Asymptotically Free Broadcast in Constant Expected Time via Packed VSS

Ittai Abraham[1], Gilad Asharov[2], Shravani Patil[3], and Arpita Patra[3]

[1] VMWare Research
iabraham@vmware.com
[2] Department of Computer Science, Bar-Ilan University, Israel
Gilad.Asharov@biu.ac.il
[3] Indian Institute of Science, Bangalore, India
{shravanip,arpita}@iisc.ac.in

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.

Keywords: MPC · Byzantine Agreement · Broadcast

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 [44,48], even when a constant error probability is allowed [4].

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 [32]. Therefore, a protocol that runs in strict constant number of rounds is impossible to achieve. The seminal works of Rabin and Ben-Or [49,10] 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 [22,13] requires $\mathcal{O}(n^2)$ bits of communication complexity, which is asymptotically optimal for any deterministic broadcast protocols [27], 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 [20].[4] 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 Mi-

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

cali [29], followed by substantial improvements and simplifications by Katz and Koo [41], requires significant communication complexity of $\mathcal{O}(n^6 \log n)$ bits in expectation for broadcasting just a single bit, or $\mathcal{O}(n^2 L + n^6 \log n)$ bits for a message of L bits.[5] However, they work in *expected constant 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 [1]), 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 [12] 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 [18,6]) and secure under universal composition [43]. 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 [22,13,41,47,46,20]. 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 [41,3,50].

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 $O(n)$

[5] Using broadcast extension of [46] 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 $n = 100$.

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 [41] achieved $\mathcal{O}(n^2 L)$ 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 $\mathcal{O}(nL)$ bits. Thus, we say that our protocol is asymptotically free for messages of size $L = \Omega(n^3 \log n)$ bits. We recall that [41] together with [46] 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 [41]). 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 [12], 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.[6] In fact, the notion of parallel-broadcast goes back to the work of Pease et al. [48]. 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 [22,13,20] 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.

[6] 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 [2] it has been shown how to reduce it to $\mathcal{O}(1)$ VSSs per party, i.e., each party might have to broadcast $\mathcal{O}(n^2 \log n)$.

Task	Reference	Total P2P (in bits)	Rounds
$1 \times \mathcal{BC}(L)$	[22,13]	$\mathcal{O}(n^2L)$	$\mathcal{O}(n)$
	[22,13] + [20]	$\mathcal{O}(nL + n^2 \log n)$	$\mathcal{O}(n)$
	[41]	$\mathcal{O}(n^2L) + E(\mathcal{O}(n^6 \log n))$	$E(\mathcal{O}(1))$
	[41] + [46]	$\mathcal{O}(nL) + E(\mathcal{O}(n^7 \log n))$	$E(\mathcal{O}(1))$
	Our work	$\boldsymbol{\mathcal{O}(nL) + E(\mathcal{O}(n^4 \log n))}$	$\boldsymbol{E(\mathcal{O}(1))}$
$n \times \mathcal{BC}(L)$	[22,13]	$\mathcal{O}(n^3L)$	$\mathcal{O}(n)$
	[41]	$\mathcal{O}(n^3L) + E(\mathcal{O}(n^6 \log n))$	$E(\mathcal{O}(1))$
	[41] + [46][7]	$\mathcal{O}(n^2L) + E(\mathcal{O}(n^7 \log n))$	$E(\mathcal{O}(1))$
	Our work	$\boldsymbol{\mathcal{O}(n^2L) + E(\mathcal{O}(n^4 \log n))}$	$\boldsymbol{E(\mathcal{O}(1))}$

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.

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. [21]. 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 [12,30]. The dealer then has to communicate $\mathcal{O}(n^2)$ field elements to share its secret. An intriguing question is 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.*

[7] Since the broadcast extension protocol of [20] requires $\mathcal{O}(n)$ rounds, combining [41] with [20] results in linear-round complexity and a worse communication complexity than what the second row ([22,13] + [20]) provides.

The best previous results achieve $\mathcal{O}(n^3 \log n)$ (point-to-point and broadcast) for sharing $\mathcal{O}(n)$ secret elements [12,30,42,5], 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 [34]. 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. [25]. However, this 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).

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 [29] ("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.*

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 [29] 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 [12,19,37,24,8,2] 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 [41], 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 [40,9,39] in perfectly-secure MPC is based on the *player elimination* framework (introduced by Hirt and Maurer and Przydatek [40]). 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 [22,13], or to $\mathcal{O}((n|C| + n^7)\log n)$ communication complexity with $\mathcal{O}(n + \mathsf{depth}(C))$ rounds (expected) using [41]. 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 [29] and Katz and Koo [41] (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 [45], Katz and Koo [41] and Cohen et al. [23]. Regarding parallel composition, unlike the black-box parallel composition of broadcasts studied by Ben-Or and El-Yaniv [11], we rely on the idea of Fitzi and Garay [33] 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 [16]. 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. [23] 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 [23], 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 [38]: Specifically, after $\mathcal{O}(n)$ attempts to terminate, the parties can run the $\mathcal{O}(n)$ rounds protocol with guaranteed termination. See also [23].

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 [44,48]. Moreover, Fischer and Lynch [32] showed a lower bound of $t + 1$ rounds for any deterministic byzantine agreement protocol or broadcast protocol. Faced with this barrier, Rabin [49] and Ben-Or [10] independently studied the effect of randomization on round complexity, which eventually culminated into the work of Feldman and Micali [31] who gave an expected constant round protocol for byzantine agreement with optimal resilience. Improving over this work, the protocol of [41] 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 [28] established a lower bound of n^2 bits for deterministic broadcast or agreement on a single bit. With a round complexity of $\mathcal{O}(n)$, [22,13] 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 [47,35] communicates $\mathcal{O}(nL)$ bits over point-to-point channels and $\mathcal{O}(n^2)$ bits through a bit-broadcast protocol. The work of [46] 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 [20] 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 $\mathcal{O}(n)$.

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

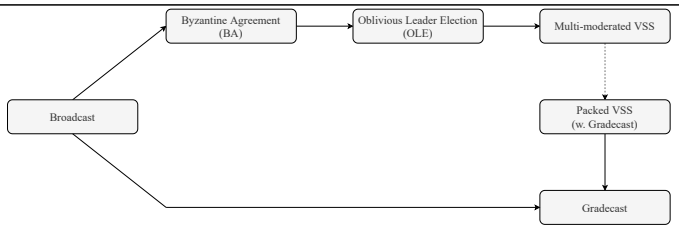

Fig. 1: Roadmap of our building blocks. All lines are compositions, except for the line from Multi-moderated VSS to Packed VSS, which is a white-box modification.

2.1 Improved Broadcast in Constant Expected Rounds

Our starting point is a high-level overview of the broadcast protocol of Katz and Koo [41], which simplifies and improves the construction of Feldman and Micali [29]. Following the approach of Turpin and Coan [51] 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 [29,41] 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 [41], 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 [12,36,7]. 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.

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[8] $S(x,y)$ 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 [2]. Specifically, the work of [2] considers the VSS protocol of [12] 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 [2] 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 [2] secret sharing while pointing out that the protocol is a variant of the VSS protocol of [12]. 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

[8] 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.

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 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 $j \notin$ 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 [26], 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 [29,41] (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 [14]. 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.

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 standard, stand-alone simulation-based model in the perfect setting [16,7] for a static adversary. Owing to the results of [18], this guarantees adaptive security with inefficient simulation. We derive universal composability [17] for free using [43]. We refer the readers to the full version for the security proofs and more details.

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 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 $P_i \in \mathcal{P}$.

3.2 Finding (n, t)-STAR

Definition 3.1. *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: (a) $|C| \geq n-2t$*; (b)* $|D| \geq n-t$*; and (c) for every* $j \in C$ *and every* $k \in D$*, the edge* (j,k) *exists in* G*.*

Canetti [14,15] 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. We refer the readers to the full version for details.

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 security proof appears in the full version of the paper.

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 VSS 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 $\mathsf{happy}_i = 1$ [9].

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.

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

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 $t < n/3$.*

Lemma 4.4. *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.*

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 [29,31]. The security of our protocol is stated in Theorem 5.3 and the proof appears in the full version.

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 **??**) 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.*

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 5.4. *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 $\mathcal{O}(n^3 \log n)$).*

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 5.5. *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^k_M. The functionality is 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 $v^k_{M_j} = 1$ for every $k \in \{1, \ldots, n\}$. Moreover:
      i. If the dealer is honest, then set $\mathsf{d}^k_{M_j} = \mathsf{accept}$ 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 $\mathsf{d}^k_{M_j} = \mathsf{accept}$ for every $k \in \{1, \ldots, n\}$. In any other case, set $\mathsf{d}^k_{M_j} = \mathsf{reject}$ for every $k \in \{1, \ldots, n\}$.
   (b) If the moderator M_j is corrupt then receive m_j from the adversary.
      i. If $m_j = (\mathsf{Agreement}, (v^k_{M_j})_{k \notin I}), \mathsf{d}_{M_j})$ where $\mathsf{d}_{M_j} \in \{\mathsf{accept}, \mathsf{reject}\}$, and for some $k \notin I$ it holds that $v^k_{M_j} = 1$. Set $(v^k_{M_j})_{k \notin I}$ as received from the adversary. Verify that $S(x, y)$ is $(2t, t)$-polynomial. If not, set $\mathsf{d}^k_{M_j} = \mathsf{reject}$ for every $k \notin I$. Otherwise, set $\mathsf{d}^k_{M_j} = \mathsf{d}_{M_j}$ for every $k \notin I$.
      ii. If $m_j = (\mathsf{NoAgreement}, (\mathsf{d}^k_{M_j})_{k \notin I})$ where each $\mathsf{d}^k_{M_j} \in \{\mathsf{accept}, \mathsf{reject}\}$, then set $v^k_{M_j} = 0$ for every $k \in \{1, \ldots, n\}$ and $\mathsf{d}^1_{M_j}, \ldots, \mathsf{d}^n_{M_j}$ as received from the adversary.
4. **Output:** Each honest party P_k ($k \notin I$) receives as output $f_i(x), g_i(y)$, $(\mathsf{d}^k_M)_{M \in \mathcal{M}}$, and flags $(v^k_M)_{M \in \mathcal{M}}$.

To clarify, each party P_i receives global shares for all moderators, and an output d^i_M and flag v^i_M for 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. *Protocol 6.2 computes $\mathcal{F}_{\mathsf{mm\text{-}pVSS}}$ (Functionality 6.1) in the presence of a malicious adversary corrupting at most $t < n/3$ 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.*

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 6.4: $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}pVSS}}$ – 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 $\mathsf{d}^i_M = \mathsf{accept}$, 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 $s^i_M = g_{\beta^*}(0)$. If there is not unique decoding, then set $s^i_M = 0$.
   (b) If $\mathsf{d}^i_M = \mathsf{reject}$, then set $s^i_M = 0$.
3. **Output:** Output $(s^i_M)_{M \in \mathcal{M}}$.

Theorem 6.5. *For each moderator $M \in \mathcal{M}$, if there exists an honest party with $v^k_M = 1$ then all honest parties hold the same $s^{k'}_M = s^k_M$.*

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, $\Pi^{\mathsf{Rec}}_{\mathsf{mm\text{-}pVSS}}$ (Fig. 6.4) of each of the above nT instances of multi-moderated

packed secret sharing is run in parallel to reconstruct the secrets previously shared.
Let $c^k_{i\to j}$ 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 $c^k_j = \sum_{i=1}^{n} c^k_{i\to j} \bmod n^4$ and outputs j that minimizes c^k_j among all $j \in \mathsf{succeeded}_k$ (break ties arbitrarily).

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

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 [41]. As described in Section 1.3, we present standalone functionalities for Byzantine agreement and broadcast, where the intricacies of sequential composition are tackled in [23]. The protocol for byzantine agreement (Π_{BA}) which follows from [41] and its proof of security appear in the full version of the paper.

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.

Theorem 8.2. *Protocol* Π_{BA} *is a Byzantine agreement protocol tolerating* $t < n/3$ *malicious parties that works in constant expected rounds and requires the transmission of* $\mathcal{O}(n^2)$ *bits plus expected* $\mathcal{O}(n^4 \log n)$ *bits of communication.*

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 8.3: $\mathcal{F}_{\mathsf{BC}}$

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 protocol below. Note that the communication complexity our protocol is asymptotically free (up to the expectation) for $L > n^3 \log n$.

Protocol 8.4: 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 8.5. *Protocol 8.4 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 constant expected 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 [33] 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 8.6. *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 8.7: $\mathcal{F}_{\mathsf{BC}}^{\mathsf{parallel}}$

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.

Acknowledgements

Gilad Asharov is 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. Shravani Patil would like to acknowledge the support of DST National Mission on Interdisciplinary Cyber-Physical Systems (NM-ICPS) 2020-2025. Arpita Patra would like to acknowledge the support of DST National Mission on Interdisciplinary Cyber-Physical Systems (NM-ICPS) 2020-2025, Google India Faculty Award, and SERB MATRICS (Theoretical Sciences) Grant 2020-2023.

References

1. Aws latency monitoring https://www.cloudping.co/grid, accessed February, 2022
2. Abraham, I., Asharov, G., Yanai, A.: Efficient perfectly secure computation with optimal resilience. In: Theory of Cryptography Conference (2021)
3. Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience. In: Financial Cryptography and Data Security (2019)
4. Abraham, I., Nayak, K.: 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
5. Applebaum, B., Kachlon, E., Patra, A.: The round complexity of perfect mpc with active security and optimal resiliency. In: FOCS (2020)
6. Asharov, G., Cohen, R., Shochat, O.: 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)

7. Asharov, G., Lindell, Y.: A full proof of the bgw protocol for perfectly secure multiparty computation. Journal of Cryptology (2017)
8. Asharov, G., Lindell, Y., Rabin, T.: Perfectly-secure multiplication for any $t < n/3$. In: Advances in Cryptology - CRYPTO 2011 (2011)
9. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Theory of Cryptography Conference (2008)
10. Ben-Or, M.: Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In: PODC (1983)
11. Ben-Or, M., El-Yaniv, R.: Resilient-optimal interactive consistency in constant time. Distributed Computing (2003)
12. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: ACM Symposium on Theory of Computing (1988)
13. Berman, P., Garay, J.A., Perry, K.J.: Bit optimal distributed consensus. In: Computer science (1992)
14. Canetti, R.: Asynchronous secure computation. Technion - Computer Science Department - Technical Report (1993)
15. Canetti, R.: Studies in secure multiparty computation and applications. Ph.D. thesis, Citeseer (1996)
16. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. (2000)
17. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS (2001)
18. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: On adaptive vs. non-adaptive security of multiparty protocols. In: EUROCRYPT 2001 (2001)
19. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: ACM Symposium on Theory of Computing (1988)
20. Chen, J.: Optimal error-free multi-valued byzantine agreement. In: DISC (2021)
21. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science (1985)
22. Coan, B.A., Welch, J.L.: Modular construction of nearly optimal byzantine agreement protocols. In: ACM Symposium on Principles of distributed computing (1989)
23. Cohen, R., Coretti, S., Garay, J.A., Zikas, V.: Probabilistic termination and composability of cryptographic protocols. J. Cryptol. (2019)
24. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: EUROCRYPT (2000)
25. Damgård, I., David, B., Giacomelli, I., Nielsen, J.B.: Compact vss and efficient homomorphic uc commitments. In: ASIACRYPT (2014)
26. Das, S., Xiang, Z., Ren, L.: Asynchronous data dissemination and its applications. In: ACM CCS Conference on Computer and Communications Security (2021)
27. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. In: Symposium on Principles of Distributed Computing (1982)
28. Dolev, D., Reischuk, R.: Bounds on information exchange for byzantine agreement. Journal of the ACM (JACM) (1985)
29. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: ACM Symposium on Theory of Computing (1988)
30. Feldman, P.N.: Optimal Algorithms for Byzantine Agreement. Ph.D. thesis, Massachusetts Institute of Technology (1988)
31. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM Journal on Computing (1997)

32. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. Information Processing Letters (1982)
33. Fitzi, M., Garay, J.A.: Efficient player-optimal protocols for strong and differential consensus. In: PODC (2003)
34. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: ACM Symposium on Theory of Computing (1992)
35. Ganesh, C., Patra, A.: Optimal extension protocols for byzantine broadcast and agreement. Distributed Computing (2021)
36. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC (2001)
37. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In: PODC (1998)
38. Goldreich, O., Petrank, E.: The best of both worlds: Guaranteeing termination in fast randomized byzantine agreement protocols. Inf. Process. Lett. (1990)
39. Goyal, V., Liu, Y., Song, Y.: Communication-efficient unconditional mpc with guaranteed output delivery. In: Annual International Cryptology Conference (2019)
40. Hirt, M., Maurer, U., Przydatek, B.: Efficient secure multi-party computation. In: ASIACRYPT (2000)
41. Katz, J., Koo, C.: On expected constant-round protocols for byzantine agreement. In: Annual International Cryptology Conference (2006)
42. Katz, J., Koo, C.Y., Kumaresan, R.: Improving the round complexity of vss in point-to-point networks. In: International Colloquium on Automata, Languages, and Programming (2008)
43. Kushilevitz, E., Lindell, Y., Rabin, T.: Information-theoretically secure protocols and security under composition. In: STOC (2006)
44. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. (1982)
45. Lindell, Y., Lysyanskaya, A., Rabin, T.: Sequential composition of protocols without simultaneous termination. In: PODC (2002)
46. Nayak, K., Ren, L., Shi, E., Vaidya, N.H., Xiang, Z.: Improved extension protocols for byzantine broadcast and agreement. arXiv preprint arXiv:2002.11321 (2020)
47. Patra, A.: Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In: International Conference On Principles Of Distributed Systems (2011)
48. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM (JACM) (1980)
49. Rabin, M.O.: Randomized byzantine generals. In: Symposium on Foundations of Computer Science (1983)
50. Shrestha, N., Bhat, A., Kate, A., Nayak, K.: Synchronous distributed key generation without broadcasts. IACR Cryptol. ePrint Arch. (2021)
51. Turpin, R., Coan, B.A.: Extending binary byzantine agreement to multivalued byzantine agreement. Information Processing Letters (1984)