;:=\; \left\{i\in\overline{N}\mid\exists j,k \text{ s.t. } (j,k)\in M \text{ and } (i,j),(i,k)\in\overline{G}\right\}\ .$$

   Let $C:=\overline{N}\setminus T$.
3. Let B the set of matched nodes that have neighbors in C. That is, set:

$$B \;:=\; \left\{j\in N\mid\exists i\in C \text{ s.t. } (i,j)\in\overline{G}\right\}\ .$$

   Let $D:=\{1,\ldots,n\}\setminus B$.
4. **Output:** If $|C|\geq n-2t$ and $D\geq n-t$ then output (C,D). Otherwise, output "star not found".

It was shown in [Can96, BCG93] that if a graph has a clique of size $n-t$, then the above procedure halts with a (C, D) star.

Claim 3.7. *Let G be a graph over $\{1, \ldots, n\}$ such that if P_i and P_j are honest then $\{i, j\} \in G$. Then, C contains at least $t+1$ indices of honest parties.*

Proof. Since honest parties trust each other, we have a clique of size at least $2t+1$ in G and thus a (C, D)-star will be found. Since there are always edges between two honest parties in G, all the edges in $\overline{G}$ are either between an honest party and a corrupted party, or between a pair of corrupted parties. Let x be the number of edges in the matching that are between pairs of corrupted parties, and let y be the number of edges in the matching that are between an honest party and a corrupted party. We have that $x+y \leq t$. Next, we claim that the number of honest parties in T (i.e., triangle-heads) is bounded by x. The only triangles in questions are those between an honest party as a head and the two neighbors as corrupted parties that are also in the matching. We claim that each edge in the matching between a pair of corrupted parties can be a part of only one triangle. That is, for $(j, k) \in M$ there exists at most one honest $i \in \overline{N}$ for which $i \in T$. Otherwise, if there exists a pair $i_1, i_2 \in \overline{N}$ such that $i_1, i_2 \in T$, then we can find a larger matching: instead of taking $(j, k) \in M$ we would take (i_1, j) and (i_2, k), in contradiction to the maximality of M.

To conclude, C is defined as $(\{1, \ldots, n\} \setminus N) \setminus T$. In N there are y indices of honest parties, and in T at most x indices. Since $x + y \leq t$, we obtain that C contains at least $t+1$ indices of honest parties. □

4 Packed Verifiable Secret Sharing

Here we present a packed VSS to generate Shamir sharing of $t+1$ secrets at the cost of $\mathcal{O}(n^2 \log n)$ bits point-to-point and broadcast communication.

The Functionality. On holding $t+1$ secrets $s_{-t}, \ldots, s_0$, the dealer chooses a uniformly random $(2t, t)$-bivariate polynomial $S(x, y)$ such that $S(l, 0) = s_l$ for each $l \in \{-t, \ldots, 0\}$ and uses the polynomial as its input. Our functionality for VSS is as follows, followed by the VSS protocol.

Functionality 4.1: $\mathcal{F}_{\mathsf{VSS}}$ – Packed Verifiable Secret Sharing Functionality

Input: The dealer holds a polynomial $S(x, y)$.

1. The dealer sends $S(x, y)$ to the functionality.
2. If $S(x, y)$ is of degree at most $2t$ in x and at most t in y, then the functionality sends to each party P_i the two univariate polynomials $S(x, i), S(i, y)$. Otherwise, the functionality sends $\perp$ to all parties.

Protocol 4.2: Π_{pVSS} – Packed VSS Protocol

Common input: The description of a field $\mathbb{F}$, two sets of distinct elements from it denoted as $\{1, \ldots, n\}$ and $\{-t, \ldots, 0\}$.

Input: The dealer holds a bivariate polynomial $S(x, y)$ of degree at most $2t$ in x and at most t in y. Each P_i initialises a happy bit $\mathsf{happy}_i = 1$ [6].

1. **(Sharing)** The dealer sends $(f_i(x), g_i(y))$ to P_i where $f_i(x) = S(x, i)$, $g_i(y) = S(i, y)$.

2. **(Pairwise Consistency Checks)** Each P_i sends $(f_i(j), g_i(j))$ to every P_j. Let (f_{ji}, g_{ji}) be the values received by P_i from P_j. If $f_{ji} \neq g_i(j)$ or $g_{ji} \neq f_i(j)$, P_i broadcasts $\mathsf{complaint}(i, j, f_i(j), g_i(j))$.

3. **(Conflict Resolution)** For each $\mathsf{complaint}(i, j, u, v)$ such that $u \neq S(j, i)$ or $v \neq S(i, j)$, dealer broadcasts $g_i^D(y) = S(i, y)$. Let pubR be the set of parties for which the dealer broadcasts $g_i^D(y)$. Each $P_i \in \mathsf{pubR}$ sets $\mathsf{happy}_i = 0$. For two mutual complaints ($\mathsf{complaint}(i, j, u, v)$, $\mathsf{complaint}(j, i, u', v')$) with either $u \neq u'$ or $v \neq v'$, if the dealer does not broadcast anything, then discard the dealer.

4. **(Identifying the CORE Set)** Each $P_i \notin \mathsf{pubR}$ broadcasts OK if $f_i(k) = g_k^D(i)$ holds for every $k \in \mathsf{pubR}$. Otherwise, P_i sets $\mathsf{happy}_i = 0$. Let CORE be the set of parties who broadcasted OK. If $|\mathsf{CORE}| < 2t + 1$, then discard the dealer.

5. **(Revealing f-polynomials for non-CORE parties)** For each $P_k \notin \mathsf{CORE}$, the dealer broadcasts $f_k^D(x) = S(x, k)$. Discard the dealer if for any $P_j \in \mathsf{pubR}$ and $P_k \notin \mathsf{CORE}$, $g_j^D(k) \neq f_k^D(j)$. Each $P_i \notin \mathsf{pubR}$ broadcasts OK if $f_k^D(i) = g_i(k)$ holds for every broadcasted $f_k^D(x)$. Otherwise P_i sets $\mathsf{happy}_i = 0$. Let $K = \{P_j | P_j \notin \mathsf{pubR}$ and did not broadcast $\mathsf{OK}\}$.

6. **(Opening g-polynomials for complaining parties)** For each $P_j \in K$, the dealer broadcasts $g_j^D(y) = S(j, y)$. Set $\mathsf{pubR} = \mathsf{pubR} \cup K$. Discard the dealer if $f_k^D(j) \neq g_j^D(k)$ for any $P_k \notin \mathsf{CORE}$ and $P_j \in K$. Each $P_i \in \mathsf{CORE}$ with $\mathsf{happy}_i = 1$ broadcasts OK if $f_i(j) = g_j^D(i)$ for every broadcasted $g_j^D(y)$. Otherwise, P_i sets $\mathsf{happy}_i = 0$. If at least $2t + 1$ parties do not broadcast OK, then discard the dealer.

7. **(Output)** If the dealer is discarded, then each P_i outputs $\perp$. Otherwise, P_i outputs $(f_i(x), g_i(y))$, where $f_i(x) = f_i^D(x)$ if $P_i \notin \mathsf{CORE}$ and $g_i(y) = g_i^D(y)$ if $P_i \in \mathsf{pubR}$.

Theorem 4.3. *Protocol Π_{pVSS} (Protocol 4.2) securely realizes $\mathcal{F}_{\mathsf{VSS}}$ (Functionality 4.1) in the presence of a static malicious adversary controlling up to t parties with $t < n/3$.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We show the existence of a simulator $\mathcal{SIM}$ in the ideal world, such that for any set of corrupted parties $\mathcal{C}$ and for all inputs, the output of all parties in the real protocol with $\mathcal{A}$ is identical to the output in the ideal world with $\mathcal{SIM}$. Depending on whether the dealer is honest or not, we have the following two cases.

Case 1 - The dealer is honest. In this case, the dealer always holds a valid $(2t, t)$-bivariate polynomial $S(x, y)$. The simulator proceeds as follows:

1. $\mathcal{SIM}$ invokes $\mathcal{A}$ on auxiliary input z.

[6] The happy bits will be used later for Multi-Moderated VSS in Section 6.

2. $\mathcal{SIM}$ receives from $\mathcal{F}_{\mathsf{VSS}}$, the polynomials $f_i(x), g_i(y)$ for every $P_i \in \mathcal{C}$ and simulates the protocol execution for $\mathcal{A}$:
   (a) **Sharing:** $\mathcal{SIM}$ sends $(f_i(x), g_i(y))$ to $\mathcal{A}$ for every $P_i \in \mathcal{C}$ on behalf of the dealer.
   (b) **Pairwise Consistency Checks:** $\mathcal{SIM}$ sends $(g_i(j), f_i(j))$ to $\mathcal{A}$ for every $P_i \in \mathcal{C}$ and every honest P_j. $\mathcal{SIM}$ receives from $\mathcal{A}$ the values (f_{ij}, g_{ij}) for each honest party P_j and every $P_i \in \mathcal{C}$. If $f_{ij} \neq f_i(j)$ or $g_{ij} \neq g_i(j)$, $\mathcal{SIM}$ broadcasts $\mathsf{complaint}(j, i, g_i(j), f_i(j))$ on behalf of P_j. $\mathcal{SIM}$ also receives $\mathsf{complaint}(\cdot, \cdot, \cdot, \cdot)$ broadcasted by $\mathcal{A}$.
   (c) **Conflict Resolution:** The dealer never broadcasts $g_j^D(y)$ for honest parties. For every $\mathsf{complaint}(i, j, u, v)$ from $\mathcal{A}$, $\mathcal{SIM}$ checks if $u = f_i(j)$ and $v = g_i(j)$. If not, $\mathcal{SIM}$ broadcasts $g_i^D(y)$ on behalf of the dealer. Define pubR to be the set of parties for which $g_i^D(y)$ was broadcasted.
   (d) CORE **Set Identification:** An honest party never belongs to pubR. Since the dealer is honest, $f_j(i) = g_i^D(j)$ holds for every honest P_j and every $P_i \in \mathsf{pubR}$. The dealer broadcasts OK on behalf of every honest party and receives the OK messages broadcasted by $\mathcal{A}$. Define CORE to be the set of parties who broadcasted OK.
   (e) **Revealing f-polynomials non-CORE parties:** An honest party will always be a part of CORE. On behalf of the dealer, $\mathcal{SIM}$ reveals $f_i^D(x)$ for each $P_i \notin \mathsf{CORE}$. These polynomials will always be consistent with the honest parties' polynomials and the g-polynomials revealed publicly. $\mathcal{SIM}$ broadcasts OK on behalf of each honest party and receives the OK broadcasted by $\mathcal{A}$. Let K be the parties which did not broadcast OK.
   (f) **Opening g-polynomials for parties in K:** Let $\mathsf{pubR} = \mathsf{pubR} \cup K$. On behalf of the dealer, $\mathcal{SIM}$ reveals $g_i^D(y)$ for each $P_i \in K$. These polynomials will always be consistent with the f-polynomials of honest parties and the f-polynomials revealed publicly. $\mathcal{SIM}$ broadcasts OK on behalf of each honest party and receives the OK messages broadcasted by $\mathcal{A}$.
3. **Output:** $\mathcal{SIM}$ outputs whatever $\mathcal{A}$ outputs, and halts.

It can be observed that, since the protocol as well as the simulation is deterministic, the adversary's view in the real execution and ideal execution is identical. Hence, our goal is to now show that the output of honest parties is the same in the real and ideal executions.

In the ideal execution, an honest dealer always invokes the functionality with a valid $(2t, t)$-bivariate polynomial $S(x, y)$. Thus, each honest party P_i outputs the polynomials $f_i(x) = S(x, i)$ and $g_i(y) = S(i, y)$ which it receives from the functionality, and never outputs $\perp$. Moreover, the corrupted parties do not have inputs and hence do not influence the output of the honest parties. We will show that the same holds in the real execution as well.

In the real execution, since the dealer is honest, it always holds a valid $(2t, t)$-bivariate polynomial $S(x, y)$ and sends $f_i(x)$ and $g_i(y)$ as prescribed by the protocol to every P_i. As per the protocol specification, a party's g and f polynomials do not change unless they are revealed publicly by the dealer in Step 3, 5 or 6. However, an honest dealer never reveals an honest party's polynomials during these phases. Hence, during the output phase, an honest party either outputs $f_i(x), g_i(y)$ consistent with $S(x, y)$, or $\perp$. We thus proceed to show that an honest party never outputs $\perp$.

Recall that an honest party outputs $f_i(x)$ and $g_i(y)$ if and only if at least $2t + 1$ parties with $\mathsf{happy} = 1$ broadcast OK during Step 6. Thus, it suffices to show that all the honest parties have their happy bit as 1 and broadcast OK. An honest party P_i has $\mathsf{happy}_i = 1$ and broadcasts OK during Step 6 if and only if the following conditions hold:

1. While resolving complaints, the dealer never broadcasts $g_i^D(y)$.
2. The dealer resolves all pairs of complaints of the type $\mathsf{complaint}(j,k,u,v)$ and $\mathsf{complaint}(k,j,u',v')$ where $u \neq u'$ or $v \neq v'$.
3. All $g_k^D(y)$ broadcasted by the dealer in Step 3 satisfy $f_i(k) = g_k^D(i)$.
4. CORE set includes at least $2t+1$ parties.
5. All $f_j^D(x)$ broadcasted by the dealer for every $P_j \notin \mathsf{CORE}$ in Step 5 and $g_k^D(y)$ broadcasted for every $P_k \in \mathsf{pubR}$ in Step 3 satisfy $f_j^D(k) = g_k^D(j)$, and $f_j^D(i) = g_i(j)$.
6. All $g_k^D(y)$ broadcasted by the dealer for every $P_k \in K$ in Step 6 and all $f_j^D(x)$ broadcasted for every $P_j \notin \mathsf{CORE}$ in Step 5 satisfy $f_j^D(k) = g_k^D(j)$, and $f_i(k) = g_k^D(i)$.

Therefore we conclude that in the real execution, every honest party has $\mathsf{happy}_i = 1$ broadcasts OK in Step 6 and hence every honest party P_i outputs $f_i(x)$ and $g_i(y)$ identical to the ideal execution.

Case 2 - The dealer is corrupt. In this case, the adversary $\mathcal{A}$ controls the dealer. The honest parties do not have any input to the protocol and the protocol is deterministic. The simulator proceeds as follows:

1. $\mathcal{SIM}$ invokes $\mathcal{A}$ on auxiliary input z.
2. $\mathcal{SIM}$ plays the role of all the honest parties while interacting with $\mathcal{A}$, as specified by protocol Π_{pVSS} (Protocol 4.2).
3. Let P_j be an arbitrary honest party emulated by $\mathcal{SIM}$. From its output in the simulated execution, let $\mathcal{G}$ be the set of parties that broadcasted OK in the simulation in Step 6. Then,
   (a) If $|G| \geq 2t+1$, then let $\mathcal{H} \subset \mathcal{G} \setminus \mathcal{C}$ be the set of $t+1$ honest parties which broadcasted OK. $\mathcal{SIM}$ finds the unique $(2t,t)$-bivariate polynomial, say $S(x,y)$ that satisfies $f_i(x) = S(x,i)$ for every $P_i \in \mathcal{H}$. Such a polynomial always exists by virtue of Claim 3.2. $\mathcal{SIM}$ sends $S(x,y)$ to $\mathcal{F}_{\mathsf{VSS}}$ to allow the honest parties to learn their output, and receives the output $f_i(x), g_i(y)$ for each $P_i \in \mathcal{C}$.
   (b) Otherwise, $\mathcal{SIM}$ invokes $\mathcal{F}_{\mathsf{VSS}}$ with an invalid polynomial, say $S(x,y) = x^{2t+1}$ causing all the honest parties to receive $\perp$ in the ideal execution.
4. $\mathcal{SIM}$ outputs whatever $\mathcal{A}$ outputs, and halts.

Since the simulator emulates the honest parties as in the real execution of the protocol, the view of the adversary in the real and ideal world is identical. Thus, it remains to be shown that the output of the honest parties in the ideal world is the same as that in the real execution. For this, we consider the following two cases:

Case I - There exists an honest party that outputs $\perp$ in the real execution. In such a case, we claim that all the honest parties output $\perp$. An honest party outputs $\perp$ only if (i) the dealer does not resolve all mutual complaints (ii) CORE set (decided based on OK messages in Step 4) includes less than $2t+1$ parties, (iii) any of the verification checks on the publicly revealed polynomials fail, or (iii) less than $2t+1$ parties from CORE broadcast OK in Step 6. In all of the above cases, the corresponding messages are broadcasted, and hence all honest parties output $\perp$. Since the real execution and simulated executions are identical, all the simulated honest parties will output $\perp$. In this case, the simulator invokes the functionality with $S(x,y) = x^{2t+1}$, which in turn rejects the polynomial and sends $\perp$ to all the honest parties.

Case II - No honest party outputs $\perp$ in the real execution. In this case, we want to show that each honest party P_i holds $f_i(x) = S(x, i)$ and $g_i(y) = S(i, y)$ consistent with some unique $(2t, t)$-bivariate polynomial $S(x, y)$.

Observe that, if an honest party did not output $\perp$, it implies that at least $2t + 1$ parties from CORE broadcast OK in Step 6. This in turn implies that there exists CORE set with at least $2t + 1$ parties at the conclusion of Step 4, which includes at least $t + 1$ honest parties. By construction of the CORE set, it is guaranteed that the $f_i(x)$ polynomial of every honest $P_i \in \mathsf{CORE}$ is consistent with $g_j(y)$ of every honest P_j. Suppose for the sake of contradiction that there exists some honest $P_i \in \mathsf{CORE}$ and an honest P_j such that $f_i(j) \neq g_j(i)$. We have the following two cases:

1. If $P_j \in \mathsf{pubR}$, the dealer must have broadcasted $g_j^D(y)$ in Step 3. If indeed $f_i(j) \neq g_j^D(i)$, then P_i would not have broadcasted OK in Step 4, which is a contradiction.
2. If $P_j \notin \mathsf{pubR}$, and indeed $f_i(j) \neq g_j(i)$, then honest P_i, P_j would have broadcasted a mutual $\mathsf{complaint}$ which the dealer would have to resolve by broadcasting either $g_i^D(y)$ or $g_j^D(y)$, which is a contradiction.

Therefore, by Claim 3.3, there exists a unique bivariate polynomial $S(x, y)$ such that every honest $P_i \in \mathsf{CORE} \setminus \mathcal{C}$ holds $S(x, i)$ and $S(i, y)$ and every honest P_j holds $S(j, y)$ by the conclusion of Step 4. We claim that all the honest parties output shares on this polynomial at the termination of the protocol. In particular, we prove the following.

Lemma 4.4. *If there exists a set H of at least $t+1$ honest parties such that every $P_h \in H$ holding $(f_h(x), g_h(y))$ has $\mathsf{happy}_h = 1$ and broadcasts OK Step 6, then every honest party P_i outputs g and f polynomial consistent with the unique bivariate polynomial $S(x, y)$ (see Claim 3.2) defined by parties in H.*

Note that, in Step 5, the dealer must have broadcasted $f_j^D(x)$ for every $P_j \notin \mathsf{CORE}$. We now show that the dealer must broadcast $f_j(x) = S(x, j)$. Assume for the sake of contradiction that $f_j(x) \neq S(x, j)$. Since both $f_j(x)$ and $S(x, j)$ are degree-$2t$ polynomials, they can agree on at most $2t$ points. However, the number of honest parties is at least $2t + 1$. Therefore, there must exists some honest party P_l for which $f_j(l) \neq g_l(j) = S(l, j)$. We thus have the following two cases to consider:

1. If $P_l \in \mathsf{pubR}$, then $g_l^D(y)$ was already revealed publicly in Step 3 and hence it must hold that $g_l^D(y) = S(l, y)$. This implies that parties can publicly verify the consistency of $f_j(x)$ and $g_l^D(y)$. If indeed $f_j(l) \neq g_l^D(j)$, every party in H would set its happy to 0 and not broadcast OK. Hence, this case is impossible.
2. If $P_l \notin \mathsf{pubR}$, then $g_l(y)$ is private. This implies that P_l would not broadcast OK during Step 5, and thus the dealer must reveal $g_l^D(y)$ in Step 6. We can have two sub-cases:
   (a) The dealer reveals $g_l^D(y) \neq S(l, y)$. Both, $g_l^D(y)$ and $S(l, y)$ are degree-t polynomials, and hence they can agree on at most t points. Moreover, since $|H| \geq t + 1$, there must exist a party $P_h \in H$ such that $g_l(h) \neq f_h(l)$. This implies that P_h would set $\mathsf{happy}_h = 0$ and not broadcast OK in Step 6, which is a contradiction.
   (b) The dealer reveals $g_l^D(y) = S(l, y)$. This implies that parties can publicly verify the consistency of $f_j(x)$ and $g_l^D(y)$. If indeed $f_j(l) \neq g_l^D(j)$, every party in H would set its happy to 0 and not broadcast OK. Hence, this case is impossible.

We therefore conclude that the dealer must reveal $f_j(x) = S(x, j)$ for every $P_j \notin \mathsf{CORE}$. Thus, if indeed $2t + 1$ parties broadcast OK in Step 6, it holds that each P_i holds $f_i(x) = S(x, i)$ and $g_i(y) = S(i, y)$ on a unique $(2t, t)$-bivariate polynomial $S(x, y)$ (see Claim 3.3).

Since the real and simulated executions are identical, the simulator reconstructs the unique polynomial $S(x, y)$ using the shares of the simulated honest parties in H and invokes the functionality $\mathcal{F}_{\mathsf{VSS}}$ with the valid polynomial $S(x, y)$. The functionality in turn sends to each P_i its shares $f_i(x) = S(x, i)$ and $g_i(y) = S(i, y)$. This is the output of honest parties in the ideal execution. This is exactly the same as output of simulated honest parties which is identical to the output of honest parties in the real execution.

□

Lemma 4.5. *Protocol Π_{pVSS} has a communication complexity of $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels and $\mathcal{O}(n^2 \log n)$ bits broadcast for sharing $\mathcal{O}(n)$ values (i.e., $\mathcal{O}(n \log n)$ bits) simultaneously in 9 rounds.*

In this section, we give details of our gradecast, multi-moderated secret sharing and oblivious leader election protocols. We conclude with the byzantine agreement and the parallel-broadcast using the above as building blocks.

5 Balanced Gradecast

In a Gradecast primitive, a dealer has an input and each party outputs a value and a grade $\{0, 1, 2\}$ such that the following properties are satisfied: **(Validity):** If the dealer is honest then all honest parties output the dealer's input and grade 2; **(Non-equivocation):** if two honest parties each output a grade ≥ 1 then they output the same value; and lastly **(Agreement):** if an honest party outputs grade 2 then all honest parties output the same output and with grade ≥ 1. We model this in terms of a functionality given in Functionality 5.1. The case of an honest dealer captures **validity**. Case 2a and Case 2b capture the **agreement** and **non-equivocation** respectively.

Functionality 5.1: $\mathcal{F}_{\mathsf{Gradecast}}$

The functionality is parameterized by the set of corrupted parties, $I \subseteq \{1, \ldots, n\}$.

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 each $g_j \geq 1$ 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.

In Section 5.1 we first describe a protocol that is not balanced, i.e., the total communication complexity is $\mathcal{O}(n^2 L)$ but in which the dealer sends $\mathcal{O}(n^2 L)$ and every other party sends $\mathcal{O}(nL)$. In Section 5.2 we show how to make the protocol balanced, in which each party (including the dealer) sends/receives $\mathcal{O}(nL)$ bits.

5.1 The Gradecast Protocol

We build our construction in Protocol 5.2 using the idea presented in Section 2.3. Recall that the gradecast used inside our protocol is the naïve gradecast with complexity $\mathcal{O}(n^2 L)$ bits for L-bit message, as in [FM88, FM97]. The security of our protocol is stated in Theorem 5.3.

Protocol 5.2: $\Pi_{\mathsf{Gradecast}}$

Input: The dealer $P \in \{P_1, \ldots, P_n\}$ holds $(t+1)^2$ field elements $(b_{i,j})_{i,j \in \{0,\ldots,t\}}$ where each $b_{i,j} \in \mathbb{F}$ that it wishes to distribute. All other parties have no input.

1. **(Dealer's polynomial distribution) The dealer:**
   (a) The dealer views its elements as a bivariate polynomial of degree at most t in both x and y, i.e., $S(x,y) = \sum_{i=0}^{t} \sum_{j=0}^{t} b_{i,j} x^i y^j$.
   (b) The dealer sends $S(x,y)$ to all parties.
2. **(Pair-wise Information Exchange) Each party P_i:**
   (a) Let $S_i(x,y)$ be the polynomial received from the dealer.
   (b) P_i sends to each party P_j the four polynomials $(S_i(x,j), S_i(j,y), S_i(x,i), S_i(i,y))$.
3. **(Informing dealer about consistency) Each party P_i:**
   (a) Initialize $\mathsf{Agreed}_i = \emptyset$. Let $(f_i^j(x), g_i^j(y), f_j^j(x), g_j^j(y))$ be the polynomials received from party P_j. If $f_i^j(x) = S_i(x,i)$, $g_i^j(y) = S_i(i,y)$, $f_j^j = S_i(x,j)$ and $g_j^j(y) = S_i(j,y)$ then add j to Agreed_i.
   (b) Send Agreed_i to the dealer.
4. **(Quorum forming by dealer) The dealer:**
   (a) Define an undirected graph G as follows: The nodes are $\{1, \ldots, n\}$ and an edge $\{i,j\} \in G$ if and only if $i \in \mathsf{Agreed}_j$ and $j \in \mathsf{Agreed}_i$. Use STAR algorithm (Algorithm 3.6) to find a set $(C,D) \in \{1,\ldots,n\}^2$ where $|C| \geq t+1$ and $|D| \geq 2t+1$, $C \subseteq D$, such that for every $c \in C$ and $d \in D$ it holds that $c \in \mathsf{Agreed}_d$ and $d \in \mathsf{Agreed}_c$.
   (b) Let E be the set of parties that agree with at least $t+1$ parties in C. That is, initialize $E = \emptyset$ and add i to E if $|\mathsf{Agreed}_i \cap C| \geq t+1$.
   (c) Let F be the set of parties that agree with at least $2t+1$ parties in E. That is, initialize $F = \emptyset$ and add i to F if $|\mathsf{Agreed}_i \cap E| \geq 2t+1$.
   (d) If $|C| \geq t+1$ and $|D|, |E|, |F| \geq 2t+1$, then gradecast (C,D,E,F). Otherwise, gradecast $(\emptyset, \emptyset, \emptyset, \emptyset)$.
5. **(First reaffirmation) Each party P_i:**
   (a) Let (C_i, D_i, E_i, F_i, g) be the message that the dealer gradecasted and let g be the associated grade.
   (b) If (1) $g = 2$; (2) $i \in C_i$; (3) $|D_i| \geq 2t+1$; and (4) $\mathsf{Agreed}_i \cap D_i = D_i$; then send OK_C to all parties. Otherwise, send nothing.
6. **(Second reaffirmation) Each party P_i:**
   (a) Let C_i' be the set of parties that sent OK_C in the previous round.
   (b) If $i \in E_i$ and $|\mathsf{Agreed}_i \cap C_i \cap C_i'| \geq t+1$ then send OK_E to all parties.
7. **(Third reaffirmation and propagation) Each party P_i:**
   (a) Let E_i' be the set of parties that sent OK_E in the previous round.

(b) If $i \in F_i$ and $|\mathsf{Agreed}_i \cap E_i \cap E'_i| \geq 2t+1$ then send $(\mathsf{OK}_F, S_i(x,j), S_i(j,y))$ to each party P_j.

8. **(Final propagation) Each party P_i:** Among all messages that were received in the previous round, if there exist polynomials $f'_i(x), g'_i(y)$ that were received at least $t+1$ times, then forward those polynomials to all. Otherwise, forward $\perp$.
9. **(Output) Each party P_i:** Let $((f'_1(x), g'_1(y)), \ldots, (f'_n(x), g'_n(y))$ be the messages received in the previous round. If received at least $2t+1$ polynomials that are not $\perp$, then use robust interpolation to obtain a polynomial $S'(x,y)$. If there is no unique reconstruction or less than $2t+1$ polynomials received, then output $(\perp, 0)$. Otherwise, if $S'(x,y)$ is unique, then:
   (a) If (1) P_i sent OK_F in Round 7; and (2) it received $2t+1$ messages OK_F at the end of Round 7 from parties in F_i with the same polynomials $(f'_i(x), g'_i(y))$; then output $(S', 2)$.
   (b) Otherwise, output $(S', 1)$.

Theorem 5.3. *Let $t < n/3$. Protocol $\Pi_{\mathsf{Gradecast}}$ (Protocol 5.2) securely realizes Functionality $\mathcal{F}_{\mathsf{Gradecast}}$ (Functionality 5.1) in the presence of a malicious adversary controlling at most t parties. The parties send at most $\mathcal{O}(n^3 \log n)$ bits where $\mathcal{O}(n^2 \log n)$ is the number of bits of the dealer's input.*

Proof.
Efficiency. The dealer sends each message to all parties in the first step, i.e., $\mathcal{O}(n^3 \log n)$ bits. Each party then sends a pair of univariate polynomial to each other party, i.e., sends or receive $\mathcal{O}(n \log n)$ for each pair of parties, or a total of $\mathcal{O}(n^3 \log n)$. Each party sends a set of size $\mathcal{O}(n \log n)$ to the dealer, which then gradecasts, using a naïve gradecast, sets of size $\mathcal{O}(n \log n)$. This is again a total of $\mathcal{O}(n^3 \log n)$ bits. Each party then sends a pair of univariate polynomials to each other party, i.e., each party sends or receives $\mathcal{O}(n^2 \log n)$ bits, and a total of $\mathcal{O}(n^3 \log n)$ bits.

Case 1 - The dealer is honest. In this case, the honest dealer sends m to the functionality $\mathcal{F}_{\mathsf{Gradecast}}$ (Functionality 5.1) and all parties receive $(m, 2)$. Moreover, the simulation is trivial. Specifically, the protocol is deterministic and excluding the dealer, parties do not have any input in the protocol. Further, the input of the dealer is known to the simulator, which it receives from the trusted party. Therefore, the simulator can just run the protocol with the exact same inputs as in the real world, and since the protocol is deterministic, the view of the adversary in the real world and the simulated execution would be identical. Thus, all that remains to be shown is that the output of honest parties in the real-world is the same as their output in the ideal world, i.e., all the honest parties output $(m, 2)$ in the protocol execution. This requirement is captured by the following claim and proven subsequently.

Claim 5.4 (Validity)**.** *If the dealer is honest and starts with input polynomial $S(x,y)$, then all honest parties output the same polynomial $S(x,y)$ and grade 2.*

Proof. If the dealer is honest, then it sends the same polynomial $S(x,y)$ to all other parties in Round 1b. This guarantees that all the honest parties are included in the Agreed set of every honest party during Round 3a. Following this, in Round 4, the dealer finds sets (C, D, E, F) as described, and gradecasts them. By the properties of gradecast, it is guaranteed that all the honest parties receive the same sets with grade 2. Moreover, since there exists a clique of size $2t+1$

(consisting of all the honest parties), the STAR algorithm (Algorithm 3.6) finds sets (C, D) as described, such that C contains at least $t+1$ honest parties (see Claim 3.7). This implies that E as well as F contain all honest parties. Then:

1. First reaffirmation (Round 5) – all honest parties in C send OK_C to all parties. This is because, by the definition of the set D, all conditions in Round 5 are satisfied for each honest party P_j for $j \in C$.
2. Second reaffirmation (Round 6) – since C contains at least $t+1$ honest parties, each honest party in E sends OK_E. Moreover, as mentioned above, all honest parties are a part of E and consequently send OK_E.
3. Third reaffirmation ((Round 7)) – as mentioned, all honest parties belong to F and also agree among themselves. Therefore, each honest party P_j sends to every party P_k, the message $(\mathsf{OK}_F, S_j(x,k), S_j(k,y))$.

Since, for every honest P_j, $S_j(x,k)$ is equal to $S(x,k)$ (and $S_j(k,y)$ is equal to $S(k,y)$ respectively) that the dealer sent, it implies that each honest party P_k receives the same polynomial at least $2t+1$ times in Round 8, which it forwards to all other parties. Therefore all honest parties receive at least $2t+1$ pairs of polynomials on $S(x,y)$ thus ensuring robust reconstruction. Moreover, since each honest party P_j had sent OK_F in Round 7, and thus received $2t+1$ OK_F messages at the end of Round 7 with the same polynomials $S(x,j), S(j,y)$, it satisfies the conditions described in step 9a and therefore outputs the polynomial that the dealer had sent, with grade 2. □

Case 2 - The dealer is corrupt. Since the honest parties have no input, simulation is trivial. In particular, the simulator just runs the protocol while simulating the honest parties interacting with the adversary. Since the protocol is deterministic, the view produced by the simulator is exactly the same as the view of the adversary in the real world. At the end of the execution, the simulator holds the outputs of the simulated honest parties, i.e., (m_j, g_j) for every $j \notin I$ where I is the set of corrupted parties. Then:

1. If there exists an honest party P_j with grade $g_j = 2$ then the simulator verifies that every other honest party P_k holds the same message $m_k = m_j$ with grade $g_k \geq 1$. If this condition holds, then it sends $(\mathsf{ExistsGrade2}, m_j, \{g_j\}_{j \notin I})$ to the trusted party. Otherwise, it fails and outputs $\perp$.
2. If all honest parties have $g_j \leq 1$, then the simulator verifies that for every $j, k \notin I$ with $g_j = g_k = 1$ it holds that $m_j = m_k$. If so, it sends $(\mathsf{NoGrade2}, (m_j, g_j)_{j \notin I})$ to the trusted party and halts. Otherwise, it fails and outputs $\perp$.

The functionality then delivers the output to all the honest parties. The following claims show that the output of honest parties in the real world is the same as the output in the ideal world, and that the simulator never fails and outputs $\perp$.

Claim 5.5 (Agreement)**.** *If the dealer is corrupted and some honest party outputs $(S', 2)$, then all honest parties output S' with grade ≥ 1.*

Proof. If some honest party P_i outputs S' with grade 2, then as per the conditions of step 9a, it must have sent OK_F in Round 7. Moreover, the party must have unique interpolation; and received $2t+1$ OK_F messages from parties in F_i with the same pair of polynomials $f_i(x), g_i(y)$ during Round 7.

Since $2t+1$ parties sent OK_F messages with the same $f_i(x), g_i(y)$, it implies that there are at least $t+1$ honest parties in F_i that sent $(\mathsf{OK}_F, f_i(x), g_i(y))$ to P_i. For each such honest party P_j that sent this message, as per the conditions of Round 7, it holds that $j \in F_j$ and that $|\mathsf{Agreed}_j \cap E_j \cap E'_j| \geq 2t+1$. Here, E'_j is the set of parties that sent OK_E to party P_j at the end of Round 6. This further implies that at least $t+1$ honest parties in E'_j sent OK_E, and moreover, the same set of honest parties sent OK_E to all the honest parties. Denote this set of honest parties as E_H.

Since there are at least $t+1$ honest parties in E_H, for each such honest party P_j it holds that $|\mathsf{Agreed}_j \cap C_j \cap C'_j| \geq t+1$, where C'_j is the set of parties that sent OK_C to P_j at the end of Round 5. This in turn implies that at least one honest party belongs to C_j and has sent OK_C, and moreover, the same set of honest parties has sent OK_C to all the other honest parties. Since at least one honest party sent OK_C, due to the conditions of Round 5, it also implies that this honest party received (C, D, E, F) with grade 2 from the dealer. This means that the tuple (C, D, E, F) has been received *identically* by all the honest parties (with grade ≥ 1) and hence we can refer to each of the sets C, D, E and F without party-specific subscript. With the above observations , we now prove the aforementioned claim using the following three claims which are proved subsequently.

Claim 5.6. *All honest parties in C that sent OK_C must hold the same bivariate polynomial, denoted as $\hat{S}(x, y)$.*

Claim 5.7. *If an honest party P_k such that $k \in E$ sent OK_E in Round 6, then there is at least one honest party that sent OK_C in Round 5 and it it holds that $S_k(x, k) = \hat{S}(x, k)$ and $S_k(k, y) = \hat{S}(k, y)$, where $\hat{S}$ is defined by the honest parties in C.*

Claim 5.8. *If an honest party P_j such that $j \in F$ sent OK_F in Round 7, then there are at least $t+1$ honest parties that sent OK_E in Round 6, and at least one honest party that sent OK_C in Round 5. Moreover, it holds that $S_j(x, y) = \hat{S}(x, y)$, where $\hat{S}$ is defined by the honest parties in C.*

Given these three claims, together with the fact that $2t+1$ OK_F messages were received by the honest party that output grade 2 at the end of the protocol, we have that $t+1$ honest parties hold the same polynomial $\hat{S}(x, y)$ in Round 7. Each such honest party sends to each party P_j the two polynomials $\hat{S}(x, j), \hat{S}(j, y)$ in Round 7. Thus, each honest party receives the same two polynomials at least $t+1$ times and forwards them to every other party. Note that no other polynomial can be received with plurality $t+1$. This implies that every honest party receives $2t+1$ pairs of polynomials on the polynomial $\hat{S}$, thus ensuring a robust interpolation. Therefore, every honest party outputs $\hat{S}$ with grade at least 1.

Proof of Claim 5.6: Let C_H be the set of honest parties in C that sent OK_C. Consider $j, k \in C_H$. We show that $S_j(x, y) = S_k(x, y)$, where S_j, S_k are the polynomials received by parties P_j, P_k respectively in Step 1b. As per the conditions of Round 5, an honest party P_j sends OK_C only if it receives (C_j, D_j, E_j, F_j) with grade 2, $j \in C_j$, $|D_j| \geq 2t+1$ and $\mathsf{Agreed}_j \cap D_j = D_j$. Since the message is received with grade 2 by some honest P_j, by the properties of gradecast we know that all honest parties receive the same message (C_j, D_j, E_j, F_j) and therefore we can omit the subscript of D_j. Moreover, since $|D| \geq 2t+1$, it holds that D contains at least $t+1$ honest parties. Further, owing to the fact that $\mathsf{Agreed}_j \cap D = D$ and $\mathsf{Agreed}_k \cap D = D$, we have $S_j(x, d) = S_k(x, d)$ and $S_j(d, y) = S_k(d, y)$ for every honest d in D. Since D has at least $t+1$ honest parties, it holds that $S_j(x, y) = S_k(x, y)$. □

Proof of Claim 5.7: Let E_H denote the set of honest parties that sent OK_E in Round 6. For each honest party P_k in E_H, it holds that $|\mathsf{Agreed}_k \cap C_k \cap C'_k| \geq t+1$ where C'_k is the set of parties that

sent OK_C in Round 5. Let C_H be the set of honest parties that sent OK_C in Round 5. It must hold that $|C_H| \geq 1$, and therefore at least one honest party sent OK_C to all honest parties. Therefore, by the condition of Round 5 and the properties of gradecast, all the honest parties receive the same sets (C, E, D, F) and we thus can omit the subscript of C_k. Moreover, from Claim 5.6, we have that all honest parties in C_H hold the same bivariate polynomial, $\hat{S}(x, y)$.

Further, since $|\mathsf{Agreed}_k \cap C \cap C'_k| \geq t + 1$ for each $k \in E_H$ we have that P_k agrees with at least one honest party in C_H. Let $j \in C_H$ such that $j \in \mathsf{Agreed}_k$. In Step 2b, P_j and P_k exchanged polynomials and since $j \in \mathsf{Agreed}_k$, it must hold that $k \in \mathsf{Agreed}_j$. This implies that P_k holds the two polynomials $\hat{S}(x, k), \hat{S}(k, y)$ in round 6. □

Proof of Claim 5.8: For each party P_j such that $j \in F$ that sends OK_F at the end of Round 7, it holds that $|\mathsf{Agreed}_j \cap E_j \cap E'_j| \geq 2t + 1$ where E'_j is the set of parties that sent OK_E to P_j in Round 6. This implies that at least $t + 1$ honest parties sent OK_E, which further implies that at least one honest party sent OK_C at the end of Round 5 (see claim 5.7). Moreover, as described in the proof of claims 5.6 and 5.7, due to the conditions of Round 5 and properties of gradecast, this also implies that we can omit the subscript of E_j since all honest parties see the same sets (C, D, E, F). Let E_H be the set of honest parties that sent OK_E in Round 6, where we have that $|E_H| \geq t + 1$.

From Claim 5.7 we know that each party P_k in E_H holds $\hat{S}(x, k), \hat{S}(k, y)$. For each honest party P_j such that $j \in F$ that sent OK_F we have that P_j agrees with $t + 1$ parties in E_H. For each such $k \in E_H$, we conclude that $S_j(x, k) = \hat{S}(x, k)$ and $S_j(k, y) = \hat{S}(k, y)$. Since this holds for at least $t + 1$ distinct indices in E_H, we conclude that $S_j(x, y) = \hat{S}(x, y)$. □

This concludes the proof of Claim 5.5. □

Claim 5.9 (Non-equivocation)**.** *If two honest parties each output a grade* ≥ 1 *then they output the same polynomial* $\hat{S}(x, y)$*.*

Proof. If an honest party outputs grade ≥ 1 then it must have received at least $2t + 1$ pairs of polynomials at the end of Round 8. This implies that at least $t + 1$ honest parties forwarded their polynomials in Round 8 thus indicating that each of these honest parties received their polynomials with plurality at least $t + 1$ at the end of Round 7. This in turn implies that there is at least one honest party that sent OK_F in Round 7. By virtue of claim 5.8, we know that all honest parties that sent OK_F in Round 7 hold the same polynomial $\hat{S}(x, y)$. As a result, the only two polynomials that can be forwarded by honest parties in Round 8 (i.e., that can be received $t + 1$ times) must lie on $\hat{S}(x, y)$. Since $t + 1$ honest parties forwarded their polynomials, it holds that all honest parties receive at least $t + 1$ univariate shares on $\hat{S}(x, y)$.

Thus, given that any two degree-t bivariate polynomials can agree on at most t shares, for any honest party that receives $2t + 1$ pairs of polynomials and runs the robust interpolation, the only polynomial that can be accepted is $\hat{S}(x, y)$. □

This concludes the proof of Theorem 5.3. □

5.2 Making the Protocol Balanced

To make the protocol balanced, note that each party sends or receives $\mathcal{O}(n^2 \log n)$ bits except for the dealer who sends $\mathcal{O}(n^3 \log n)$. We therefore change the first round of the protocol as follows:

1. **The dealer:**
   (a) The dealer views its elements as a bivariate polynomial of degree at most t in both x and y, i.e., $S(x,y) = \sum_{i=0}^{t}\sum_{j=0}^{t} b_{i,j}x^i y^j$.
   (b) The dealer sends $S(x,i)$ to each party P_i.
2. **Each party P_i:**
   (a) Forwards the message received from the dealer to every other party.
   (b) Given all univariate polynomials received, say $u(x,1),\ldots,u(x,n)$, runs the Reed-Solomon decoding procedure to obtain the bivariate polynomial $S_i(x,y)$. If there is no unique decoding, then use $S_i(x,y) = \perp$.
3. Continue to run Protocol $\Pi_{\mathsf{Gradecast}}$ (Protocol 5.2) from Step 2 to the end while interpreting $S_i(x,y)$ decoded from the prior round as the polynomial received from the dealer.

Theorem 5.10. *The modified protocol securely realizes Functionality $\mathcal{F}_{\mathsf{Gradecast}}$ (Functionality 5.1) in the presence of a malicious adversary controlling at most t parties. Each party, including the dealer sends or receives $\mathcal{O}(n^2 \log n)$ bits (giving a total communication complexity of $\mathcal{O}(n^3 \log n)$).*

Proof. We show that the above procedure is equivalent to let the dealer send S to each party separately.

The case of an honest dealer. In that case, the dealer holds a (t,t)-bivariate polynomial $S(x,y)$ and sends to each party its share on S, which is essentially a univariate polynomial with degree-t in x. Since there are at least $2t+1$ honest parties, each honest party holds at least $2t+1$ univariate shares (in x) on the polynomial S defined by the dealer and therefore the decoding will result in the polynomial S that the dealer initially holds.

The case of a corrupted dealer. In this case, any attack by the adversary in the modified steps of the protocol can be mapped to a malicious behaviour of the dealer in the polynomial distribution step of the unbalanced protocol. Specifically, for an honest party P_j, the failure to obtain a bivariate polynomial $S_j(x,y)$ upon decoding in the balanced protocol is equivalent to a corrupt dealer sending $\perp$ to P_j in the first step of the unbalanced protocol. Thus, for any adversary $\mathcal{A}$ attacking the modified protocol we can build an adversary $\mathcal{A}'$ for $\Pi_{\mathsf{Gradecast}}$. Since $\Pi_{\mathsf{Gradecast}}$ tolerates at most t parties and securely realizes Functionality $\mathcal{F}_{\mathsf{Gradecast}}$, we get that the modified protocol also securely realizes Functionality $\mathcal{F}_{\mathsf{Gradecast}}$. $\mathcal{A}'$ corrupts the same set of parties as $\mathcal{A}$. It simulates the honest parties in the first two rounds of the modified protocol and obtains the polynomial S_j that each honest party P_j uses. It then simply hands S_j as the polynomial the corrupted dealer sends to P_j in Protocol $\Pi_{\mathsf{Gradecast}}$. □

5.3 Conclusions

The following is a simple corollary, where for general message length of L bits the dealer simply breaks the message into $\ell = \lceil L/(t+1)^2 \log n \rceil$ blocks and runs ℓ parallel executions of gradecast. Each party outputs the concatenation of all executions, with the minimum grade obtained on all executions. The protocol is optimal for $L > n^2 \log n$. We thus obtain the following corollary.

Corollary 5.11. *Let $t < n/3$. There exists a gradecast protocol in the presence of a malicious adversary controlling at most t parties, where for transmitting L bits, the protocol requires the transmission of $\mathcal{O}(nL + n^3 \log n)$ bits, where each party sends or receives $\mathcal{O}(L + n^2 \log n)$ bits.*

6 Multi-Moderated Packed Secret Sharing

At a high level multi-moderated packed secret sharing is a packed VSS moderated by a set $\mathcal{M}$ of $t+1$ distinguished parties called moderators. The parties output a flag for every moderator in the end. We represent the flag for a moderator $M \in \mathcal{M}$ held by a party P_k as v_M^k. In addition, each party P_k holds a variable d_M^k taking values from $\{\mathsf{accept}, \mathsf{reject}\}$ for each $M \in \mathcal{M}$ which identifies whether the dealer is accepted or rejected when M assumes the role of the moderator.

If a moderator M is honest, then every honest party P_k will set $v_M^k = 1$ and the properties of VSS will be satisfied irrespective of whether the dealer is honest or corrupt. If the dealer is honest, every honest P_k will set $\mathsf{d}_M^k = \mathsf{accept}$. For a corrupt dealer, the bit can be 0 or 1 based on the dealer's behaviour, but all the honest parties will unanimously output the same outcome.

If a moderator M is corrupt, then it is guaranteed that: if some honest party P_k sets the flag $v_M^k = 1$, then the properties of VSS will be satisfied irrespective of whether the dealer is honest or corrupt. That is, if the dealer is honest every honest P_k outputs $\mathsf{d}_M^k = \mathsf{accept}$. For a corrupt dealer, it is guaranteed that all the honest parties unanimously output the same outcome for the dealer. We note that when no honest party sets its flag to 1 for a moderator M, then irrespective for whether the dealer is honest or corrupt, it is possible that the parties do not have agreement on their d_M^k.

We capture these properties in a functionality, defined as follows:

Functionality 6.1: $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ – Multi-Moderated Packed Secret Sharing

---

The functionality is parameterized by the set of corrupted parties $I \subseteq \{1, \ldots, n\}$, a set $\mathcal{M}$ of $t+1$ distinguished parties called as moderators.

1. The dealer sends polynomials $f_j(x), g_j(y)$ for every j. If the dealer is honest, then there exists a single $(2t, t)$ polynomial $S(x, y)$ that satisfies $f_j(x) = S(x, j)$ and $g_j(y) = S(j, y)$ for every $j \in \{1, \ldots, n\}$.
2. If the dealer is honest, then send $f_i(x), g_i(y)$ for every $i \in I$ to the adversary.
3. For each moderator $M_j \in \mathcal{M}$:
   (a) If the moderator M_j is honest, then set $v_{M_j}^k = 1$ for every $k \in \{1, \ldots, n\}$. Moreover:
      i. If the dealer is honest, then set $\mathsf{d}_{M_j}^k = \mathsf{accept}$ for every $k \in \{1, \ldots, n\}$.
      ii. If the dealer is corrupt, then receive a message m_j from the adversary. If $m_j = \mathsf{accept}$ then verify that the shares of the honest parties define a unique $(2t, t)$-polynomial. If so, set $\mathsf{d}_{M_j}^k = \mathsf{accept}$ for every $k \in \{1, \ldots, n\}$. In any other case, set $\mathsf{d}_{M_j}^k = \mathsf{reject}$ for every $k \in \{1, \ldots, n\}$.
   (b) If the moderator M_j is corrupt then receive m_j from the adversary.
      i. If $m_j = (\mathsf{Agreement}, (v_{M_j}^k)_{k \notin I}), \mathsf{d}_{M_j})$ where $\mathsf{d}_{M_j} \in \{\mathsf{accept}, \mathsf{reject}\}$, and for some $k \notin I$ it holds that $v_{M_j}^k = 1$. Set $(v_{M_j}^k)_{k \notin I}$ as received from the adversary. Verify that $S(x, y)$ is $(2t, t)$-polynomial. If not, set $\mathsf{d}_{M_j}^k = \mathsf{reject}$ for every $k \notin I$. Otherwise, set $\mathsf{d}_{M_j}^k = \mathsf{d}_{M_j}$ for every $k \notin I$.
      ii. If $m_j = (\mathsf{NoAgreement}, (\mathsf{d}_{M_j}^k)_{k \notin I})$ where each $\mathsf{d}_{M_j}^k \in \{\mathsf{accept}, \mathsf{reject}\}$, then set $v_{M_j}^k = 0$ for every $k \in \{1, \ldots, n\}$ and $\mathsf{d}_{M_j}^1, \ldots, \mathsf{d}_{M_j}^n$ as received from the adversary.
4. **Output:** Each honest party P_k ($k \notin I$) receives as output $f_i(x), g_i(y)$, $(\mathsf{d}_M^k)_{M \in \mathcal{M}}$, and flags $(v_M^k)_{M \in \mathcal{M}}$.

To clarify, each party P_i receives global shares for all moderators, and an output d^i_M and flag v^i_M per each moderator $M \in \mathcal{M}$. If the dealer and the moderator are honest, then all the flags are 1 and the parties accept the shares. If the moderator M_j is corrupted, then as long as there is one honest party P_k with $v^k_{M_j} = 1$ there will be an agreement in the outputs $\mathsf{d}^1_{M_j}, \ldots, \mathsf{d}^n_{M_j}$ (either all the honest parties accept or all of them reject). When $v^k_{M_j} = 0$ for all the honest parties, we might have inconsistency in the outputs $\mathsf{d}^1_{M_j}, \ldots, \mathsf{d}^n_{M_j}$ with respect to that moderator.

The protocol. We build on the discussion given in Section 2.1. We consider the protocol of VSS where the dealer inputs some bivariate polynomial $S(x, y)$ of degree at most $2t$ in x and degree at most t in y. For multi-moderated packed secret sharing, essentially, each broadcast from Π_{pVSS} is simulated with two sequential gradecasts. The first gradecast is performed by the party which intends to broadcast in the underlying packed VSS protocol, while the second is executed by a moderator. Note that these gradecasts are realized via the protocol $\Pi_{\mathsf{Gradecast}}$, presented in the Section 5, having the optimal communication complexity. Up to Step 6 of Π_{pVSS} (Protocol 4.2), the dealer is the moderator for each gradecast. At Step 6, we fork into $t + 1$ executions, with a unique party acting as the moderator in each execution. Since the protocol steps remain similar to Π_{pVSS}, we describe the multi-moderated packed secret sharing protocol below in terms of how the broadcast is simulated at each step and the required changes at Step 6 of the packed VSS protocol.

Protocol 6.2: $\Pi_{\mathsf{mm\text{-}pVSS}}$ **– Multi-Moderated Packed Secret Sharing**

Simulating broadcast up to (including) Step 6 of Π_{pVSS}:

1. Simulating broadcast of a message by the dealer.
   (a) **The dealer:** When the dealer has to broadcast a message m it gradecasts it.
   (b) **Party P_i:** Let (m, g) be the message gradecasted by the dealer, where m is the message and g is the grade. Proceed with m as the message broadcasted by the dealer. If $g \neq 2$, then set $\mathsf{happy}_i = 0$ within the execution of Π_{pVSS}.
2. Simulating broadcast of a party P_j.
   (a) **Party P_j:** When P_j wishes to broadcast a message m, it first gradecasts it.
   (b) **The dealer:** Let (m, g) be the message and g its associated grade. The dealer gradecasts m.
   (c) **Each party P_i:** Let (m', g') be the messages 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 $\mathsf{happy}_i = 0$ within the execution of Π_{pVSS}.

After Step 6 of Π_{pVSS}:

1. **Each party P_i:** Set $v^i_{M_j} = 1$, and let $f_i(x), g_i(y)$ be the pair of shares P_i is holding at end of Step 6. Gradecast accept if $\mathsf{happy}_i = 1$ and reject otherwise.
   At this point, we fork into $|\mathcal{M}|$ executions, one per moderator $M_j \in \mathcal{M}$ as follows:
   (a) **The moderator M_j:** Let $(a_1, \ldots, a_n)$ be the decisions of all parties as received from the gradecast. Gradecast $(a_1, \ldots, a_n)$.
   (b) **Each party P_i:** Let $(a_1, \ldots, a_n)$ be the decisions received directly from the parties, and let $(a'_1, \ldots, a'_n)$ be the message gradecasted from the moderator M_j with associated grade g'. Set $v^i_{M_j} = 0$ if $g' \neq 2$, or there exists a_k received from P_k with grade 2 but for which $a_k \neq a'_k$. Then:

i. If there exists $2t+1$ accepts within $(a'_1, \ldots, a'_n)$, then set $\mathsf{d}^i_{M_j} = \mathsf{accept}$.

ii. Otherwise, set $\mathsf{d}^i_{M_j} = \mathsf{reject}$.

2. **Output:** P_i outputs $(f_i(x), g_i(y)), (\mathsf{d}^i_{M_1}, \ldots, \mathsf{d}^i_{M_t})$ and $(v^i_{M_1}, \ldots, v^i_{M_t})$.

Theorem 6.3. *Let $t < n/3$. Protocol $\Pi_{\mathsf{mm\text{-}pVSS}}$ (Protocol 6.2) computes $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ (Functionality 6.1) in the presence of a malicious adversary corrupting at most t parties. The protocol requires the transmission of $\mathcal{O}(n^2 \log n)$ bits over point-to-point channels, the dealer gradecasts $\mathcal{O}(n^2 \log n)$ bits, and each party gradecasts at most $n \log n$ bits.*

Proof. **Case 1 - The dealer is honest.** Let $\mathcal{S}_{VSS}$ be the simulator of the VSS protocol (Protocol 4.2). The simulator receives from the trusted party, the shares of the corrupted parties – $S(x,i), S(i,y)$ as in Step 2 of the functionality, and simulates the view of the adversary similar to $\mathcal{S}_{VSS}$. While all the modifications described in the protocol for $\Pi_{\mathsf{mm\text{-}pVSS}}$ (Protocol 6.2) are deterministic, the simulator now also has to emulate Functionality 5.1. We claim that when the dealer is honest, no honest party sets $\mathsf{happy}_i = 0$. This holds from the same reasoning as in $\mathcal{S}_{VSS}$, and due to the fact that the modifications in the multi-moderated packed secret sharing do not affect this claim. In particular, note that the dealer's broadcasted messages, which are now emulated using gradecast will always be received with grade 2 by all the honest parties. Furthermore, observe that every other party's broadcast which is emulated via that party's gradecast followed by dealer's gradecast will not lead to an honest party setting $\mathsf{happy}_i = 0$. This is because neither an honest dealer's gradecast will have grade less than 2, nor a gradecast by some party which is received by an honest party with grade 2 will follow the dealer's gradecast with a different output value. The latter holds due to the property of gradecast, where if an honest party receives a message with grade 2 then all the honest parties (including the dealer) would receive the same message with grade ≥ 1, and the dealer would further gradecast the same message. As a result, no honest party sets $\mathsf{happy}_i = 0$.

Consequently, at the modified Step 6, an honest party's initial decision will be accept, and the final decision of whether to accept or reject, and the grade of the moderator is determined by the messages it receives from the moderator and the decisions it received from other parties. Thus, the simulator can completely simulate this stage as well for all moderators $M_j \in \mathcal{M}$. It can then obtain the outputs of all honest parties, i.e., the set $(\mathsf{d}^k_{M_j}, v^k_{M_j})_{k \notin I, M_j \in \mathcal{M}}$, although it cannot necessarily compute the output share of each honest party. For each moderator $M_j \in \mathcal{M}$:

1. If the moderator is honest, then the trusted party does not expect a message from the simulator and just delivers to each honest party $\mathsf{d}^k_{M_j} = \mathsf{accept}$ and $v^k_{M_j} = 1$. We will show below (Claim 6.4) that the same happens in the simulated execution, and in the real execution.
2. If there exists a simulated honest party that outputs $v^k_{M_j} = 1$, then we prove below (Claim 6.5) that for every honest P'_k we have that $\mathsf{d}^{k'}_{M_j} = \mathsf{d}^k_{M_j}$. Then, the simulator sends $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k \notin I})$ to the trusted party. Since the dealer is honest, $S(x,y)$ is of degree $(2t,t)$, and the trusted party would just forward $\mathsf{d}^k_{M_j}, v^k_{M_j}$ to every honest party P_k.
3. If for every $k \notin I$ it holds that $v^k_{M_j} = 0$ then the simulator sends $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k \notin I})$ to the trusted party. The trusted party just forwards $\mathsf{d}^k_{M_j}$ to every honest party P_k with grade 0 (as in the simulated execution).

The above shows that the output of all honest parties in the ideal world is exactly the same as the output of honest parties simulated by $\mathcal{S}_{VSS}$ in the simulated execution. As mentioned in the proof of Π_{pVSS} (Theorem 4.3), the view of the adversary in the simulation of $\mathcal{S}_{VSS}$ and in the real execution is exactly the same. We now show that the output of all honest parties in the real execution is exactly the same as the output of all simulated honest parties in the simulated execution. To conclude the case of an honest dealer, we have the following claims:

Claim 6.4. *When the dealer is honest and the moderator M_j is honest, every honest party P_k sets $\mathsf{d}^k_{M_j} = \mathsf{accept}$ and $v^k_{M_j} = 1$.*

Proof. Note that at Step 6, if some honest party receives the decision a_i with grade 2 from a party P_i, then by the properties of gradecast, the honest moderator M_j receives the same a_i (with grade ≥ 1) and further gradecasts the exact same message. Moreover, the message $(a_1, \ldots, a_n)$ gradecasted by an honest moderator is received by all the honest parties with grade 2. Thus, for every honest party P_k it holds that $v^k_{M_j} = 1$. Additionally, as described, for the case of an honest dealer, for every honest party P_i it holds that $\mathsf{happy}_i = 1$ and hence every honest party gradecasts its initial decision as accept, ensuring a total of at least $2t+1$ accept decisions. Thus, for every honest party P_k, it holds that $\mathsf{d}^k_{M_j} = \mathsf{accept}$. Note that in the ideal world, this corresponds to Step 3(b)i of the functionality (Functionality 6.1), where similar to the real world, all honest parties output $\mathsf{d}^k_{M_j} = \mathsf{accept}$ and $v^k_{M_j} = 1$.

□

Claim 6.5. *If the dealer is honest and the moderator M_j is corrupted then if some honest party P_k holds $v^k_{M_j} = 1$ then all other honest parties P'_k have $\mathsf{d}^{k'}_{M_j} = \mathsf{accept}$.*

Proof. If an honest party P_k outputs $v^k_{M_j} = 1$, then it must have received the decisions $(a'_1, \ldots, a'_n)$ from M_j with grade 2, and moreover, all the decisions corresponding to honest parties must agree with those gradecasted previously by the respective honest parties themselves. As described earlier, for the case of an honest dealer, every honest party gradecasts its initial decision as accept. This implies that, of $(a'_1, \ldots, a'_n)$, at least $2t+1$ decisions are accept. Furthermore, as guaranteed by gradecast, all the other honest parties see the exact same decisions $(a'_1, \ldots, a'_n)$ for the moderator M_j, though they might have grade equal to 1. As a result, every other honest party P'_k sets $\mathsf{d}^{k'}_{M_j} = \mathsf{accept}$ while $v^{k'}_{M_j}$ is either 0 or 1 (depending on the grade of the message $(a'_1, \ldots, a'_n)$ gradecasted by M_j). In the ideal execution, the simulator in this case sends $(\mathsf{Agreement}, \cdot)$ as described above and the outputs $\mathsf{d}^k_{M_j}$ are the same for all honest P_k. □

Case 2 - The dealer is corrupt. When the dealer is corrupted, simulation is trivial since all honest parties have no input. The simulator just simulates the honest parties interacting with the adversary as in the protocol specifications, and simulates the behavior of the gradecast functionality as in Functionality 5.1. Moreover, since the protocol is deterministic, the view of the adversary in the real execution and in the ideal execution is exactly the same.

The simulator proceeds by simulating all the honest parties' messages and let $(f_i(x), g_i(y), (\mathsf{d}^i_{M_1}, \ldots, \mathsf{d}^i_{M_t}), (v^i_{M_1}, \ldots, v^i_{M_t})$ be the output of the simulated honest party P_i. The decision gradecasted by an honest party is the same for all the moderators. The proof of packed VSS (Theorem 4.3) shows that if there are at least $t+1$ honest parties that decide on accept in Step 6, then there exists

a unique bivariate polynomial $S(x, y)$ of degree-$(2t, t)$ such that the shares of all the honest parties lie on this polynomial. Then, for each moderator M_j:

1. **If M_j is honest:** We will show below (Claim 6.6) that the output of every simulated honest party P_k is $v^k_{M_j} = 1$ and that all the honest parties have the same output $\mathsf{d}^k_{M_j}$. If all of them have $\mathsf{d}^k_{M_j} = \mathsf{accept}$, then the simulator sets $m_j = \mathsf{accept}$. If all have $\mathsf{d}^k_{M_j} = \mathsf{reject}$, then it sets $m_j = \mathsf{reject}$.
2. **If M_j is corrupted:** We have the following sub-cases to consider.
   (a) **There exists an honest party P_k with $v^k_{M_j} = 1$ and $\mathsf{d}^k_{M_j} = \mathsf{accept}$:** We show below (Claim 6.7) that every other honest party P'_k must have $\mathsf{d}^{k'}_{M_j} = \mathsf{accept}$, and the shares of all the honest parties lie on a unique bivariate polynomial $S(x, y)$ of degree-$(2t, t)$. In this case, the simulator sets $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k \notin I}, \mathsf{accept})$.
   (b) **There exists an honest party P_k with $v^k_{M_j} = 1$ and $\mathsf{d}^k_{M_j} = \mathsf{reject}$:** We show subsequently (Claim 6.7) that every other honest party P'_k must have $\mathsf{d}^{k'}_{M_j} = \mathsf{reject}$. Thus the simulator sets $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k \notin I}, \mathsf{reject})$.
   (c) **All honest parties have $v^k_{M_j} = 0$:** In this case, there is no guarantee on the output. Therefore, the simulator sets $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k \notin I})$.

The simulator sends m_j for every $M_j \in \mathcal{M}$ to the trusted party. It is easy to see by inspection of the functionality (Functionality 6.1) that upon sending these messages to the trusted party, all the honest parties in the ideal world receive the exact same output as the simulated honest parties in the simulated execution. Further, since the simulated execution is exactly the same as the corresponding execution in the real world, the outputs of honest parties and views are identical. To conclude the proof, we prove the following claims.

Claim 6.6. *For an honest moderator M_j, each honest party P_k outputs a grade $v^k_{M_j} = 1$. Moreover, all the honest parties have the same output $\mathsf{d}^k_{M_j}$, i.e., either all* accept *or all* reject.

Proof. Note that all the honest parties receive the messages gradecasted by the honest moderator M_j identically and with grade 2. Moreover, for every message gradecasted by some party which is received with grade 2 by any honest party, the properties of gradecast ensure that the moderator receives the same message (although with grade ≥ 1). This implies that the moderator gradecasts exactly the same message. In particular, this also holds true for the gradecast of the decisions $(a'_1, \ldots, a'_n)$ where each a_i was previously gradecasted by party P_i. Hence, for every honest P_k it must hold that $v^k_{M_j} = 1$. Moreover, by the property of gradecast, since all the honest parties receive the same $(a'_1, \ldots, a'_n)$, if the decisions contain at least $2t + 1$ accept messages, then every honest party P_k sets $\mathsf{d}^k_{M_j} = \mathsf{accept}$. On the other hand, if the decisions do not contain $2t+1$ accept messages, then all honest parties set $\mathsf{d}^k_{M_j} = \mathsf{reject}$.

For the former case, we have that at least $t+1$ honest parties decided on accept. Therefore, according to Lemma 4.4, this implies that all honest parties have shares on the same $(2t, t)$ polynomial $S(x, y)$.

□

Claim 6.7. *For a corrupted moderator M_j, if some honest party P_k holds $v^k_{M_j} = 1$ then every other honest party P'_k holds $\mathsf{d}^{k'}_{M_j} = \mathsf{d}^k_{M_j}$.*

Proof. If an honest party P_k outputs $v^k_{M_j} = 1$ with $\mathsf{output}^k_{M_j} = \mathsf{accept}$, then it must have received $2t+1$ decisions $(a'_1, \ldots, a'_n)$ of accept from the moderator with grade 2 such that the decisions corresponding to the honest parties actually agree with the ones gradecasted by the honest parties themselves. Due to the property of gradecast, this implies that all honest parties receive the same decisions $(a'_1, \ldots, a'_n)$. Therefore, all other honest parties P'_k also set $\mathsf{d}^{k'}_{M_j} = \mathsf{accept}$. We now show that there is a unique polynomial $S(x, y)$ of degree at most $2t$ in x and degree t in y, and all the honest parties output shares on this polynomial.

Since the decisions of honest parties are received with grade 2, and as described these agree with the decisions of the corresponding honest parties in $(a'_1, \ldots, a'_n)$, there must be at least $t+1$ honest parties that decided accept. For each such honest party P_i it holds that $\mathsf{happy}_i = 1$. This implies that all the broadcasts within the VSS protocol were simulated by the dealer (as a moderator) correctly: P_i received all the gradecasts made by the dealer with grade 2, each complaint in the protocol that an honest party made was received with grade 2 and was echoed by the dealer with grade 2. Since there are $t+1$ honest parties with $\mathsf{happy}_i = 1$, we have that all the complaints of honest parties were publicly resolved by the dealer, and received by all other honest parties with grade at least 1. Thus, all honest parties see the same messages, and all parties that are not happy have public shares. By the properties of the underlying VSS protocol, all shares of honest parties lie on the same bivariate polynomial $S(x, y)$.

If an honest party P_k outputs $v^k_{M_j} = 1$ with $\mathsf{d}^k_{M_j} = \mathsf{reject}$, then every honest party P'_k sets $\mathsf{d}^{k'}_{M_j} = \mathsf{reject}$. The proof follows similar to the previous case. Specifically, if P_k outputs $v^k_{M_j} = 1$ with $\mathsf{d}^k_{M_j} = \mathsf{reject}$, this implies that P_k must have received $(a'_1, \ldots, a'_n)$ gradecasted by M_j with grade 2 such that the decisions of honest parties in $(a'_1, \ldots, a'_n)$ agree with the ones gradecasted by the respective parties themselves. Thus, all honest parties receive the same $(a'_1, \ldots, a'_n)$. Moreover, $(a'_1, \ldots, a'_n)$ does not contain $2t+1$ accept messages. Thus, each honest party P'_k sets $\mathsf{d}^{k'}_{M_j} = \mathsf{reject}$. □

Efficiency. As in the packed VSS protocol, the sharing phase requires $\mathcal{O}(n^2 \log n)$ bits of point-to-point communication. Every party gradecasts $\mathcal{O}(n)$ values in the worst case to communicate its complaints, while the dealer performs $\mathcal{O}(n^2)$ gradecasts to resolve the complaints. □

6.1 Reconstruction

The reconstruction protocol ensures that even for a corrupt moderator, all the honest parties reconstruct the same value when its flag is set to 1 by some honest party. This aligns with the guarantees of the sharing phase, which ensures that the protocol achieves VSS corresponding to a moderator when there exists an honest party with its flag set to 1 at the end of the sharing phase.

Protocol 6.8: $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}pVSS}}$ **– Reconstruct of Multi-Moderated Packed Secret Sharing**

The protocol is parameterized by the set of moderators $\mathcal{M}$ and a set B containing $|\mathcal{M}|$ distinct non-zero values in the field. To be specific B denotes the set $\{-t, \ldots, 0\}$ used in Π_{pVSS}. We assume a one-to-one mapping between $\mathcal{M}$ and $\{-t, \ldots, 0\}$.

Input: Each party P_i holds $(f_i(x), g_i(y)), (\mathsf{d}^i_M)_{M \in \mathcal{M}}$ and $(v^i_M)_{M \in \mathcal{M}}$.

1. Each party sends $f_i(x)$ to all. Let $(f_1(x)', \ldots, f_n(x)')$ be the polynomials received.
2. For each $M \in \mathcal{M}$ (let $\beta^* \in B$ be its associated value):

(a) If $\mathsf{d}_M^i = \mathsf{accept}$, then use Reed Solomon decoding procedure to reconstruct the unique degree-t polynomial $g_{\beta^*}(y)$ that agrees with at least $2t+1$ values $f_1(\beta^*), \ldots, f_n(\beta^*)$ and set $s_M^i = g_{\beta^*}(0)$. If there is not unique decoding, then set $s_M^i = 0$.

(b) If $\mathsf{d}_M^i = \mathsf{reject}$, then set $s_M^i = 0$.

3. **Output:** Output $(s_M^i)_{M \in \mathcal{M}}$.

Theorem 6.9. *For each moderator $M \in \mathcal{M}$, if there exists an honest party with $v_M^k = 1$ then all honest parties hold the same $s_M^{k'} = s_M^k$.*

Proof. By the properties of $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ (Functionality 6.1), if there exists an honest party P_k that holds $v_M^k = 1$ for some moderator M then all honest parties hold the same value d_M, i.e., all accept or reject. For each $\mathsf{d}_M = \mathsf{reject}$, every honest party P_i sets $s_M^i = 0$ in the reconstruction protocol. On the other hand, if there exists an honest party with $\mathsf{d}_M = \mathsf{accept}$, then according to Functionality 6.1, even in the case of a corrupted dealer we have that all the shares of honest parties lie on the same $(2t, t)$-bivariate polynomial $S(x, y)$. As a result, each polynomial $g_{\beta^*}(y) = S(\beta^*, y)$ is of degree-t. Moreover, the parties send their shares to one another, and thus the $2t+1$ honest parties send the degree-$2t$ polynomials $S(x, i)$ to one another. Therefore, each honest party would have at least $2t+1$ correct points on $S(\beta^*, y)$, for which the Reed-Solomon error correction will return a unique decoding, thus ensuring that every honest party P_i obtains the same output $s_M^i = S(\beta^*, 0)$. □

7 Oblivious Leader Election

We start with the functionality which captures OLE with fairness δ, where each party P_i outputs a value $\ell_i \in \{1, \ldots, n\}$ such that with probability at least δ there exists a value $\ell \in \{1, \ldots, n\}$ for which the following conditions hold: (a) each honest P_i outputs $\ell_i = \ell$, and (b) P_ℓ is an honest party. The functionality is parameterized by the set of corrupted parties I, a parameter $\delta > 0$ and a family of efficiently sampling distributions $\mathcal{D} = \{D\}$. Each $D \in \mathcal{D}$ is a distribution $D : \{0,1\}^{\mathsf{poly}(n)} \to \{1, \ldots, n\}^n$ satisfying: $\Pr_{r \leftarrow \{0,1\}^{\mathsf{poly}(n)}}[D(r) = (j, \ldots, j) \text{ s.t. } j \notin I] \geq \delta$.

Functionality 7.1: $\mathcal{F}_{\mathsf{OLE}}$ – Oblivious Leader Election Functionality

---

The functionality is parameterized by the set of corrupted parties $I \subset \{1, \ldots, n\}$ and the family $\mathcal{D}$.

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. It hands r to the adversary and it hands ℓ_i to every party P_i .

Looking ahead, our protocol will define a family $\mathcal{D}$ in which the functionality can efficiently determine whether a given sampler D is a member of $\mathcal{D}$. Specifically, we define the sampler as a parametrized algorithm with some specific values hardwired. Therefore, the ideal adversary can just send those parameters to the functionality to specify D in the family.

Protocol 7.2: Π_{OLE} – Oblivious Leader Election Protocol

---

1. **Choose and commit weights:** Each party $P_i \in \mathcal{P}$ acts as the dealer and chooses $c_{i\to j}$ as random values in $\{1,\ldots,n^4\}$, for every $j \in \{1,\ldots,n\}$. P_i then runs the following for $T := \lceil n/t+1 \rceil$ times in parallel. That is, for $\ell \in [1,\ldots,T]$, each P_i acting as the dealer executes the following in parallel:
   (a) Let the set of moderators be $\mathcal{M}_\ell = (P_{(\ell-1)\cdot(t+1)+1},\ldots,P_{\ell\cdot(t+1)})$.
   (b) The dealer P_i chooses a random $(2t,t)$-bivariate polynomial $S^{i,\ell}(x,y)$ while hiding the $t+1$ values $c_{i\to j}$ for every $j \in \{(\ell-1)\cdot(t+1)+1,\ldots,\ell\cdot(t+1)\}$, one corresponding to each moderator $P_j \in \mathcal{M}_\ell$. Specifically, P_i chooses $S^{i,\ell}(x,y)$ such that $S^{i,\ell}(0,0) = c_{i\to(\ell-1)\cdot(t+1)+1}$ and so on till $S^{i,\ell}(-t,0) = c_{i\to\ell\cdot(t+1)}$. The parties invoke $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ (Fig. 6.1) where P_i is the dealer, and the moderators are parties in $\mathcal{M}_\ell$.
   (c) Each party P_k gets as output a pair of shares $f_k^{i,\ell}(x), g_k^{i,\ell}(y)$, outputs $\mathsf{d}_{i,j}^k$ and a flag $v_{i,j}^k$ for each moderator $P_j \in \mathcal{M}_\ell$.

   Note that the above is run for all dealers $P_1,\ldots,P_n$ in parallel, where each dealer has T parallel instances (in total $T\cdot n$ invocations).
   Upon completion of the above, let $\mathsf{succeeded}_i$ be the set of moderators for which P_i holds a flag 1 in all executions, i.e., $\mathsf{succeeded}_i := \{j \mid v_{d,j}^i = 1 \text{ for all dealers } P_d \in \mathcal{P}\}$.
2. **Reconstruct the weights and pick a leader:** The reconstruction phase, $\Pi_{\mathsf{mm\text{-}pVSS}}^{\mathsf{Rec}}$ (Fig. 6.8) of each of the above nT instances of multi-moderated packed secret sharing is run in parallel to reconstruct the secrets previously shared.
   Let $c_{i\to j}^k$ denote P_k's view of the value $c_{i\to j}$ for every $i,j \in \{1,\ldots,n\}$, i.e., the reconstructed value for the instance where P_i is the dealer and P_j is the moderator.
   Each party P_k sets $c_j^k = \sum_{i=1}^n c_{i\to j}^k \bmod n^4$ and outputs j that minimizes c_j^k among all $j \in \mathsf{succeeded}_k$ (break ties arbitrarily).

Theorem 7.3. *Let $t < n/3$. Protocol Π_{OLE} (Protocol 7.2) computes $\mathcal{F}_{\mathsf{OLE}}$ (Functionality 7.1) in the presence of a malicious adversary corrupting at most t parties. The protocol requires a transmission of $\mathcal{O}(n^4 \log n)$ bits over point-to-point channels.*

Proof. The simulator first simulates $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ (Functionality 6.1) for all the packed secret sharings of all the honest dealers. For this, it generates random shares $f_k^{i,\ell}(x), g_k^{i,\ell}(y)$ for each $\ell \in \{1,\ldots,T\}$ on behalf of every honest dealer P_i for the parties corrupted by the adversary. Observe that these shares do not determine the underlying values $c_{i\to j}$ (see Claim 3.4) for each $j \in \{1,\ldots,n\}$. Note that each party acts as the moderator in one instance of multi-moderated secret sharing where P_i is the dealer. Hence, for every honest moderator P_j, it sets $\mathsf{d}_{i,j}^k = \mathsf{accept}$ and flag $v_{i,j}^k = 1$ for every P_k. For each corrupted moderator P_j, the simulator receives from the adversary, a message m_j as per Functionality 6.1, which determines the output $\mathsf{d}_{i,j}^k$ and flag $v_{i,j}^k$ for all honest parties P_k and whether the reconstruction would be a default value (i.e., 0 when $\mathsf{d}_{i,j}^k = \mathsf{reject}$) or the secret of the dealer otherwise. Following this, the simulator simulates Functionality 6.1 for all the packed secret sharings of all corrupted dealers. First, for every corrupt P_i, it receives from the adversary the shares $f_k^{i,\ell}(x), g_k^{i,\ell}(y)$ for each honest P_k and every $l \in \{1,\ldots,T\}$. For each honest moderator P_j, it sets the flag $v_{i,j}^k = 1$ for every P_k. Further, it receives from the adversary a message m_j as per Functionality 6.1. Depending on m_j and whether the shares of the honest parties received from the adversary define a unique $(2t,t)$ polynomial, the simulator sets $\mathsf{d}_{i,j}^k$ to accept or reject as per the functionality. Further, as in the case of honest dealers, it receives from the adversary, for each

corrupted moderator, a message as per Functionality 6.1, which similarly determines the outputs of all honest parties and whether the reconstruction would be a default value (i.e., 0) or the secret of the dealer. Thus, given the shares of the honest parties, $\mathsf{d}^k_{i,j}$ and flag $v^k_{i,j}$, the simulator can fully compute the view of each party P_k of the value – $c^k_{i\to j}$ for every corrupted P_i, every moderator P_j. Moreover, given the flags $v^k_{i,j}$, the simulator can compute $\mathsf{succeed}_k$ set for each every honest P_k.

Given all the above information, the simulator can set the sampling algorithm D as follows (this also defines the family $\mathcal{D}$). The sampling algorithm D is parameterized with the values $c^k_{i\to j}$ for every corrupted P_i and every moderator P_j, and the flags $v^k_{i,j}$ for every P_i, P_j. Further, the sampling algorithm uses its randomness r to pick all the secret $c_{i\to j}$ values for every honest dealer P_i and every moderator P_j. Then, given the values $c_{i\to j}$ and the corresponding $v^k_{i,j}$ for each P_k, the algorithm can simulate the output $c^k_{i\to j}$ for each party P_k, i.e., P_k's view of the value $c_{i\to j}$ for every honest dealer P_i and moderator P_j. Consequently, it can compute $c^k_j = \sum_{i=1}^n c^k_{i\to j} \bmod n^4$ and set ℓ_k as the index $j \in \{1,\ldots,n\}$ that minimizes c^k_j among all $j \in \mathsf{succeed}_k$, just as the protocol.

Note that the family $\mathcal{D}$ is defined as described in the algorithm above, and therefore to specify D it is enough for the simulator to just send the parameterized values $c^k_{i\to j}$ for every corrupted P_i, and the flags $v^k_{i,j}$. Therefore the functionality itself is also efficient.

Upon defining the sampling algorithm as above, the simulator sends D to the functionality. The functionality returns the randomness used for sampling, which is essentially all the values $c_{i\to j}$ corresponding to every honest dealer P_i and every moderator P_j. Using these values, the simulator can then generate the polynomials $S^{i,\ell}(x,y)$ for each $\ell \in \{1,\ldots,T\}$ that each honest dealer uses in its T instance of the multi-moderated secret sharing, as a function of the shares it has sent to the adversary so far. That is, the simulator sets $S^{i,\ell}(x,y)$ as described in Step 1b under the constraint that $S^{i,\ell}(x,k) = f^{i,\ell}_k(x)$ and $S^{i,\ell}(k,y) = g^{i,\ell}_k(y)$ sent to the adversary for each corrupt P_k. The simulator then uses these polynomials to simulate the reconstruction phase of each instance of multi-moderated secret sharing. For this, the simulator sends $f^{i,\ell}_k(x)$ for each honest P_k to the adversary, where for an honest P_i, the simulator sets $f^{i,\ell}_k(x) = S^{i,\ell}(x,i)$ using $S^{i,\ell}(x,y)$ computed as described. By construction of the polynomials $S^{i,\ell}(x,y)$ and the sampler D, the result of the reconstruction phase would be exactly the output of $D(r)$, as is the output of all parties in the ideal execution. Below, we show that D is a valid sampler.

Proving that D is valid. We next show that D is a valid sampler, namely, that for a random r, $D(r)$ outputs the index of an honest party with some probability $\delta > 1/2$. The proof is almost verbatim from [KK06].

Towards that end, define:

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

where H is the set of all parties that were honest at the end of phase 1 of the protocol. Recall that by the guarantees of multi-moderated secret sharing, even if a single honest party P_k holds $v^k_{d,j} = 1$ then in the execution where P_d is the dealer and P_j is the moderator, the honest parties would have an agreement on the reconstructed value.

Hence, by the properties of multi-moderated secret sharing, if reconstruction is successful and if $k \in \mathsf{succeeded}$, then for any honest P_i, P_j and any $1 \le \ell \le n$ we have that $c^i_{\ell\to k} = c^j_{\ell\to k}$, i.e., the outputs of P_i and P_j are the same in the reconstruction associated with the instance with P_ℓ as a dealer and P_k as the moderator. As a result, we can omit the superscripts i and j.

We claim that all values c_k for $k \in \mathsf{succeeded}$ are uniformly distributed in $\{1, \ldots, n^4\}$. Note that the set $\mathsf{succeeded}$ might contain parties that are controlled by the adversary, but acted honestly while moderating the sharing phases and therefore are also considered. Consider the value $c_{i \to k}$, that is, the instance of multi-moderated secret sharing where P_i is the dealer and P_k is the moderator. We have the following cases to consider:

1. The moderator P_k is honest, then all the honest parties see the same values $c_{1\to k}, \ldots, c_{n\to k}$ regardless of whether each respective dealer $P_1, \ldots, P_n$ is honest or not. Moreover, k is in the set $\mathsf{succeeded}_\ell$ of all honest parties P_ℓ. Furthermore,
   (a) If the dealer P_i is honest then $c_{i\to k}$ is uniformly distributed in $\{1, \ldots, n^4\}$. Moreover, the shares received by the corrupted parties are independent of $c_{i\to k}$.
   (b) If the dealer P_i is corrupted then $c_{i\to k}$ must have been chosen independently of all other values, due to the secrecy property of secret sharing.
2. The moderator P_k is corrupted, then all the honest parties see the same values $c_{1\to k}, \ldots, c_{n\to k}$ regardless of whether each respective dealer $P_1, \ldots, P_n$ is honest or not. However, k might not be in the set $\mathsf{succeeded}_\ell$ of all honest parties P_ℓ. Furthermore,
   (a) If the dealer P_i is honest then $c_{i\to k}$ is uniformly distributed in $\{1, \ldots, n^4\}$.
   (b) If the dealer P_i is corrupted then $c_{i\to k}$ must have been chosen independently of all other values.

We therefore conclude that all the c_k for $k \in \mathsf{succeeded}$ are distributed uniformly at random in $\{1, \ldots, n^4\}$. Let $\mathsf{HonestChosen}$ be the event where the index k for which c_k is minimal among all parties in $\mathsf{succeeded}$ is an index of an honest party. We have that:

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

□

8 Broadcast

8.1 Byzantine Agreement

In a Byzantine agreement, every party P_i holds initial input v_i and the following properties hold: **(Agreement):** All the honest parties output the same value; **(Validity):** If all the honest parties begin with the same input value v, then all the honest parties output v. We simply plug in our OLE in the Byzantine agreement of [KK06]. As described in Section 1.3, we present standalone functionalities for Byzantine agreement and broadcast, where the intricacies of sequential composition are tackled in [CCGZ19].

Functionality 8.1: $\mathcal{F}_{\mathsf{BA}}$ – Byzantine Agreement

The functionality is parameterized by the set of corrupted parties I.

1. The functionality receives from each honest party P_j its input $b_j \in \{0,1\}$. The functionality sends $(b_j)_{j \notin I}$ to the adversary.
2. The adversary sends a bit $\hat{b}$.
3. If there exists a bit b such that $b_j = b$ for every $j \notin I$, then set $y = b$. Otherwise, set $y = \hat{b}$.
4. Send y to all parties.

The protocol for byzantine agreement appears below, followed by the proof of its security.

Protocol 8.2: Π_{BA} **– Byzantine Agreement Protocol**

Input: Each party P_i holds a bit b_i.
Initialization: Each party initializes $\mathsf{decided}_i = \texttt{false}$ and $\mathsf{openToAcceptRandom} = \texttt{false}$. 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 received from P_j (if no value was received, use the value from the previous iteration; at the outset 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 $\mathsf{decided}_i = \texttt{true}$.
   (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 $\mathsf{decided}_i = \texttt{true}$.
   (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 $\mathsf{decided}_i = \texttt{false}$ then set $\mathsf{openToAcceptRandom}_i = \texttt{true}$.
   (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 $\mathsf{openToAcceptRandom}_i = \texttt{false}$.
   (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 $\mathsf{openToAcceptRandom}_i = \texttt{false}$.
   (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 7.2) and let ℓ_i be the output of P_i.
   (b) If $\mathsf{openToAcceptRandom}_i = \texttt{true}$, then set $b_i = b_{\ell_i}$.
   (c) If $\mathsf{decided}_i = \texttt{true}$, then output b_i and terminate. Otherwise, proceed to the next iteration.

Theorem 8.3. *Protocol Π_{BA} (Protocol 8.2) is a Byzantine agreement protocol tolerating t malicious parties that works in constant expected rounds and requires the transmission of $\mathcal{O}(n^4 \log n)$ bits in expectation for $n \geq 3t + 1$.*

Proof. Simulation is straightforward: since the simulator receives all the inputs of the honest parties it can perfectly simulate them. Moreover, the simulator can also perfectly simulate the $\mathcal{F}_{\mathsf{OLE}}$. It just receives a (valid) sampler algorithm from the adversary, runs it, and gives the randomness it used to the adversary, while also receiving the outputs of each honest party. Thus, the view of the adversary is identical between the real and ideal executions.

The simulator then sees the output of the simulated honest parties $\hat{b}$, and send that bit $\hat{b}$ to the trusted party. We will next show that all simulated honest parties must output the same bit (this is essentially the "agreement" property). The trusted party then decides what to send to the honest parties. If all honest parties sent the same input b to the trusted party, then it ignores the bit that the simulator had sent it and just output b (this is essentially the "validity" property). Otherwise, it outputs $\hat{b}$.

We now show that the protocol satisfies agreement and validity. This in particular shows that the output of the simulated honest parties in the simulated execution is the same as the output of the honest parties in the ideal execution. Moreover, this also implies that the output in the ideal execution is identical to the real.

The proof of agreement and validity here is taken almost verbatim from [KK06]. The proofs of the following properties can be found in [KK06] and we give them here for completeness.

Claim 8.4. *Protocol Π_{BA} (Protocol 8.2) satisfies the following properties:*

Property I: *If at the beginning of some iteration, all (remaining) honest parties P_i hold the same value $b_i = b$, then all honest parties who have not yet terminated will output b and terminate the protocol at the end of the iteration.*

Property II: *If some party P_i sets $\mathsf{decided}_i = \texttt{true}$ at some iteration, then by the end of that iteration, each honest party P_j that has not yet terminated holds $b_j = b_i$, regardless of the result of the OLE protocol.*

Property III: *If an honest party P_i sets $\mathsf{openToAcceptRandom}_i = \texttt{false}$ in some iteration and holds a bit $b_i = b$, then all honest parties that have not yet terminated hold $b_j = b_i = b$ by the end of Round V of that iteration.*

Property IV: *If some party P_i terminates with output $b_i = b$, then all honest parties terminate with identical output in either the current iteration or in the next one, regardless of the results of the OLE protocol.*

Property V: *If an honest leader P_ℓ is elected in Round VI of some iteration, then all honest parties P_i terminate by the end of the next iteration.*

Property I. Assume that $b_i = b = 0$ at the beginning of the iteration for every honest party P_i. Then, it holds that $|\mathcal{S}_i^0| \geq n - t$ and hence P_i sets $\mathsf{decided}_i = \texttt{true}$ at Step 2b. This implies that b_i cannot be changed in Round III and remains 0. Consequently, in Round IV, P_i sets $\mathsf{openToAcceptRandom}_i = \texttt{false}$ and b_i stays 0 (as $|\mathcal{S}_i^0| \geq n - t$). Further, b_i remains unchanged in Round V (since $|\mathcal{S}_i^1| \leq t$). Finally, in Round VI the parties run the OLE protocol, but ignore its value since $\mathsf{openToAcceptRandom}_i = \texttt{false}$. Since $\mathsf{decided}_i = \texttt{true}$, every honest P_i outputs $b_i = 0$ and terminates.

The case where all parties start with $b_i = b = 1$ is shown analogously.

Property II. Assume that P_i sets $\mathsf{decided} = \texttt{true}$ at Step 2b. This implies that $|\mathcal{S}_i^0| \geq n - t$. Since at most t parties from $\mathcal{S}_i^0$ can be corrupt, for every other honest party P_j it holds that $\left||\mathcal{S}_i^0| - |\mathcal{S}_j^0|\right| \leq t$ and thus $|\mathcal{S}_j^0| \geq n - 2t \geq t + 1$. Hence, at the end of Step 2b every honest party P_j sets $b_j = b_i = 0$. All honest parties then send their new bits in Step 2c, and thus at Round III we have that $|\mathcal{S}_j^1| \leq t$ for every honest P_j and therefore b_j remains 0. As a result, in Round IV all honest parties set $\mathsf{openToAcceptRandom}_j = \texttt{false}$, and b_j remains 0 for all honest parties at the end of Round V. In Round VI, parties run the OLE protocol, but ignore its value since $\mathsf{openToAcceptRandom}_i = \texttt{false}$.

Consequently, in the next round, all honest parties (that did not terminate) start with the same input, and as follows from Property I, all terminate with the same value as the output. An analogous argument can be shown for the case when P_i sets $\mathsf{decided}_i = \texttt{true}$ at Step 3b.

Property III. A party P_i sets $\mathsf{openToAcceptRandom}_i = \texttt{false}$ if $|\mathcal{S}_i^0| \geq n - t$ in Step 4c. As shown in the proof of the previous property, this implies that for every other honest party P_j it holds that $|\mathcal{S}_j^0| \geq t + 1$, and thus P_j sets $b_j = 0$ (although it might keep the flag $\mathsf{openToAcceptRandom}_i = \texttt{true}$ if $|\mathcal{S}_j^0| < n - t$). The value b_j does not change during round V, from a similar reasoning as in the previous claim.

A similar argument holds for the case when P_i sets $\mathsf{openToAcceptRandom}_i = \texttt{false}$ in Step 5b.

Property IV. A party P_i terminates only when $\mathsf{decided}_i = \texttt{true}$. Property II shows that all other honest parties P_j would hold $b_j = b_i = b$ at the end of the iteration, while some might terminate. Further, by virtue of Property I, all the honest parties which do not terminate at the end of the iteration are guaranteed to terminate by the end of the next iteration.

Property V. When an honest leader P_ℓ is elected in Round VI, every honest party P_j obtains the same value $\ell_j = \ell$ as the output of Π_{OLE}. Moreover, all the honest parties must have received the same bit b_ℓ from an honest P_l in the prior rounds. If all the honest parties P_j hold $\mathsf{openToAcceptRandom}_j = \texttt{true}$, then all set $b_j = b_\ell$, and thus begin the next iteration with the same value. By virtue of Property I, this implies that all honest parties output b_ℓ by the end of the next iteration. Otherwise, if some honest party P_i has $\mathsf{openToAcceptRandom}_i = \texttt{false}$, then due to Property III it holds that $b_\ell = b_i$ by the end of Round V. Thus, every other honest party P_j sets $b_j = b_\ell = b_i$ in Round VI. Similar to the prior case, this implies that all the (remaining) honest parties begin the next iteration with the same value and hence output b_ℓ by the end of the next iteration.

It is guaranteed that the OLE protocol (Π_{OLE}, Protocol 7.2) elects an honest leader with constant probability as shown in Theorem 7.3. It thus follows that agreement is reached in expected number of iterations, where each iteration requires only a constant number of rounds.

Efficiency. In each iteration the parties send $O(n^2)$ bits over the point-to-point channels, and then run OLE protocol (Π_{OLE}, Protocol 7.2), which requires $O(n^4 \log n)$ bits of communication over point-to-point channels. □

8.2 Broadcast and Parallel-broadcast

In a broadcast protocol, a distinguished dealer $P^* \in \mathcal{P}$ holds an initial input M and the following hold: **(Agreement):** All honest parties output the same value; **Validity:** If the dealer is honest, then all honest parties output M. We formalize it using the following functionality:

Functionality 8.5: $\mathcal{F}_{\mathsf{BC}}$

The functionality is parametrized with a parameter L.

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

To implement this functionality, the dealer just gradecasts its message M and then parties run Byzantine agreement on the grade they received, while parties use input 1 for the Byzantine agreement if and only if the grade of the gradecast is 2. If the output of the Byzantine agreement is 1, then they output the message they received in the gradecast, and otherwise, they output $\perp$. We simply plug in our gradecast and Byzantine agreement in the above protocol. Note that the above communication complexity is asymptotically free (up to the expectation) for $L > n^3 \log n$.

Protocol 8.6: Π_{BC}**– Broadcast Protocol for a single dealer**

- **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 the input of P_i is 1 if $g = 2$, and otherwise the input is 0.
- **Output:** If the output of the Byzantine agreement is 1 then output M'. Otherwise, output $\perp$.

Theorem 8.7. *Protocol 8.6 is a secure broadcast tolerating $t < n/3$ malicious parties. For an input message M of length L bits, the protocol requires $\mathcal{O}(nL)$ plus expected $\mathcal{O}(n^4 \log n)$ bits total communication, and expected constant number of rounds.*

Proof. We prove the protocol in the $\mathcal{F}_{\mathsf{BA}}$-$\mathcal{F}_{\mathsf{Gradecast}}$ hybrid model.

The case of a corrupted sender. In case of a corrupted sender, the simulator simulates the $\mathcal{F}_{\mathsf{Gradecast}}$ functionality and receives from the adversary either:

1. $(\mathsf{ExistsGrade2}, M, (g_j)_{j \notin I})$. Then, simulate the $\mathcal{F}_{\mathsf{BA}}$ functionality where all honest parties input either 1 or 0 according to $(g_j)_{j \notin I}$, where note that $\mathcal{F}_{\mathsf{Gradecast}}$ guarantees that all $g_j \geq 1$ and there exists at least one index for which $g_j = 2$. The $\mathcal{F}_{\mathsf{BA}}$ sends to the adversary all the bits of the honest parties and then receives back one bit, $\hat{b}$. If $g_j = 2$ for every j, or if $\hat{b} = 1$, then the simulator sends M to $\mathcal{F}_{\mathsf{BC}}$. Otherwise, it sends $\perp$ to $\mathcal{F}_{\mathsf{BC}}$. $\mathcal{F}_{\mathsf{BC}}$ forwards the chosen message to all parties.
2. $(\mathsf{NoGrade2}, (M_j)_{j \notin I}, (g_j)_{j \notin I})$. This time, it is guaranteed that all honest parties have $g_j \leq 1$. Then, simulate the $\mathcal{F}_{\mathsf{BA}}$ functionality where all honest parties input 0 to $\mathcal{F}_{\mathsf{BA}}$. This implies that the output of $\mathcal{F}_{\mathsf{BA}}$ is $\perp$ to all parties. The simulator then sends $\perp$ to $\mathcal{F}_{\mathsf{BC}}$, which forwards that message to all parties.

From inspection, the view of the adversary in the real and ideal is identical. Likewise, the output of the honest parties.

The case of an honest sender. In this case, the simulator receives M from the trusted party. It then simulates the $\mathcal{F}_{\mathsf{Gradecast}}$ sending $(M, 2)$ to all corrupted parties. Next, it simulates $\mathcal{F}_{\mathsf{BA}}$, considering all honest parties send 1 to $\mathcal{F}_{\mathsf{BA}}$. It receives some bit $\hat{b}$ from the adversary which it ignores as its input to $\mathcal{F}_{\mathsf{BA}}$, and simulates the output of $\mathcal{F}_{\mathsf{BA}}$ to be 1.

From inspection, it is easy to see that the joint distribution of the view of the adversary and the outputs of the honest parties in the real is identically distributed to the view and the outputs in the ideal.

Efficiency. The protocol gradecasts a message which requires $O(nL)$ bits of communication and runs in constant rounds. In addition, we run Byzantine agreement, which requires expected $O(n^4 \log n)$ bits of communication in expected constant rounds.

□

Parallel Broadcast. Parallel broadcast relates to the case where n parties wish to broadcast a message of size L bits in parallel. In that case, we rely on an idea of Fitzi and Garay [FG03] that applies to OLE-based protocols. The idea is that the multiple broadcast sub-routines are run in parallel when only a single election per iteration is required for all these sub-routines. This results in the following corollary:

Corollary 8.8. *There exists a perfectly secure parallel-broadcast with optimal resilience, which allows n parties to broadcast messages of size L bits each, at the cost of $\mathcal{O}(n^2 L)$ bits communication, plus $\mathcal{O}(n^4 \log n)$ expected communicating bits. The protocols runs in constant expected number of rounds.*

For completeness, we provide the functionality for parallel broadcast below, and omit the proof since it follows from broadcast.

Functionality 8.9: $\mathcal{F}_{\mathsf{BC}}^{\mathsf{parallel}}$

The functionality is parametrized with a parameter L.

1. Each $P_i \in \mathcal{P}$ sends the functionality its message $M_i \in \{0,1\}^L$.
2. The functionality sends to all parties the message $\{M_i\}_{i \in \{1,\ldots,n\}}$.

Efficiency. The protocol gradecasts n messages, each of which requires $O(nL)$ bits of communication and runs in constant rounds. In addition, we run Byzantine agreement where a single leader election per iteration is necessary across all the instances, which requires expected $O(n^4 \log n)$ bits of communication in expected constant rounds.

References

[AAY21] Ittai Abraham, Gilad Asharov, and Avishay Yanai. Efficient perfectly secure computation with optimal resilience. In *Theory of Cryptography Conference*, 2021.

[ACP21] C Anirudh, Ashish Choudhury, and Arpita Patra. A survey on perfectly-secure verifiable secret-sharing. *Cryptology ePrint Archive*, 2021.

[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.

[ADD+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security, 2019*, volume 11598, pages 320–334. Springer, 2019.

[AKP20] Benny Applebaum, Eliran Kachlon, and Arpita Patra. The round complexity of perfect mpc with active security and optimal resiliency. In *Annual Symposium on Foundations of Computer Science (FOCS)*, 2020.

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

[ALR11] Gilad Asharov, Yehuda Lindell, and Tal Rabin. Perfectly-secure multiplication for any $t < n/3$. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, 2011.

[AN21] Ittai Abraham and Kartik Nayak. Crusader agreement with $\leq 1/3$ error is impossible for $n \leq 3f$ if the adversary can simulate. *Decentralized Thoughts, Blog Post*, 2021. https://tinyurl.com/decentralizedthougts, accessed: September 2021.

[BCG93] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of ACM symposium on Theory of computing*, 1993.

[BE03] Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[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.

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

[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.

[BTH08] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *Theory of Cryptography Conference*, 2008.

[Can93] Ran Canetti. Asynchronous secure computation. *Technion - Computer Science Department - Technical Report*, CS0755, 1993.

[Can96] Ran Canetti. *Studies in secure multiparty computation and applications*. PhD thesis, Citeseer, 1996.

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

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[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.

[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.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 2000.

[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.

[Che21] Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *DISC*, 2021.

[CHP13] Ashish Choudhury, Martin Hirt, and Arpita Patra. Asynchronous multiparty computation with linear communication complexity. In Yehuda Afek, editor, *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, volume 8205 of *Lecture Notes in Computer Science*, pages 388–402. Springer, 2013.

[CP16] Ashish Choudhury and Arpita Patra. An efficient framework for unconditionally secure multiparty computation. *IEEE Transactions on Information Theory*, 2016.

[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.

[DDGN14] Ivan Damgård, Bernardo David, Irene Giacomelli, and Jesper Buus Nielsen. Compact vss and efficient homomorphic uc commitments. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 213–232. Springer, 2014.

[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.

[DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 1985.

[DXR21] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *ACM CCS Conference on Computer and Communications Security*, 2021.

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

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

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

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 1997.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 699–710. ACM, 1992.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *ACM symposium on Theory of computing*, 2001.

[GLS19] Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional mpc with guaranteed output delivery. In *Annual International Cryptology Conference*, 2019.

[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.

[GP21] Chaya Ganesh and Arpita Patra. Optimal extension protocols for byzantine broadcast and agreement. *Distributed Computing*, 34(1):59–77, 2021.

[GRR98] Rosario Gennaro, Michael O Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *ACM symposium on Principles of distributed computing*, 1998.

[HMP00] Martin Hirt, Ueli Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In *International conference on the theory and application of cryptology and information security*, 2000.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 466–485. Springer, 2010.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Annual International Cryptology Conference*, 2006.

[KKK08] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. Improving the round complexity of vss in point-to-point networks. In *International Colloquium on Automata, Languages, and Programming*, 2008.

[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.

[lat] Aws latency monitoring. accessed February, 2022.

[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.

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

[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.

[Pat11] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *International Conference On Principles Of Distributed Systems*, 2011.

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

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

[SBKN21] Nibesh Shrestha, Adithya Bhat, Aniket Kate, and Kartik Nayak. Synchronous distributed key generation without broadcasts. *IACR Cryptol. ePrint Arch.*, page 1635, 2021.

[TC84] Russell Turpin and Brian A Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[TLP20] Georgios Tsimos, Julian Loss, and Charalampos Papamanthou. Gossiping for communication-efficient broadcast. *Cryptology ePrint Archive*, 2020.