Broadcast-Efficient Secure Multiparty Computation

Juan Garay* Clint Givens† Rafail Ostrovsky‡

Abstract

Secure multiparty computation (MPC) is perhaps the most popular paradigm in the area of cryptographic protocols. It allows several mutually untrustworthy parties to jointly compute a function of their private inputs, without revealing to each other information about those inputs. In the case of unconditional (information-theoretic) security, protocols are known which tolerate a dishonest minority of players, who may coordinate their attack and deviate arbitrarily from the protocol specification.

It is typically assumed in these results that parties are connected pairwise by authenticated, private channels, and that in addition they have access to a "broadcast" channel. Broadcast allows one party to send a consistent message to all other parties, guaranteeing consistency even if the broadcaster is corrupted. Because broadcast cannot be simulated on the point-to-point network when more than a third of the parties are corrupt, it is impossible to construct general MPC protocols in this setting without using a broadcast channel (or some equivalent addition to the model).

A great deal of research has focused on increasing the efficiency of MPC, primarily in terms of round complexity and communication complexity. In this work we propose a refinement of the round complexity which we term *broadcast complexity*. We view the broadcast channel as an expensive resource and seek to minimize the number of rounds in which it is invoked.

1. We construct an MPC protocol which uses the broadcast channel only *three* times in a preprocessing phase, after which it is never required again. To the best of our knowledge, ours is the first unconditionally secure MPC protocol for $t < n/2$ to achieve a *constant* number of broadcast rounds. In contrast, the best previous protocols we are aware of require $\Omega(\min\{n, D\})$ broadcast rounds, for n parties and circuit of depth D.

2. In the negative direction, we show a lower bound of *two* broadcast rounds for the specific functionality of Weak Secret Sharing (a.k.a. Distributed Commitment), also a very natural functionality and central building block of many MPC protocols.

The broadcast-efficient MPC protocol relies on new constructions of Pseudosignatures and Verifiable Secret Sharing, both of which might be of independent interest.

*AT&T Labs – Research. Email: `garay@research.att.com`.

†Department of Mathematics, UCLA. Email: `cgivens@math.ucla.edu`.

‡Departments of Computer Science and Mathematics, UCLA, 3732D Boelter Hall, Los Angeles CA 90095-1596, U.S. Email: `rafail@cs.ucla.edu` Supported in part by NSF grants 0830803, 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

1 Introduction

A number of mutually distrustful parties wish to jointly compute a function $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, where player P_i holds the private input x_i and receives output y_i. Under what conditions can they do so without revealing any information about their private inputs, other than what is implied by the function's output? Attempts to answer this question led to the development of an entire field of research in cryptography known as *secure multiparty computation* (MPC).

In an early triumph of the field, MPC researchers proved the following beautiful result: Assuming only the existence of pairwise private and authenticated channels, it is possible to construct *unconditionally*, a.k.a *information-theoretically secure* MPC protocols. These protocols remain robust and private even in the presence of a computationally unbounded adversary who *actively* corrupts up to $t < n/3$ participants, coordinating their attack and directing them to deviate arbitrarily from the protocol specification [BGW88, CCD88].

Our focus in this work is rather on unconditionally secure protocols with honest majority ($t < n/2$). Here an addition to the model is necessary, as secure channels by themselves no longer suffice. In fact when $t \geq n/3$, the specific functionality of Byzantine agreement/broadcast [LSP82, PSL80] is impossible to securely realize, even probabilistically. In some sense, however, the inability of players to reach consensus is the only obstacle, as a second early triumph of the field reveals: Namely, if we grant a "physical broadcast channel" (that is, a black-box which securely implements broadcast), then unconditionally secure MPC becomes possible once again for $t < n/2$ [RB89, Bea91b, CDD$^+$01]. (Even with a broadcast channel, protocols with $n/3 \leq t < n/2$ are subject to some (negligibly small) error probability [DDWY93, CCD88] and cannot achieve so-called perfect security, which is possible when $t < n/3$.[1] By contrast, when $t \geq n/2$, even *computationally* secure MPC is impossible without some additional physical assumptions [Cle86].)

Other assumptions may replace the physical broadcast channel to allow secure computation with $t \geq n/3$, the most common being the existence of a PKI setup and secure signatures. With these in hand, it becomes possible to simulate broadcast on point-to-point channels for any $t < n$ by means of an *authenticated broadcast* protocol. In a closely related vein, Pfitzmann and Waidner [PW96] introduced *pseudosignatures*, an information-theoretic analogue of digital signatures which tolerates an unbounded, active adversary. Given a pseudosignature setup, authenticated broadcast becomes possible for any $t < n$, and hence unconditional MPC is possible for $t < n/2$ by simulating physical broadcast with authenticated broadcast. However, the *construction* of pseudosignatures in [PW96] still relies on physical broadcast in a preprocessing phase. A key contribution of the current work is a dramatically more efficient pseudosignature construction in the case $t < n/2$ (the relevant case for unconditional MPC).

As with other distributed protocols, the efficiency of MPC is commonly measured in terms of round complexity and communication complexity. Our focus in this work is on a refinement of round complexity, namely *broadcast complexity*: the number of rounds in which the broadcast primitive is invoked.

High-level descriptions of MPC protocols tend to treat broadcast as a black-box. When $t < n/3$, this may be viewed simply as a convenient abstraction, since broadcast in any case can be simulated in a point-to-point network using Byzantine agreement. (Trouble comes, however, when analyzing round complexity: as observed in [KK07, Koo07, KKK08], Byzantine agreement is round-expensive, and the compilation from black-box broadcast to simulated broadcast blows up the number of rounds substantially.)

When $t < n/2$, the black-box treatment of broadcast is (as described above) no longer a convenience but a requirement. Nevertheless, we argue there are still good reasons to consider it more expensive than "mere" secure channels. Indeed the latter can be realized via the physical exchange (using trusted couriers) of large one-time pads between every pair of players, which may be done in an asynchronous preprocessing phase and without any centrally trusted party. By contrast, we see no equally straightforward approach to physically implement *broadcast* without a trusted party, and when the participants are geographically scattered. Hence it seems quite plausible to treat physical broadcast as an expensive resource, and in particular to treat a protocol's *broadcast rounds* as more expensive than ordinary rounds. Additionally, the question of how many broadcast rounds are required for MPC is compelling from a theoretical perspective.

[1] The result is mentioned also in [RB89, BGW88]; [CCD88] give an informal argument and [DDWY93] a formal proof.

Our results. Thus motivated to better understand the broadcast requirements of secure computation when $t < n/2$, in this work we present new upper and lower bounds on the broadcast complexity of MPC and Verifiable Secret Sharing (VSS).

In the positive direction, we present an MPC protocol for $t < n/2$ which uses only a constant number of broadcast rounds—namely, *three.* It is based on our construction of a constant-round, linear VSS which uses 3 broadcasts in the sharing phase and none in reconstruction—what we call a $(3, 0)$-broadcast VSS. In fact, we derive our MPC result by means of a broadcast-round-preserving reduction to *any* $(*, 0)$-broadcast linear VSS; hence a future construction of (in particular) a $(2, 0)$-broadcast linear VSS would translate directly to an MPC with two broadcasts. To our knowledge, ours is the first unconditional MPC with $t < n/2$ to run in *constant* number of broadcast rounds. The best previous protocols we are aware of use $O(\min\{n, D\})$ broadcast rounds, where D is the multiplicative depth of the circuit to be evaluated. As an additional attractive feature of our construction, all broadcasts occur in a preprocessing phase during which players need not know their inputs nor the function to be computed. (It shares this feature with the pseudosignature approach [PW96, CR90], which inspires our solution.)

On the negative side, we address an open question of Katz and Koo [KK07], who ask whether there exist any constant-round MPC protocols for $t < n/2$ using only a *single* broadcast round, that do *not* rely on a PKI. We show that when the adversary is computationally unbounded, the Weak Secret Sharing functionality in particular (and hence VSS, and MPC in general) requires at least *two* physical broadcast rounds for $t \geq \frac{2}{5}n$; in fact this lower bound holds independently of the overall round complexity of the WSS protocol!

Related work. The role of broadcast in multiparty protocols has been studied in a number of previous works. Katz *et al.* [KKK08, KK07, Koo07], seeking to improve overall round complexity when broadcast is simulated over point-to-point channels, construct constant-round protocols for VSS and MPC whose descriptions use only a single broadcast round. However, for $t < n/2$ they assume a PKI infrastructure (e.g. pseudosignatures) is already in place, whereas our (complementary) goal is precisely to generate such a setup using as few *physical* broadcasts as possible.

Fitzi *et al.* [FGMR02, FGH$^+$02], as well as Goldwasser and Lindell [GL05], consider broadcast and MPC protocols for $t < n$ which do not use physical broadcast at all (nor equivalent assumptions), but instead weaken the guarantees provided by the protocol. In particular these protocols are not robust and so may fail to deliver any output at all. On the other hand, the so-called *detectable broadcast* (and *detectable MPC*) protocols of [FGMR02, FGH$^+$02] do achieve consistency among honest players: either the broadcast (MPC) succeeds and all honest parties receive output, or it fails, in which case all honest parties agree that it failed. (Such protocols are especially interesting with regard to the light they shed on our lower bound—see second Remark in Appendix B.)

As we have seen, our MPC construction is based on pseudosignatures [PW96, CR90], about which we defer further details to Section 2.

To our knowledge, the most efficient VSS protocol in terms of broadcast rounds for $t < n/2$ is the $(2, 2)$-broadcast, $(3, 2)$-round protocol of Kumaresan *et al.* [KPC10], which is exponential-time and not (apparently) linear. They also give a $(3, 2)$-broadcast, $(4, 2)$-round VSS which is polynomial-time. Hence our $(3, 0)$-broadcast protocol improves overall broadcast complexity (although it is not as round-efficient at $(9, 1)$ rounds).

Turning now to survey the most broadcast-efficient MPC protocols in the literature, we first recall the general shape of an MPC protocol in the "share-compute-reveal" paradigm. (1) In the *share phase*, each P_i commits to his input using Verifiable Secret Sharing. (2) In the *compute phase*, the shared inputs are used to evaluate an arithmetic circuit C gate-by-gate. Typically a linear VSS scheme is used, so that each player may non-interactively compute addition and scalar-multiplication gates; on the other hand, multiplication gates require interaction, and in general are quite costly—more on this in a moment. (3) When the computation reaches the circuit's output gate, the players collectively possess a verifiable sharing of $f(x_1, \ldots, x_n)$; the *reveal phase* follows, in which they publicly reconstruct this value.

Again, the computation of multiplication gates is the most expensive part of an MPC protocol, and multiplication subroutines for $t < n/2$ typically require the broadcast channel. Using such a protocol in the compute phase incurs $\Omega(D)$ broadcast rounds, where D is the (multiplicative) depth of the circuit to be evaluated.

To address this issue, Beaver [Bea91a] introduced a technique for evaluating multiplication gates efficiently based on *multiplication triples*, vectors $(a, b, c) \in \mathbb{F}^3$ such that $ab = c$. Beaver showed that players who verifiably share a random multiplication triple (that is, with a and b uniformly random) can subsequently compute one multiplication gate at the low cost of *one VSS reconstruction phase.* A caveat is that to generate the triples requires the use of a pre-existing multiplication subprotocol; hence the primary savings comes from generating all triples *in parallel* in preprocessing.

Note, however, that in terms of broadcast rounds, this will not improve on $\Omega(D)$ *unless* the VSS reconstruction phase in question uses zero broadcasts—which is typically not the case. For instance, the two most commonly cited protocols are those of Cramer *et al.* [CDD+01] and Rabin and Ben-Or [RB89]. The former is a constant-round protocol which uses broadcast in reconstruction in an essential way, as part of "IC-signatures." The latter uses $\Omega(n)$ rounds in the sharing phase, and as described uses broadcast in the constant-round reconstruction phase. However, Baum *et al.* [BPW91] give a variant of the Rabin–Ben-Or VSS, which is the only example we know of a VSS which does *not* require broadcast for reconstruction. Using the [BPW91] VSS to generate multiplication triples and share inputs, yields an MPC protocol with $\Omega(n)$ rounds of broadcast.[2]

As we are unaware of any linear VSS which has both (1) constant number of broadcast rounds in the sharing phase, and (2) zero broadcasts in reconstruction, we believe the best previous broadcast complexity is $\Omega(\min\{n, D\})$. We remark that one drawback of our VSS and MPC protocols is that they are only proved secure for static adversaries. It would be interesting to extend these results to obtain a constant-broadcast, adaptively secure MPC.

For the sake of readability, only a proof sketch of the lower bound appears in the main body; the full proof can be found in the appendix.

2 Model, Definitions and Tools

We consider a complete, synchronous network of n players $P_1, \ldots, P_n$ who are pairwise connected by secure (private and authenticated) channels, and who additionally have access to a broadcast channel. Some of these players are corrupted by a centralized adversary $\mathcal{A}$ with *unbounded computing power.* The adversary is *active*, directing players under his control to deviate from the protocol in arbitrary ways. As noted, we consider only *static* rather than adaptive adversary in this work, meaning that he chooses which players to corrupt prior to the start of protocol execution. The computation evolves as a series of rounds. In a given round, honest players' messages depend only on information available to them from prior rounds; $\mathcal{A}$, however, is *rushing*, and receives all messages (and broadcasts) sent by honest players before deciding on the messages (and broadcasts) of corrupted players. Sometimes we refer to $\mathcal{A}$ thus defined as a *t-adversary.* We consider statistical security (since, as mentioned above, perfect security is unachievable in this setting), and let $\kappa = \mathsf{poly}(n)$ denote the error parameter, $\kappa \geq 2n$.

Information checking. An *information checking scheme (IC)* [RB89] is a triple of protocols (ICSetup, ICValidate, ICReveal) which achieves a limited signature-like functionality for three players: a *dealer* D, *intermediary* I, and *receiver* R. D holds as input a secret $s \in \mathbb{F}$, which he passes to I in ICSetup. ICValidate insures that even if D cheats, I knows a value which R will accept. In ICReveal, I sends s to R, together with some authenticating data, on the basis of which R accepts or rejects s as having originated from D. More formally, the scheme should satisfy the following guarantees:

CORRECTNESS: If D, I, and R are honest, then R will accept s in ICReveal.

NON-FORGERY: If D and R are honest, then R will reject any incorrect value $s^* \neq s$ passed to him in ICReveal, except with negligible probability.

COMMITMENT: If I and R are honest, then at the end of ICValidate I knows a value s such that R will accept s in ICReveal, except with negligible probability.

PRIVACY: If D and I are honest, then prior to ICReveal, a cheating R has no information on s.

We call an IC scheme *linear* if it meets the following additional condition.

[2] Actually, as written the protocol would still use $\Omega(n + D)$ broadcast rounds, since it uses the Rabin–Ben-Or information-checking protocol which requires WSS (hence broadcast) at each gate of the circuit—replacing this with the more efficient, linear IC protocol of [CDD+01] should suffice to achieve $\Omega(n)$.

Linearity: If D, I, and R have invoked ICSetup and ICValidate with respect to several secrets $\{s_i\}$, then I may (without further interaction) invoke ICReveal to authentically disclose any (public) linear combination of the s_i.

Our weak secret sharing and verifiable secret sharing protocols make use of a linear IC subprotocol based on that of [CDD+01], with some minor adjustments to increase broadcast efficiency. For completeness, the protocol and proof of security appear in Appendix A.

Weak secret sharing. An (n, t)-*weak secret sharing scheme (WSS)* is a pair of protocols (WSS-Share, WSS-Rec) for a set of players $\mathcal{P} = \{P_1, \ldots, P_n\}$, one of whom, the *dealer* D, holds input $s \in \mathbb{F}$. It must satisfy the following guarantees in the presence of an unbounded adversary corrupting up to t of the parties:

Weak Commitment: W.h.p., at the end of WSS-Share there exists a fixed value $s^* \in \mathbb{F} \cup \{\perp\}$, defined by the joint view of the honest parties, such that all honest parties will output the same value, either s^* or $\perp$, in WSS-Rec. If D is honest, then w.h.p. all honest parties will output $s^* = s$.

Privacy: If D is honest, then prior to WSS-Rec $\mathcal{A}$ gains no information on s (i.e., his view is statistically independent of s).

WSS is useful as an information-theoretic, distributed commitment for the dealer D. Thus we may say that a dealer who completes WSS-Share has *committed* to his (effective) input s, and that upon completing WSS-Rec he *decommits* (if a proper value is reconstructed). We call a commitment to a value in $\mathbb{F}$ a *proper commitment* (regardless of whether it equals the dealer's actual input), and a commitment to $\perp$ an *improper* (or *garbage*) *commitment*. We will also need a slightly relaxed version of WSS called *WSS-without-agreement* (or *very weak secret sharing* [BPW91]), in which the Commitment property above is replaced by

Weak Commitment Without Agreement: W.h.p., at the end of WSS-Share there exists a fixed value $s^* \in \mathbb{F} \cup \{\perp\}$, defined by the joint view of the honest parties, such that each honest party will output either s^* or $\perp$ in WSS-Rec (but some may output s^* and others $\perp$). If D is honest, then w.h.p. all honest parties will output $s^* = s$.

Furthermore, we will call a WSS(-without-agreement) *linear* if it satisfies the following in addition:

Linearity: If D has properly committed to several secrets $\{s^{(k)}\}$, then he may (without further interaction) invoke WSS-Rec to decommit to any (public) linear combination of the $s^{(k)}$. If some of the commitments are garbage, there still exists a fixed value $s^* \in \mathbb{F} \cup \perp$ which is reconstructed as the "linear combination" (w.h.p.).

We can slightly strengthen this requirement in the case of the sum of two values, to say that if one is properly committed and the other is garbage, their sum is garbage also (as opposed to any fixed value, which Linearity gives). We will use this property later on in the construction of VSS protocols.

Proper + Improper: If D has committed separately to $s \in \mathbb{F}$ and to $\perp$, then the reconstruction of the sum $s + \perp$ (or $\perp + s$) will yield $\perp$ (w.h.p.).

Our WSS(-without-agreement) protocol is presented in Section 3.2. It has a single sharing phase, which uses two broadcasts, and two different reconstruction phases: one which uses a single broadcast round and achieves ordinary WSS, and one which uses no broadcast but achieves only WSS-without-agreement.

Verifiable secret sharing. An (n, t)-*verifiable secret sharing scheme* [CGMA85] is a pair of protocols (VSS-Share, VSS-Rec) for a set of players $\mathcal{P} = \{P_1, \ldots, P_n\}$, one of whom, the *dealer* D, holds input $s \in \mathbb{F}$. In addition to the Privacy property above in the WSS case, VSS must satisfy the following, stronger guarantee in the presence of an unbounded adversary corrupting up to t of the parties:

Commitment: W.h.p., at the end of VSS-Share there exists a fixed value $s^* \in \mathbb{F}$, defined by the joint view of the honest parties, such that all honest parties will output s^* in VSS-Rec. If D is honest, then $s^* = s$. On the other hand, if D is dishonest, then the value of s^* is efficiently computable given the sharing-phase views (messages sent and received) of all corrupt parties. (Combined with Privacy, this latter property guarantees malicious dealers' inputs are independent of honest dealers' inputs when several copies of VSS are run in parallel.)

VSS strengthens WSS by guaranteeing that even when a cheating D does not cooperate in the Reconstruction phase, the honest players can still recover the value he committed to (which we now require to be a proper field element, not $\perp$). This makes possible a stronger variant of linearity, in which honest players can reconstruct linear combinations of secrets shared by *different dealers*. This strong linearity property is crucial for MPC applications of VSS.

We say that the parties *verifiably share* a secret s if each (honest) party maintains some state such that, when the honest parties invoke VSS-Rec on that joint state, they will reconstruct the value s (w.h.p.). Clearly, if a dealer D has just completed VSS-Share with effective input s, then the parties verifiably share s.

LINEARITY: If the parties verifiably share secrets $\{s^{(k)}\}$, then they also (without further interaction) verifiably share any (public) linear combination of the secrets.

Our VSS protocol, which uses 3 broadcast rounds in the sharing phase and none in reconstruction, appears in Section 3.2.

Pseudosignatures. *Pseudosignatures*, introduced by Pfitzmann and Waidner [PW96] as an extension of and improvement to a scheme by Chaum and Roijakkers [CR90], are an information-theoretic authentication technique for multiparty protocols. Suppose there is a setup phase during which the parties enjoy access to a physical broadcast channel (but need not know their future inputs). The parties may use such a setup phase to implement pseudosignatures; then, using the pseudosignatures for authentication, they may simulate future invocations of broadcast by running an authenticated Byzantine agreement protocol (e.g., [DS83]), thus avoiding any need for a physical broadcast channel during the main phase of the protocol.

One does pay a price in removing cryptographic assumptions—in the case of pseudosignatures, the price is *limited transferability.* The integrity of a pseudosignature "degrades" each time it is passed from one party to another, so that it only remains valid for an *a priori* bounded number of transfers[3].

At a high level, the pseudosignatures of [PW96] rely on a subprotocol implementing a *secure, many-to-one anonymous channel* [Cha88] to any single player P^* such that the following conditions are satisfied in the presence of a t-adversary $\mathcal{A}$:

ANONYMITY: Even if P^* is corrupt, $\mathcal{A}$ learns no more than the multi-set of messages sent by the honest players (and in particular gains no information on which messages came from which honest players).

PRIVACY: If P^* is honest, then $\mathcal{A}$ learns no information on messages sent by honest players.

$(1-\epsilon)$-RELIABILITY: Let $\{x_1, \ldots, x_k\}$ be the set of honest senders' messages, which we assume for simplicity are all unique.[4] If P^* is honest, then with probability $\geq 1-\epsilon$, his output will be an ordered list Y, which includes each value x_i.

NON-MALLEABILITY: If P^* is honest and honest messages $\{x_1, \ldots, x_k\}$ are drawn uniformly at random, then the distribution of the output list Y is statistically indistinguishable from a copy of Y in which honest messages are removed, k independently sampled values are added, and the list is re-sorted. (Note this does not follow immediately from Privacy!)

Privacy and anonymity may hold either perfectly, or statistically (in which case a statistically negligible amount of information may be leaked). We (and they) additionally require these security properties to hold under parallel composition of the anonymous channel.

Assuming such a channel, the pseudosignature scheme is roughly as follows, for a one-time signature (see [PW96] for formal details): Each player chooses a large number of random keys, which will function as information-theoretic message authentication codes. The players invoke the anonymous channel many times in parallel, each sending one key per invocation, privately and anonymously, to the signer P^*. Hence for each invocation of the channel, P^* receives an associated *signature block* containing $n-1$ anonymous keys. In order to (pseudo)sign a message M, P^* simply signs it (i.e., computes the message authentication code on the message) using every authentication key, in every block—these individual signatures are referred to as *minisignatures*, and the collection of all of them, arranged in blocks, is P^*'s *pseudosignature* on M.

[3] Another price is finite, fixed-in-advance player set!

[4] If not, have players append random tags to their messages, which P^* strips off upon receipt—this decreases reliability by at most a negligible amount (the probability two honest tags collide).

Verification is more involved. The *first* verifier V_1 (i.e., the player to whom P^* originally sends M with pseudosignature) accepts the signature provided that, in *every* signature block, one of the minisignatures matches the key V_1 sent for that block. The second verifier V_2 accepts the signature provided that, in *most* of the signature blocks, one of the minisignatures matches the key V_2 sent for that block. The third verifier accepts the signature provided *a fair number* of minisignatures check out, and so on, where each new verifier has a lower acceptance threshold than the previous one.

The rationale for the increasingly tolerant verifiers is the fear that a cheating signer P^*, even though he does not know whose keys are whose in any given block, could (for example) correctly sign M with every key except for half the keys in a certain block. There is a good chance then that V_1 will find a correct minisignature in every block, but will pass it on to V_2 who will *not* find a correct minisignature in the half-signed block. If V_1 accepts the signature while V_2 rejects, P^* has successfully broken the signature scheme.

The anonymous channel in [PW96] is implemented using algebraic techniques; it relies on a broadcast channel and is not robust to malicious behavior. Rather, the protocol uses fault detection and localization: any failed invocation leads to public identification of either a single corrupt player, or a pair of players at least one of whom is corrupt. Even with honest majority, there are $\Omega(n^2)$ pairs of players with one of them corrupt; therefore the adversary can force $\Omega(n^2)$ rounds where broadcast is used. In contrast, in Section 3.1 we show how to implement the anonymous channel in a robust fashion, using only a *constant* number (namely, 3) of broadcast rounds, which is what our VSS protocol requires. On the other hand, we should note that the [PW96] construction of pseudosignatures works for any $t < n$, while ours is limited to $t < n/2$ (which is not any *additional* limitation when the purpose of the pseudosignatures is to simulate broadcast in an honest-majority MPC protocol).

We are now ready to describe our construction. For the sake of simplicity, we adopt in all our protocols the usual convention that whenever a party fails to send an expected message, or sends a syntactically incorrect message, it is replaced with some default message. Thus we do not deal separately with the case of missing or improper messages.

3 Honest-Majority MPC in Three Broadcast Rounds

As discussed in Section 1, in order to achieve MPC with no further broadcasts, it suffices to implement a broadcast-efficient anonymous channel (secure under parallel composition). If we can do this using only a constant number of broadcast rounds, then we are done.

Our main idea is to instantiate the anonymous channel using a special-purpose MPC protocol. Each player's input to the computation will be his randomly chosen authentication key, and the output—revealed only to the designated signer!—will be a random permutation of the inputs. The players collectively sample a permutation π at random by the method of "throwing darts" (e.g., [Hag91]). To wit, each player chooses a random index in a long vector, and that player's message appears at that location. (Collisions, of course, may occur.) Provided the players can correctly simulate the dart-throwing in such a way that each player's choice of index remains private from the receiver, and so that the values of the messages are revealed only to the receiver, the resulting permutation is clearly uniform and so the channel will be anonymous.

3.1 A new pseudosignature construction

With ingredients in hand, we now present a protocol, AnonChan, which implements an anonymous channel for $t < n/2$ using only black-box access to a linear VSS protocol. Although the VSS itself must rely on physical broadcast when $t \geq n/3$, our construction uses no additional broadcast rounds beyond those required by the calls to VSS. In fact, if the VSS uses B_s broadcast rounds in the sharing phase and B_r broadcast rounds in the reconstruction phase, then our protocol uses only $B_s + 3B_r$ broadcast rounds. When VSS is implemented with the $(3, 0)$-broadcast VSS protocol of Section 3.2, AnonChan therefore uses a total of 3 broadcast rounds.

At a high level, the protocol follows the "throwing darts" paradigm described above. Each player P_i commits, by invoking the VSS sharing phase in many parallel executions, to a sufficiently long and sparse vector $\mathbf{v}^{(i)}$ in which the few nonzero entries are set equal to P_i's input. If all players do so correctly, then the sum of the vectors will have relatively few collisions among nonzero entries—in particular, it will preserve at least one instance of each input value with high probability. Then the players use the linearity of the VSS scheme to locally compute shares of the sum of the $\mathbf{v}^{(i)}$'s, and send the resulting shares to the receiver P^* (privately). P^* then simulates the VSS reconstruction function internally to recover the sum, which contains at least one copy of each player's input (along with some additional garbage, which we can tolerate).

Of course, dishonest players may not prepare their $\mathbf{v}^{(i)}$'s properly—if a cheater were to commit to (say) a vector full of random entries, then including it in the sum would destroy all information about honest players' inputs. Therefore we require that, after committing to his vector, each player must *prove* that it is indeed sparse, and that nonzero entries are equal. A simple cut-and-choose proof suffices: The prover P_i prepares many additional vectors $\mathbf{w}_j^{(i)}$, $1 \leq j \leq \kappa$, in the same manner as the original $\mathbf{v}^{(i)}$, except that he rerandomizes the locations of nonzero entries. He also commits, for each $\mathbf{w}_j^{(i)}$, to both the list of locations of nonzero entries, and to a permutation on vector entries which sends nonzero locations of $\mathbf{v}^{(i)}$ to nonzero locations of $\mathbf{w}_j^{(i)}$.

The parties then jointly generate a random challenge r: For each $\mathbf{w}_j^{(i)}$, P_i is challenged to reveal either (a) the permutation, in which case the parties reconstruct the difference of the permuted $\mathbf{v}^{(i)}$ and $\mathbf{w}_j^{(i)}$, to verify it is the zero vector; or (b) the nonzero entries of $\mathbf{w}_j^{(i)}$, in which case the parties reconstruct the alleged *zero* entries to verify they are all zero, and reconstruct the *differences* of nonzero entries, which should be zero to verify that all nonzero entries are equal. Vectors which fail this check are excluded, and with overwhelming probability all accepted vectors are indeed sparse, with nonzero entries equal.

The protocol parameters are the player set $\mathcal{P} = \{P_1, \ldots, P_n\}$, a designated receiver $P^* \in \mathcal{P}$, and error parameter κ. All computations are done over a finite field $\mathbb{F} = GF(2^\kappa)$, where we require $|\mathbb{F}| > n$. Protocol AnonChan is shown in Figure 1.

Theorem 1. *Let* VSS *be a verifiable secret sharing protocol which is either perfectly or statistically private and* $(1 - 2^{-\Omega(\kappa)})$*-reliable in the presence of an unbounded adversary corrupting up to* $t < n/2$ *parties. Then protocol* AnonChan *implements a statistically anonymous, statistically private,* $(1-2^{-\Omega(\kappa)})$*-reliable many-to-one channel.*

Proof. Anonymity. Without loss of generality consider an adversary corrupting P^*. By the perfect (statistical) privacy of VSS, $\mathcal{A}$'s view of the VSS sharing phases at the end of step 1 reveals zero (negligible) information on the honest players' inputs. The random value r, reconstructed in step 2, is independent of the honest players' inputs.

With probability $\geq 1 - 2^{-\Omega(\kappa)}$, no honest player is disqualified in any instance of VSS-Share. Moreover, each honest player P_i will (with high probability) have correctly shared sparse vectors $\mathbf{v}^{(i)}, \mathbf{w}_j^{(i)}$. Consequently, in step 3 $\mathcal{A}$ can predict beforehand (with high probability of correctness), the outcome of all reconstructions associated with honest players' vectors (namely, they will be zero), and so $\mathcal{A}$ learns negligible information on honest inputs. Also, w.h.p. all honest players will be included in PASS.

In step 4, P^* receives shares of $\mathbf{v} = \sum_{P_i \in \mathsf{PASS}} \mathbf{v}^{(i)}$. Because $\mathcal{A}$ knows the value of $\mathbf{v}^{(j)}$ for each dishonest player P_j in PASS, he may subtract these from $\mathbf{v}$ and obtain $\mathbf{v}_{\text{honest}} = \sum_{P_i \text{ honest}} \mathbf{v}^{(i)}$. Which is to say, $\mathcal{A}$ learns no more information from $\mathbf{v}$ than he would from learning $\mathbf{v}_{\text{honest}}$ directly. But since each honest P_i chose his 4κ indices of $\mathbf{v}^{(i)}$ independently at random, the distribution of $\mathbf{v}_{\text{honest}}$ is a function only of the *set* of honest players' (unique) values. Hence $\mathcal{A}$ gains no information from seeing $\mathbf{v}_{\text{honest}}$ other than what he could learn from seeing the set of honest players' values.

Privacy. Assume P^* is honest. As above, by the privacy of VSS, $\mathcal{A}$ learns at most a negligible amount about honest players' inputs from the VSS sharing phases in step 1; no information from the reconstruction of r in step 2, which is independent of the x_i's; and a negligible amount from the step 3 proof of sparseness, the outcome of which he can predict with all but negligible probability. Then statistical privacy is ensured since all communication in step 4 is on the private channels to P^*.

$(1 - 2^{-\Omega(\kappa)})$-Reliability. We must show that the list output by an honest P^* contains a copy of each honest input x_i, with overwhelming probability. As noted in the proof of Anonymity, each honest player is included in PASS with high probability.

Call a vector *improper* if it is not 4κ-sparse, or if it has unequal nonzero entries. We claim that if a dishonest P_i shares in step 1 an improper vector $\mathbf{v}^{(i)}$, he will be disqualified with probability $\geq 1-2^{-\Omega(\kappa)}$.

Protocol $\mathsf{AnonChan}(\mathcal{P}, P^*, \kappa)$

Input: Each $P_i \in \mathcal{P} \setminus \{P^*\}$ has as input a message $x_i \in \mathbb{F} \setminus \{0\}$. For simplicity assume all x_i are unique (see Footnote 2).
Output: P^* outputs a list $Y = \{y_1, \ldots, y_{n'}\}$ ($n' = O(n)$) which (with overwhelming probability) contains x_i for every honest P_i (but may contain other values). Moreover, an adversary corrupting up to $t < n/2$ players gains no information regarding which honest players sent which messages.

1. Set $\ell = 96\kappa n$. P_i constructs a random 4κ-sparse vector $\mathbf{v}^{(i)} \in \mathbb{F}^\ell$ whose 4κ nonzero entries are x_i.
   At the same time, P_i constructs κ additional random, 4κ-sparse vectors $\mathbf{w}_1^{(i)}, \ldots, \mathbf{w}_\kappa^{(i)} \in \mathbb{F}^\ell$. Each $\mathbf{w}_j^{(i)}$, like $\mathbf{v}^{(i)}$, has x_i at its nonzero entries.
   For each $\mathbf{w}_j^{(i)}$, P_i chooses a random permutation $\pi_j^{(i)} : [\ell] \to [\ell]$ such that $\mathbf{v}^{(i)}[k] = \mathbf{w}_j^{(i)}[\pi_j^{(i)}(k)]$ for all $k \in [\ell]$. (I.e., $\pi_j^{(i)}$ is random subject to the condition that it send nonzero indices of $\mathbf{v}^{(i)}$ to nonzero indices of $\mathbf{w}_j^{(i)}$.)
   P_i invokes $\mathsf{VSS\text{-}Share}$ $O(\kappa^2 n)$ times in parallel to verifiably share each of the following values:
   - Each coordinate of $\mathbf{v}^{(i)}$ and of the $\mathbf{w}_j^{(i)}$'s;
   - each of the permutations $\pi_j^{(i)}$;
   - for each $\mathbf{w}_j^{(i)}$, its list of nonzero indices;
   - a random element $r^{(i)} \in \mathbb{F}$.

   Any player disqualified during some $\mathsf{VSS\text{-}Share}$ is disqualified here.
2. The players invoke $\mathsf{VSS\text{-}Rec}$ to reconstruct the sum $r := \sum r^{(i)}$, and interpret it as a random bit-string of length κ.
3. For each bit $b_j \in r$ (in parallel):

   $b_j = 0$:
   1. The players invoke n instances of $\mathsf{VSS\text{-}Rec}$ to reconstruct the permutations $\pi_j^{(i)}$, for all P_i. (If the result is not a valid permutation, P_i is disqualified.)
   2. For each $\mathbf{w}_j^{(i)}$, the players invoke ℓ instances of $\mathsf{VSS\text{-}Rec}$ to reconstruct the vector $\mathbf{u} := \pi_j^{(i)}(\mathbf{v}^{(i)}) - \mathbf{w}_j^{(i)}$. If $\mathbf{u} \neq 0 \in \mathbb{F}^\ell$, P_i is disqualified. ($\pi_j^{(i)}$ acts on a vector by permuting its components.)

   $b_j = 1$:
   1. The players invoke n instances of $\mathsf{VSS\text{-}Rec}$ to reconstruct the list of nonzero indices for $\mathbf{w}_j^{(i)}$, for all P_i. (If the result is not a valid list of 4κ distinct indices in $[\ell]$, P_i is disqualified.)
   2. For each $\mathbf{w}_j^{(i)}$, invoke $\ell - 4\kappa$ instances of $\mathsf{VSS\text{-}Rec}$ to reconstruct the values at (alleged) zero-indices of $\mathbf{w}_j^{(i)}$. If any are actually nonzero, P_i is disqualified. Also invoke $4\kappa - 1$ instances of $\mathsf{VSS\text{-}Rec}$ to reconstruct consecutive *differences* of (alleged) nonzero entries of $\mathbf{w}_j^{(i)}$. If any such differences are nonzero, P_i is disqualified.
4. Let PASS denote the set of players who are not disqualified. Each player locally computes his shares of the vector $\mathbf{v} := \sum_{P_i \in \mathsf{PASS}} \mathbf{v}^{(i)}$. He sends the shares, over private channel, to the receiver P^*.
   P^* uses the received shares to internally simulate $\mathsf{VSS\text{-}Rec}$ and recover the vector $\mathbf{v}$. He outputs the list Y of nonzero entries which appear $\geq 3\kappa$ times in $\mathbf{v}$.

Figure 1: *A protocol implementing secure anonymous channel using black-box VSS.*

Note P_i must commit to all $\mathbf{w}_j^{(i)}$'s in step 1, before $\mathcal{A}$ has any information on the value of the challenge r, which will be uniformly distributed over $GF(2^\kappa)$.

For each j: If $\mathbf{w}_j^{(i)}$ is proper, then no permutation will map the entries of an improper $\mathbf{v}^{(i)}$ onto matching entries of $\mathbf{w}_j^{(i)}$; hence the vector $\mathbf{u} := \pi_j^{(i)}(\mathbf{v}^{(i)}) - \mathbf{w}_j^{(i)}$ will be nonzero and P_i will be disqualified w.h.p. provided $b_j = 0$. Alternatively, if $\mathbf{w}_j^{(i)}$ is improper, then either the alleged zero entries will actually contain some nonzero value, or the alleged *non*zero entries will contain unequal values (or both). In either case, opening the alleged zero entries, and the differences of alleged nonzero entries, will reveal an error and P_i will be disqualified w.h.p. provided $b_j = 1$.[5]

[5]The reason P_i will only be disqualified "with high probability" is that, with negligible probability, the adversary may succeed in causing incorrect values to be reconstructed for one or more of the VSS sharings.

Since each b_j is uniformly random, P_i will be disqualified with probability $\geq 1 - 2^{-\Omega(\kappa)}$, and with probability $\geq 1 - t \cdot 2^{-\Omega(\kappa)} = 1 - 2^{-\Omega(\kappa)}$, *every* dishonest P_i who commits to an improper $\mathbf{v}^{(i)}$ will be disqualified.

Condition on the event that $\mathbf{v}^{(i)}$ is proper for all $P_i \in \mathsf{PASS}$. W.h.p. P^* will correctly reconstruct the sum $\mathbf{v} = \sum_{P_i \in \mathsf{PASS}} \mathbf{v}^{(i)}$, and having done so he takes as output Y the list of values in $\mathbf{v}$ which appear $\geq 3\kappa$ times. Fix an honest P_i, and we must show that w.h.p. $\mathbf{v}$ contains x_i at least 3κ times.

Now there are at most $4\kappa(n-1) \leq 4\kappa n$ nonzero indices associated with all the other proper vectors. Assume exactly $4\kappa n$ such indices (for simplicity). The intersection of P_i's choice of 4κ indices, and these fixed indices, follows a hypergeometric distribution: There is an urn consisting of $\ell = 96\kappa n$ balls, and $4\kappa n$ of which are red (corresponding to the fixed indices). We will draw 4κ balls without replacement, and we wish to bound the probability that the number of red balls drawn is $\geq \kappa$.

Let X denote the number of red balls, then $\mathbb{E}[X] = 4\kappa \cdot 4\kappa n/\ell = \kappa/6$, and we may invoke the (weaker) tail bound on the hypergeometric distribution appearing in [Chv79] to conclude that, for any $C \geq 0$:

$$\Pr[X \geq \mathbb{E}[X] + C \cdot 4\kappa] \leq \exp(-C^2 \cdot 4\kappa).$$

In particular, we may take (say) $C = 1/6$ to obtain

$$\Pr[X \geq \frac{5}{6}\kappa] \leq \exp(-\frac{1}{9}\kappa) = 2^{-\Omega(\kappa)}.$$

Given that $X \leq \frac{5}{6}\kappa$, then in particular $X \leq \kappa$ and at least 3κ indices at which P_i placed x_i are untouched, so P^* receives x_i correctly.

Non-Malleability. By independence of inputs for the VSS, all the values which dishonest players commit to in step 1 are independent of those which honest players commit to in step 1.

Consider the output list $Y = \{y_1, \ldots, y_{n'}\}$ of an honest receiver P^*, and condition on the (high-probability) events that all honest players are included in PASS and all cheaters who shared improper vectors are excluded. The list Y consists of all nonzero entries of $\mathbf{v}$ which appear $\geq 3\kappa$ times. Write $\mathbf{v} := \mathbf{v}_{\text{honest}} + \mathbf{v}_{\text{dishonest}}$, where

$$\mathbf{v}_{\text{honest}} := \sum_{P_i \text{ honest}} \mathbf{v}^{(i)}, \qquad \text{and} \qquad \mathbf{v}_{\text{dishonest}} = \sum_{\substack{P_i \text{ dishonest} \\ P_i \in \mathsf{PASS}}} \mathbf{v}^{(i)}$$

The first observation we make is that $\mathbf{v}_{\text{honest}}$ consists of $\leq 4\kappa n$ nonzero entries, each of which has marginal distribution uniformly random in $\mathbb{F} \setminus \{0\}$ Likewise there are at most $4\kappa t \leq 4\kappa n$ nonzero entries of $\mathbf{v}_{\text{dishonest}}$ (though they may not be random at all). It follows by union bound that the probability of any intersection between the set of nonzero values in $\mathbf{v}_{\text{honest}}$, and those in $\mathbf{v}_{\text{dishonest}}$, is negligible. Therefore we condition on the event that no such intersection exists.

In the same vein, consider a pair of nonzero entries of $\mathbf{v}$, say $\mathbf{v}[j], \mathbf{v}[k]$, $j \neq k$, and suppose $\mathbf{v}[j] = \mathbf{v}[k]$. By definition of $\mathbf{v}$, this means that

$$\begin{aligned} \mathbf{v}_{\text{honest}}[j] + \mathbf{v}_{\text{dishonest}}[j] &= \mathbf{v}_{\text{honest}}[k] + \mathbf{v}_{\text{dishonest}}[k] \\ \implies \mathbf{v}_{\text{honest}}[j] - \mathbf{v}_{\text{honest}}[k] &= \mathbf{v}_{\text{dishonest}}[k] - \mathbf{v}_{\text{dishonest}}[j]. \end{aligned}$$

Now if $\mathbf{v}_{\text{honest}}[j] \neq \mathbf{v}_{\text{honest}}[k]$, then its value has marginal distribution uniform over $\mathbb{F} \setminus \{0\}$. Since there are $\leq 4\kappa n$ nonzero entries in $\mathbf{v}_{\text{honest}}$ (resp., $\mathbf{v}_{\text{dishonest}}$), there are $\leq (4\kappa n)^2$ *pairs* of nonzero values from $\mathbf{v}_{\text{honest}}$ (resp., $\mathbf{v}_{\text{dishonest}}$), and by a union bound the probability that any nonzero difference $\mathbf{v}_{\text{honest}}[j] - \mathbf{v}_{\text{honest}}[k]$ equals $\mathbf{v}_{\text{dishonest}}[j] - \mathbf{v}_{\text{dishonest}}[k]$, is negligible. Condition on the event that no such equality holds.

It follows that $\mathbf{v}[j] = \mathbf{v}[k]$ *only holds* when $\mathbf{v}_{\text{honest}}[j] = \mathbf{v}_{\text{honest}}[k]$. In this case we of course have $\mathbf{v}_{\text{dishonest}}[j] = \mathbf{v}_{\text{dishonest}}[k]$ as well. The following two claims will come in handy.

Claim 2. *Let $S_{\text{honest}} = \{x_1, \ldots, x_{n-t}\}$ denote the set of honest players' messages. With overwhelming probability, no two distinct, nonempty subsets $I, J \subseteq S_{\text{honest}}$ have the same sum.*

Proof. By canceling common terms, we may assume I and J are disjoint. In fact since we are working in a field of characteristic 2, the condition $\sum_{i \in I} x_i = \sum_{j \in J} x_j$ for disjoint $I, J \subseteq [n-t]$ is equivalent to

$\sum_{i \in I \cup J} x_i = 0$, so it suffices to show that with high probability there is no subset of S_{honest} which sums to zero. For any fixed non-empty subset, the probability that the subset sums to zero is $1/|\mathbb{F}| = 2^{-\kappa}$ (here the probability is over the uniformly random choices of the x_i in that subset). But the number of such subsets is $2^{n-t} \leq 2^n \leq 2^{\kappa/2}$ (recall we assume $\kappa \geq 2n$). Hence by a union bound the probability that there exists any subset summing to zero is $\leq 2^{\kappa/2} \cdot 2^{-\kappa} = 2^{-\kappa/2} = 2^{-\Omega(\kappa)}$, which completes the proof of the claim. □

Claim 3. *With overwhelming probability, any nonzero $x \in \mathbb{F}$ which appears $\geq 2\kappa$ times in $\mathbf{v}_{\text{honest}}$ is actually some honest message, i.e., $x \in S_{\text{honest}}$.*

Proof. Condition on the event that no two nonempty subsets of S_{honest} have the same sum, which occurs w.h.p. by the previous claim. Then $x \notin S_{\text{honest}}$ will only appear $\geq 2\kappa$ times in $\mathbf{v}_{\text{honest}}$ if some fixed subset of honest players have overlap of size $\geq 2\kappa$ on the indices they choose. But by the proof of reliability, each individual honest player chooses $\geq 3\kappa$ entirely unique indices (w.h.p.), leaving $\leq \kappa$ possible overlapping indices. □

Now we return to consider a set of $\geq 3\kappa$ indices in $\mathbf{v}$ which are equal (and hence determine an element of the output list Y). By the argument above, we may condition on the event that in fact the corresponding entries of $\mathbf{v}_{\text{honest}}$ are all equal. If the latter are equal to *zero*, then the resulting $y \in Y$ is independent of $\mathbf{v}_{\text{honest}}$ and, in particular, of all honest players' messages. Alternatively, if the $\mathbf{v}_{\text{honest}}$ entries are nonzero, then by Claim 3 they are equal to some honest message $x \in S_{\text{honest}}$. Since all the corresponding entries of $\mathbf{v}_{\text{dishonest}}$ must be equal as well, we need only bound the probability that nonzero entries of $\mathbf{v}_{\text{dishonest}}$ have overlap $\geq 3\kappa$ with some honest message. But the proof of reliability guarantees that even an overlap of $\geq \kappa$ is of negligible probability.

Hence any occurrence of $\geq 3\kappa$ equal entries in $\mathbf{v}$ either corresponds to an honest message, or to an output which is independent of all honest messages, which proves the Non-Malleability property.

□

3.2 A broadcast-efficient VSS protocol

In this section we present our new $(3,0)$-broadcast VSS protocol for $t < n/2$. Its overall round complexity is $(9,1)$. This is the first linear VSS protocol enjoying such a small number of broadcast rounds without trusted setup. To our knowledge, the most broadcast-efficient unconditional VSS protocol in the literature, due to Kumaresan *et al.* [KPC10], has broadcast complexity $(2,2)$, with $(3,2)$ rounds overall. However, the authors do not claim it to be linear, and it has *exponential* communication complexity. The same authors give a $(3,2)$-broadcast, $(4,2)$-round protocol which is polynomial-time and linear (we believe—though the authors do not claim it here either), at the expense of an additional round in the sharing phase.

Considering protocols which use zero broadcasts during reconstruction—which are more suitable for broadcast-efficient MPC (see next section)—we are aware only of Baum-Waidner *et al.* [BPW91]. They propose a modification of the Rabin and Ben-Or VSS [RB89] which eliminates the latter's use of broadcast in the reconstruction phase; unfortunately, both these protocols' sharing phases require $\Theta(n)$ broadcast rounds.

Our VSS protocol is also inspired by [RB89], but leverages a number of optimizations to reduce the overall round and broadcast complexity to constant. Its sharing phase uses a WSS protocol, which we now describe.

Weak secret sharing protocol. Our WSS(-without-agreement) protocol uses two broadcasts in its sharing phase, and admits two different reconstruction phases: one which uses a single broadcast round and achieves ordinary WSS, and one which uses no broadcast but achieves only WSS-without-agreement. In turn, the protocol makes use of a linear IC subprotocol based on that in [CDD+01] (Appendix A). The WSS protocol(s) is shown in Figure 2. Since its sharing and reconstruction phases are invoked at different rounds of the VSS protocol's sharing phase, we specify them as separate protocols for convenience.

Theorem 4. $\mathsf{WSS} = (\mathsf{WSS\text{-}Share}, \mathsf{WSS\text{-}Rec})$ *is a linear weak secret sharing scheme secure against a static, unbounded adversary corrupting $t < n/2$ players. Furthermore,* $\mathsf{WSS}^* = (\mathsf{WSS\text{-}Share}, \mathsf{WSS\text{-}Rec\text{-}NoBC})$

Protocol WSS-Share($\mathcal{P}, D, s$)

1. D chooses a random polynomial $f(x)$ of degree $\leq t$ such that $f(0) = s$, and sets $s_i := f(i)$; this will be P_i's share. For each pair $P_i, P_j \in \mathcal{P} - \{D\}$, run ICSetup($D, P_i, P_j, s_i$).

2–5. **2 x BROADCAST in 4,5:** For each $P_i, P_j \in \mathcal{P} - \{D\}$, run ICValidate($D, P_i, P_j, s_i$).

Protocol WSS-Rec($\mathcal{P}, D, s$)

1. For each pair $P_i, P_j \in \mathcal{P} - \{D\}$, run ICReveal($P_i, P_j, s_i$).

2. **BROADCAST:** D broadcasts the polynomial $f(x)$ which he used to share the secret. P_i broadcasts the list of pieces $\{(j, s_j)\}$ which he accepted in ICReveal in the previous step.
Let HAPPY denote the set of players who accept at least $n - t$ pieces, and all of whose accepted pieces lie on the polynomial $f(x)$. If $|\text{HAPPY}| \geq n - t$, all players take $s = f(0)$ to be the secret, otherwise $\perp$.

Protocol WSS-Rec-NoBC($\mathcal{P}, D, s$)

1. For each pair $P_i, P_j \in \mathcal{P} - \{D\}$, run ICReveal($P_i, P_j, s_i$). If P_i accepts at least $n - t$ pieces, and all accepted pieces lie on a polynomial $f(x)$ of degree $\leq t$, then P_i takes $s = f(0)$ to be the secret, otherwise $\perp$.

Figure 2: *The WSS protocol, with its two different reconstruction phases.*

is a linear WSS-without-agreement scheme, secure against a static, unbounded adversary who corrupts $t < n/2$ players.

Proof. Commitment. First consider a cheating D. At the end of WSS-Share, an honest P_i holds s_i which all honest parties will accept (due to the Commitment property of the IC protocol). Now these pieces s_i held by the honest parties define a polynomial $f^*(x)$; if $\deg f^*(x) > t$, then each honest party will accept pieces not lying on the dealer's broadcast polynomial $f(x)$. Therefore we will have $|\text{HAPPY}| < n - t$, and $\perp$ will be reconstructed. Note that this situation is precisely a garbage commitment.

Otherwise $\deg f^*(x) \leq t$ (and the commitment is proper). If the dealer's broadcast polynomial $f(x) \neq f^*(x)$ then again each honest party will accept pieces not on $f(x)$, and so $\perp$ will be reconstructed. If $f^*(x) = f(x)$ then it may be the case that $\perp$ is reconstructed (depending on the values honest parties accept from dishonest parties), or that $s^* = f(0) = f^*(0)$ is reconstructed. Regardless, there is only one non-$\perp$ value which may be reconstructed, and it is fixed by the joint view of the honest parties at the end of WSS-Share.

Now if D is honest, then by the IC Non-Forgery property no cheating party can fool an honest party into accepting a value other than s_i during ICReveal (except with negligible probability). It follows that each honest player will accept $\geq n - t$ pieces, and all their accepted values will lie on the dealer's polynomial $f(x)$. Thus $|\text{HAPPY}| \geq n - t$ and the parties output $s = f(0)$.

Privacy. If D is honest, then by the IC Privacy property, the adversary has no information on any s_i value held by an honest player P_i prior to ICReveal. Hence the adversary learns only the t points on the polynomial $f(x)$ corresponding to dishonest players' shares, and in particular has no information on $f(0) = s$ prior to WSS-Rec.

Commitment Without Agreement. Define $f^*(x)$ as above, by the shares of the honest parties. As before if $\deg f^*(x) > t$, all honest parties will accept a set of pieces which do not lie on any degree t polynomial, and they will all output $\perp$.

If $\deg f^*(x) \leq t$, then honest party P_i will output $s^* = f^*(0)$ only if all the pieces he accepts from dishonest parties lie on $f^*(x)$; otherwise the set of pieces he accepts will lie on no polynomial of degree t, and he will output $\perp$.

For an honest D, the argument is the same as in the with-agreement case: Due to IC Non-Forgery, all honest parties will (w.h.p.) accept only values which lie on $f(x)$, and so all will output the correct value $s = f(0)$.

Linearity. Suppose D has properly committed to values $\{s^{(k)}\}$, using polynomials $f_k(x)$. Then for each value $s^{(k)}$, player P_i holds a share $s_i^{(k)}$. To decommit to a linear combination of the $s^{(k)}$, in WSS-Rec P_i reveals the linear combination of his $s_i^{(k)}$ during ICReveal (in place of "s_i"), and D broadcasts the linear combination of these polynomials (in place of "$f(x)$"). Then the properties of commitment and privacy remain in place, since taking a linear combination of polynomials of degree at most t results in a new polynomial of degree at most t.

If some of the commitments were garbage, this means exactly that some of the polynomials (defined by the shares of the honest players) were of degree $> t$. Nevertheless, taking a linear combination of these polynomials results in a single, fixed polynomial whose free term is the only possible non-$\perp$ value which honest parties will reconstruct (and then only if the new polynomial has degree $\leq t$).

PROPER + IMPROPER. A proper commitment is associated with a polynomial of degree $\leq t$, and an improper commitment with one of degree $> t$. Thus the sum of the two has degree $> t$, corresponds to another improper commitment, and will yield $\perp$ (w.h.p.). □

The broadcast-efficient VSS protocol. Here we present our new VSS protocol, which requires three broadcast rounds, and uses the WSS protocol from the last section in its sharing phase. At a high level, the protocol is inspired by that of Rabin and Ben-Or [RB89], and has a similar structure. First D distributes shares of a t-degree polynomial f where $s := f(0)$ and of additional random t-degree polynomials g_k. Each player P_i commits to all shares via WSS. Then the parties jointly carry out a cut-and-choose process in which the players are challenged to reconstruct either g_k or $f + g_k$ for each k, which must be degree t. Players who complain of incompatible shares, or fail to participate, have their shares broadcast (and hence fixed) by D.

As mentioned earlier, Rabin and Ben-Or's VSS requires $\Theta(n)$ broadcast rounds in the share phase. The main novelty which allows us to reduce total round complexity to $O(1)$ is that we require the *dealer* as well as the players to commit via WSS to the shares he distributed. As a consequence, we are able to deal with cut-and-choose challenges efficiently in parallel, rather than sequentially: The challenges are broadcast in step 7. In step 8, the parties respond to the challenges by using WSS-without-agreement to reconstruct the shares of the appropriate polynomials. In the final step 9, a broadcast is used to confirm the results of the WSS-without-agreement; at the same time D has a chance (and is obligated) to broadcast shares of players for whom he did not reconstruct the correct share in step 8.

An additional trick which saves us a broadcast round can be seen in step 6, which is inserted between the last two rounds of the WSS share phase. In this step, the parties perform a *pre-broadcast* by sending each other player their intended WSS final-round broadcast on point-to-point channels. In step 7, they officially complete WSS by echoing the pre-broadcast. This forces a cheating player to "semi-commit" in step 6 to one of at most $n - t$ possible final-round broadcasts for WSS, since a majority (including at least one honest player) must confirm his pre-broadcast. Luckily, semi-commitment restricts cheaters' options enough that players are able to broadcast the cut-and-choose challenges in the *same round*—step 7—rather than waiting for full commitment and then using another broadcast. (Note that in the case of a non-rushing adversary, step 6 is unnecessary.)

Protocol VSS-Share$(\mathcal{P}, D, s)$

1. D chooses a random polynomial $f(x)$ of degree $\leq t$ such that $f(0) = s$, and sets $s_i := f(i)$. Also for $1 \leq k \leq \kappa n$, D chooses random polynomials $g_k(x)$ of degree $\leq t$, and sets $t_{ki} := g_k(i)$. D sends $(s_i, \{t_{ki}\}_k)$ to P_i.

2–5. **BROADCAST:** P_i and D will now each act as WSS dealers to commit to P_i's share s_i. We reserve s_i to denote the value D commits to, and let s_i^* denote that which P_i commits to (these may be different if D and/or P_i is dishonest). D and P_i act as dealer in steps 1–4 of WSS-Share(D, s_i), WSS-Share(P_i, s_i^*), WSS-Share(D, t_{ki}), and WSS-Share(P_i, t_{ki}^*) $(1 \leq k \leq \kappa n)$.

6. The parties have just completed WSS-Share step 4/ICValidate step 3. In the next step (corresponding to WSS-Share step 5/ICValidate step 4) the WSS/IC dealer will resolve conflicts. Instead of doing so immediately, let BC_i denote the broadcast which P_i would make. P_i first sends-to-all BC_i.
   Also, if D conflicted with any P_i in the previous step (namely in ICValidate step 3) then in the following round D will broadcast *all* the values $(s_i, \{t_{ki}\}_k)$. For now, D sends-to-all these values, which we call *public pieces.*

7. **BROADCAST:** Now P_i broadcasts BC_i, which completes WSS-Share step 5/ICValidate step 4, and D broadcasts the values $(s_i, \{t_{ki}\}_k)$ which he sent-to-all in the previous step. Of course each player broadcasts his view of the previous step; if it is not the case that at least $t + 1$ players agree that P_i's broadcast this round matches what he told them in the previous round, then P_i is disqualified.

Additionally, each $P_i \neq D$ broadcasts a random challenge $C_i \in \{0,1\}^\kappa$ for D and for the other P_j's. The challenge indicates, for each index $k \in [\kappa n]$ assigned to P_i (κ such in total), whether:

(1) D and P_j should reveal $f(x) + g_k(x)$, in which case set $v_{kj} = s_j + t_{kj}$ and $v^*_{kj} = s^*_j + t^*_{kj}$; or

(2) D and P_j should reveal $g_k(x)$, in which case set $v_{kj} = t_{kj}$ and $v^*_{kj} = t^*_{kj}$.

8. $\forall k \in [\kappa n]$, $j \in [n]$, P_i participates in $\mathsf{WSS\text{-}Rec\text{-}NoBC}(D, v_{kj})$ and $\mathsf{WSS\text{-}Rec\text{-}NoBC}(P_j, v^*_{kj})$. P_i's outputs from these protocols are $v^{(i)}_{kj}$ and $v^{*(i)}_{kj}$, respectively.

9. **BROADCAST:** Each P_i broadcasts his view of the previous round—namely, the reconstructed shares $v^{(i)}_{kj}$ and $v^{*(i)}_{kj}$, for all k, j.
If a majority of players agrees on a non-$\perp$ reconstructed value for v_{kj} (resp. v^*_{kj}), then such value is the *broadcast (BC) consensus* for the given commitment, and the players who agree *participate in the consensus.* If no BC consensus exists, or if the player who shared the value does not participate, then the sharing player is disqualified. Consequently, if D is not so disqualified, then there exists a BC consensus (which D endorses) for all v_{kj}. Assuming this is the case, then D is nevertheless disqualified if for any k, the set of shares $\{v_{kj}\}_j$, together with appropriate public pieces, does not lie on a polynomial of degree $\leq t$.
In addition to broadcasting his view as described above, D also accuses player P_j, by publicly broadcasting the shares $(s_j, \{t_{kj}\}_k)$, if either of the following occurred:

(1) D output $\perp$ in any $\mathsf{WSS\text{-}Rec\text{-}NoBC}$ instance for which P_j was dealer; or

(2) D reconstructed an incorrect value for P_j's share of any challenge polynomial ($v^{*(D)}_{kj} \neq v_{kj}$).

If any such public pieces fail to lie on the appropriate degree-t polynomial, or if D neglects to accuse P_j when there exists a BC consensus that $v^*_{kj} \neq v_{kj}$, then D is disqualified.
Let HAPPY denote the set of **non-disqualified** players who were **not accused** by D. If $|\mathsf{HAPPY}| < n - t$, then D is disqualified.

Protocol $\mathsf{VSS\text{-}Rec}(\mathcal{P}, s)$

1. Each player $P_i \in \mathsf{HAPPY}$ invokes $\mathsf{WSS\text{-}Rec\text{-}NoBC}(P_i, s_i)$.
Each player $P_i \in \mathcal{P}$ creates a list of shares consisting of those s_j which he accepts from any $\mathsf{WSS\text{-}Rec\text{-}NoBC}(P_j, s_j)$ (including his own), together with all public pieces s_j. He takes any $t+1$ shares from the list, interpolates a polynomial $f(x)$, and outputs $s := f(0)$ as the secret.

Theorem 5. $\mathsf{VSS} = (\mathsf{VSS\text{-}Share}, \mathsf{VSS\text{-}Rec})$ *is a linear verifiable secret sharing scheme secure against an unbounded adversary who corrupts $t < n/2$ players.*

The proof of Theorem 5 is broken up into three lemmas, as follows.

Lemma 6 (Privacy)**.** *If D is honest, then w.h.p. the adversary $\mathcal{A}$ gains no information on s prior to* $\mathsf{VSS\text{-}Rec}$.

Proof. The secret-sharing properties of degree-t polynomials assure that the joint distribution of all shares handed by D to the corrupted parties in step 1, is uniformly random, in particular independent of s.

By the privacy property of protocol WSS employed in steps 2–7, the individual shares $(s_i, \{t_{ki}\}_k)$ of any honest party remain independent of the adversary's view. If in step 7 D broadcasts $(s_i, \{t_{ki}\}_k)$ for some P_i who conflicted with D in an instance of $\mathsf{ICValidate}$, then that P_i must have been corrupt and hence $\mathcal{A}$ already knew these values (as well as the fact that D would broadcast them).

In step 7, $\mathcal{A}$ learns the honest parties' random challenges, which are independent of s and its shares, and thus yield no additional information.

The values reconstructed in step 8 are, for each challenge, either $f(x) + g_k(x)$ or $g_k(x)$. The $g_k(x)$'s themselves were chosen uniformly at random, and until step 8 $\mathcal{A}$ knew nothing about them except for the shares held by corrupt parties, by WSS Privacy. Hence, conditioned on $\mathcal{A}$'s view up to that point, the revealed polynomial is uniformly random subject to consistency with the shares held by corrupted parties. Since D is honest he will answer all challenges correctly, and so $\mathcal{A}$ knows in advance that all honest parties will "accept" D's responses.

In step 9, $\mathcal{A}$ knows in advance what each honest party reconstructed from each WSS, so those broadcasts reveal nothing. Additionally, if D accuses P_j, then D output either $\perp$ or an incorrect value in some instance $\mathsf{WSS\text{-}Rec\text{-}NoBC}(P_j, *)$. This implies w.h.p. that P_j was dishonest (WSS Commitment Without Agreement), in which case $\mathcal{A}$ learns nothing when D broadcasts the shares $(s_j, \{t_{kj}\}_k)$.[6] □

Lemma 7 (Commitment)**.** *With high probability, at the end of* VSS-Share *there exists a fixed* $s^* \in \mathbb{F}$ *such that all honest players output* s^* *during* VSS-Rec*. If* D *is honest, then* $s^* = s$.

Proof. Consider a cheating D. As in the protocol description, we let (s_i, t_{ki}) denote the values which D commits to in steps 2–7, and (s_i^*, t_{ki}^*) the values which P_i commits to. For dishonest players, these may of course be improper commitments. Nevertheless:

Claim 8. *If in step 9 there exists a BC consensus that* P_i *(or* D*) reconstructed* $z \neq \perp$ *in step 8 (not a sum of two shared values), then w.h.p.* P_i *(*D*) properly committed to* z *in steps 2–7.*

Proof. Since BC consensus requires a majority, at least one honest player must be part of the BC consensus. By the Commitment Without Agreement property of WSS, if any honest player reconstructs a non-$\perp$ value z, this value must have been properly committed to by the WSS dealer (w.h.p.). □

Claim 9. *If in step 9 there exists a BC consensus that* P_i *(or* D*) reconstructed a sum of two values,* $z = z_1 + z_2 \;\neq \perp$ *in step 8, then w.h.p. either (1)* z_1, z_2 *were each properly committed to in steps 2–7; or (2)* z_1, z_2 *were both* improperly *committed to in steps 2–7 via polynomials* $h_1(x), h_2(x)$ *of degree* $> t$*, where* $h_1 + h_2$ *is of degree* $\leq t$*. In particular,* P_i *(*D*) was committed at the end of step 7 to a fixed, unique value which honest players would reconstruct as the "sum"* $z_1 + z_2$*.*

Proof. Again at least one honest player must have joined the consensus. Then the claim follows directly from the Proper + Improper property of WSS. □

Recall that a player is disqualified if any value which he ostensibly committed to, lacks a BC consensus in step 9. In light of this, we will say that player P "decommitted" to value v, provided that a BC consensus agrees (in step 9) that P reconstructed v in step 8. In the case that there is no BC consensus (or the consensus is $\perp$), we say P "failed to decommit" and according to the protocol, P is disqualified.

Claim 10. (1) *If any of the shares* s_i^* *committed to by a party* P_i *is improper, i.e.* $s_i^* = \perp$*, then w.h.p.* P_i *will be disqualified.*
(2) *If any of the shares* s_i *committed to by* D *is improper, i.e.* $s_i = \perp$*, then w.h.p.* D *will be disqualified.*

Proof. (1) Technically, P_i is not committed to s_i^* until the end of step 7, since that is when the last step of the WSS protocol takes place. Nevertheless, at the end of step 6, a P_i who will not be disqualified is committed to a set of at most $n - t$ possible polynomials which must be his final commitment. Why? Because P_i is required in step 6 to send-to-all his upcoming broadcast (not including challenges), which fixes his commitment. According to the protocol, he will be disqualified in step 7 unless at least $t + 1$ players agree that he has faithfully re-broadcast the message he sent them in step 6; therefore at least one honest party must have received in step 6 the broadcast which P_i makes in step 7.

Since honest parties also broadcast their *challenges* in step 7, a rushing adversary can make corrupt P_i's commitment depend on these challenges, from among the $n - t$ possibilities fixed in step 6. We must argue that this is not enough freedom for the adversary to dishonestly circumvent the challenges. Consider a *single* possible broadcast, of the potentially $n - t$ which P_i can make in step 7 without being immediately disqualified, and suppose this broadcast represents an improper commitment to $s_i^* = \perp$ (and various t_{ki}^*, each of which may or may not be improper). If some associated $t_{ki}^* \neq \perp$ is a proper commitment, then the Proper + Improper property of WSS implies that w.h.p. P_i will fail to decommit if he must reconstruct $s_i^* + t_{ki}^*$. On the other hand, if P_i made a garbage commitment $t_{ki}^* = \perp$, then he will fail to decommit if challenged to reconstruct t_{ki}^*. It follows that P_i can successfully decommit to *at most one of* $s_i^* + t_{ki}^*$ or t_{ki}^*.

But each honest challenge consists of κ independent requests to reveal either $s_i^* + t_{ki}^*$ or t_{ki}^* (for varying t_{ki}^*). Thus for a fixed honest challenge C_j, P_i will fail to correctly decommit with probability

[6] As this discussion suggests, it may happen that D broadcasts an honest party's shares in step 9; this can only happen if $\mathcal{A}$ succeeds in an IC forgery attempt (hence with negligible probability). As a consequence, our protocol achieves statistical but not perfect privacy. On the other hand, privacy *is* perfect conditioned on the event that $\mathcal{A}$ is unsuccessful in all forgery attempts, as a failed forgery by itself reveals nothing about s.

$\geq 1-2^{-\kappa}$. Thus the probability that C_j does *not* detect $s_i^* = \perp$ is $\leq 2^{-\kappa}$. Now since P_i can choose from $n-t$ possible broadcasts in step 7, the probability that there exists *at least one* improper commitment which C_j fails to detect, is $\leq (n-t)\cdot 2^{-\kappa} = \mathsf{negl}$. Therefore w.h.p if P_i makes *any* improper commitment (from among the up to $n-t$ potentially available to him in step 7), he will be disqualified w.h.p.
(2) The same argument works if we just replace P_i with D, s_i^* with s_i, and t_{ki}^* with t_{ki}. □

Claim 11. *If any of the shares t_{ki} committed to by D is improper, i.e. $t_{ki} = \perp$, then w.h.p. D will be disqualified.*

Proof. By the preceding claim, a non-disqualified D must have properly committed to all s_i (w.h.p.). Condition on an execution where this holds, and suppose D made a garbage commitment to $t_{ki} = \perp$. If D must reconstruct $v_{ki} = s_i + t_{ki}$ in step 8, then by the Proper + Improper property of WSS, he will w.h.p. fail to decommit, and be disqualified. Alternatively, if D must reconstruct $v_{ki} = t_{ki}$ in step 8, he will fail w.h.p. since t_{ki} is itself improperly committed, and be disqualified. He's required to do one of these by the relevant challenge broadcast in step 7. □

Claim 12. *Assume D is not disqualified. If in steps 2–7, P_i committed to $s_i^* \neq s_i$ (i.e., a share different from the one committed to by D), then w.h.p. either P_i will be disqualified, or P_i will be accused by D in step 9.*

Proof. Assume P_i is not disqualified—then there exists a BC consensus in step 9 for the value v_{ki}^*. Since D is not disqualified, there exists also a BC consensus for v_{ki} (which D participates in). Each consensus is (w.h.p.) correct in the sense that P_i and D were committed already in steps 2–7 to reconstruct these values (Claims 8 and 9).

If D does not participate in the BC consensus for v_{ki}^*, then he must accuse P_i. Then P_i will be unhappy, and we are done. Otherwise D does participate in the BC consensus for v_{ki}^*. Then since $s_i^* \neq s_i$, we have that for all k corresponding to some fixed honest challenge, either

$$s_i + t_{ki} \neq s_i^* + t_{ki}^* \qquad \text{or} \qquad t_{ki} \neq t_{ki}^*. \tag{1}$$

Now as in the previous claims, if D and/or P_i are dishonest, they are only partially committed to their shares at the time they see the honest challenges: At the end of step 6, there exist $\leq n-t$ possible broadcasts which each one can make in step 7, finally committing them to a fixed set $(s_i, \{t_{ki}\}_k)$ or $(s_i^*, \{t_{ki}^*\}_k)$. Nevertheless, for any fixed pair of possible broadcasts D and P_i could make in step 7 (and not be disqualified), the probability that a given honest challenge will force D and P_i to reconstruct some unequal values $v_{ki} \neq v_{ki}^*$ is $\geq 1-2^{-\kappa}$. There are (at most) $(n-t)^2$ possible pairs of step 7 broadcasts, hence with probability $\geq 1-(n-t)^2 2^{-\kappa} = 1-\mathsf{negl}$, D and P_i will be committed to reconstruct unequal values regardless of which broadcasts they choose in step 7.

Since BC consensus exists for each of these unequal values (otherwise one or both of D, P_i are disqualified), D will be disqualified unless he accuses P_i in step 9. □

In light of Claims 10 and 11, a cheating D who is not disqualified must have (w.h.p.) committed to genuine field elements $(s_i, \{t_{ki}\}_k)$ for all k, i. Given that this is the case, and with slight abuse of notation, let $f(x)$ denote the polynomial interpolating all s_i, and for each k let $g_k(x)$ be the one interpolating all t_{ki}.

Claim 13. *If $\deg f(x) > t$, then w.h.p. D is disqualified.*

Proof. Suppose that $\deg f(x) > t$. For each fixed k, we have a guarantee that either

$$\deg(f(x) + g_k(x)) > t \qquad \text{or} \qquad \deg g_k(x) > t. \tag{2}$$

D is partially committed to the set $(s_i, \{t_{ki}\}_k)$ at the end of step 6, in that there are at most $n-t$ possible broadcasts he can successfully make in step 7 to conclude the WSS. In step 8, based on the step 7 challenges, he must reconstruct either the values $v_{ki} = s_i + t_{ki}$, i.e. the polynomial $f(x) + g_k(x)$, or the values $v_{ki} = t_{ki}$, i.e. the polynomial $g_k(x)$.

Again, for any *single, fixed* broadcast D could make, and given honest challenge, the probability that he can successfully decommit in step 8 (leading to a BC consensus in step 9 involving only polynomials of degree $\leq t$) is negligible; hence even allowing him to choose from among the $n-t$, his success probability remains negligible, and w.h.p. he will be disqualified. □

Claim 14. *No honest player P_i is disqualified. (Hence at the end of* VSS-Share*, each honest player is either happy, or has been accused by D and his shares made public.)*

Proof. Honest P_i will not be disqualified for misbehaving during any WSS subprotocol. In step 6, P_i will faithfully send his pre-broadcast to all other honest players, hence at least $t+1$ players will confirm it in step 7, and he will not be disqualified there. In step 8, the properties of WSS-Rec-NoBC ensure that for an honest dealer all honest players will correctly reconstruct v^*_{ki}, thus these values will have BC consensus in step 9 (in which P_i participates), and he will not be disqualified there.

As for honest D, the same holds. Additionally, since he will have shared s_i and $\{t_{ki}\}_k$ using polynomials of correct degree, and announce correct public pieces in step 7, then all values v_{ki} which he reconstructs in WSS-Rec-NoBC in step 8 will lie on polynomials of appropriate degree. □

Recall that in VSS-Rec, each happy party whose share s_i is not yet public is supposed to invoke WSS-Rec-NoBC(P_i, s_i) to reveal it. By Claim 12, we see that all parties in HAPPY can reveal only the share $s^*_i = s_i$ (or possibly no share at all, if dishonest). In short, all public pieces and all committed values of happy parties are equal to s_i and thus lie on $f(x)$, which is of degree $\leq t$ (Claim 13). Since only such shares are used during VSS-Rec to construct $f(x)$—and since at least all $\geq t+1$ shares associated with honest parties will be recovered by every honest party—each honest player will reconstruct $f(x)$ and output $s = f(0)$, a value which is fixed by the joint view of the honest parties at the end of VSS-Share.

It is easy to see that if in fact D is honest, then the polynomial $f(x)$ will satisfy $f(0) = s$, and all honest parties will reconstruct the correct value.

Finally, as shown above, if D is dishonest and not disqualified, then w.h.p. the secret s^* which he commits to is $f(0)$, where f interpolates all values s_i which D committed to via WSS. In turn, the values s_i are computable from D's view of the shares D distributed in the WSS sharing phase, which are themselves computable from D's messages in the underlying IC protocols. Hence the criterion for input independence holds as well. □

Lemma 15 (Linearity)**.** *If the parties verifiably share secrets $\{s^{(k)}\}$, then they also (without further interaction) verifiably share any (public) linear combination of the $s^{(k)}$.*

Proof. Consider the situation when parties verifiably share a secret s according to the protocol, for a dealer D who was not disqualified. By Claim 10 of the Commitment proof, we know that w.h.p. D's WSS-commitment to s_i is proper for all i, and Claim 12 ensures that each happy player has properly WSS-committed to the same value. Since happy P_i have made proper WSS-commitments, the linearity of WSS-commitment implies that such P_i can reveal (and are committed to) any linear combination thereof.

Now consider secrets $s^{(k)}$ which are verifiably shared with shares $s_i^{(k)}$, interpolating polynomials $f_k(x)$ all of degree $\leq t$. (We ignore the "shares" of players who are disqualified in some execution of VSS-Share—such players must be corrupt and without loss of generality other players simply ignore their messages and shares during VSS-Rec.) Then any $t+1$ of the summed shares $\sum_k s_i^{(k)}$ interpolate the polynomial $f(x) = \sum_k f_k(x)$, which is of degree $\leq t$ with free term $\sum_k s^{(k)}$.

For any given non-disqualified P_i each share $s_i^{(k)}$ associated with that player is (w.h.p.) either (1) properly WSS-shared among all parties; or (2) publicly known. If *all* the $s_i^{(k)}$ for P_i are publicly known, then other players simply use the public sum $\sum_k s_i^{(k)}$ as P_i's share during VSS-Rec. Otherwise, by the linearity property of WSS P_i can reveal any sum of $s_i^{(k)}$'s. In particular, he can reveal exactly the sum of those $s_i^{(k)}$'s which are not already public, and this is what he does when revealing his "share" in VSS-Rec. This is of course the functional equivalent of revealing the sum of all the $s_i^{(k)}$'s since the other players need only add the public values to the reconstructed value to obtain the "true" share $\sum_k s_i^{(k)}$ of $\sum_k s^{(k)}$. (And revealing the sum of all shares reveals exactly the same information as revealing the sum of the non-public shares.) □

3.3 Putting it all together

Given the $(3,0)$-broadcast VSS protocol from last section, in turn used by the anonymous channel protocol for the pseudosignature setup, we now have all components lined up for our promised result.

Theorem 16. *Given any efficiently computable functionality $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, there exists an n-party protocol Π_f which computes f in the secure-channels-with-broadcast model. Π_f is information-theoretically secure against an active, rushing, static t-adversary $\mathcal{A}$ for any $t < n/2$. Moreover, Π_f requires only* three *rounds of physical broadcast, and these may occur in a constant-round preprocessing phase during which the parties need not know their inputs nor the functionality f to be evaluated.*

Proof sketch. In Π_f, the parties start by invoking the constant-round protocol AnonChan of Section 3.2 with each P_i, $1 \leq i \leq n$, acting as receiver for many sessions in parallel, in order to generate a pseudosignature setup for at least one future broadcast, as in [PW96]. For the VSS subroutine, they use the constant-round VSS protocol of Section 3.1, resulting in a total of 3 broadcasts during the (parallel) sharing phases.

The parties now leverage the information-theoretic PKI to run a constant-round protocol generating sufficiently-many random multiplication triples, for example the 29-round authenticated protocol of Koo [Koo07, Section 4.3.5], which requires only one invocation of authenticated broadcast (simulated over point-to-point channel)[7]. This concludes the preprocessing phase.

To compute the circuit, parties first share their inputs using VSS, e.g. the protocol of Section 3.2, replacing calls to physical broadcast with point-to-point authenticated broadcast. (Note: If the parties already know their inputs in the preprocessing phase, they can share them there during AnonChan and skip this step.)

Then following [Bea91a], the parties can use the multiplication triples to evaluate any arithmetic circuit without further use of broadcast channel, at the cost of one multiplication triple used per multiplication gate and $\Omega(D)$ rounds, where D is the multiplicative depth of the circuit. □

4 Weak Secret Sharing Requires Two Broadcast Rounds

In this section we prove that one broadcast round is not sufficient to implement a weak secret sharing functionality whenever $t \geq \frac{2}{5}n$. Since the WSS sharing phase is itself a specific functionality which could be implemented using a generic MPC scheme, it follows that MPC itself requires two broadcast rounds in general for $t \geq \frac{2}{5}n$.

As an initial observation, we may reduce the general case with $t \geq \frac{2}{5}n$ to the specific case $n = 5$, $t = 2$ using a standard player partitioning argument.

Theorem 17. *Let $\Pi = (\mathsf{WSS\text{-}Share}, \mathsf{WSS\text{-}Rec})$ be a perfectly private WSS protocol for $n = 5$ players, which tolerates a static, computationally unbounded, non-rushing adversary who corrupts at most $t = 2$ parties. If Π uses at most a single physical broadcast during* WSS-Share *and zero broadcasts in* WSS-Rec, *then Π will fail to satisfy the Correctness property with probability $\geq 1/20$.*

Proof sketch. See Appendix B for complete details, including additional remarks.

We consider the player set $\mathcal{P} = \{D, P_1, P_2, P_3, P_4\}$, where the dealer D holds input $s \in \{0, 1\}$.

Now let Π be a WSS protocol as in the statement of the theorem; we must show Commitment is violated with constant probability. For each execution, we divide the rounds of Π's sharing phase WSS-Share into three segments: all *pre-broadcast* rounds; the *broadcast* round itself (which may also include point-to-point communication); and all *post-broadcast* rounds.

At the beginning of the protocol, the adversary randomly selects one of three strategies, call them A, B, and C, which entail the following corruptions:
Strategy A: corrupt D, P_1 (probability 1/4);
Strategy B: corrupt D, P_2 (probability 1/4);
Strategy C: corrupt P_3, P_4 (probability 1/2).

[7] The protocol to generate multiplication triples does *not* require one-way functions, just the (expected) constant-round MPC protocol of Theorem 4.3.9.

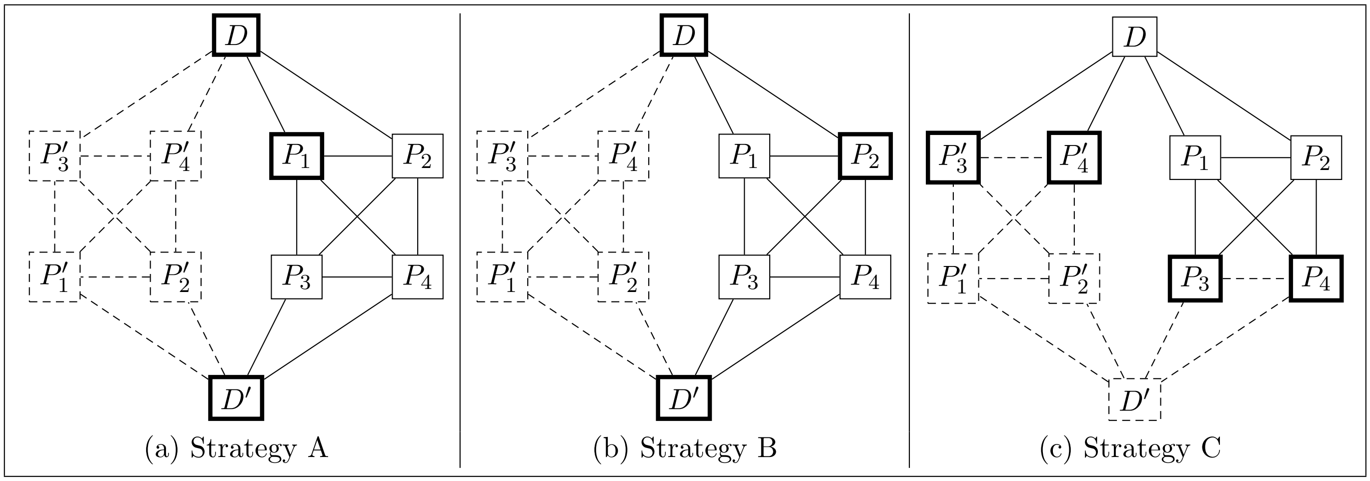


Figure 3: *Pre-broadcast segment for Strategies A, B, and C.*

Pre-broadcast. In the pre-broadcast segment of the protocol, $\mathcal{A}$ internally simulates additional players, and has the cheating players act in such a manner, so as to simulate a distributed computation running on one of the isomorphic networks shown in Figure 3(a)–(c), depending on strategy. Here (copies of) corrupt nodes are bold, and dashed edges and nodes are internally simulated by $\mathcal{A}$.

When $\mathcal{A}$ controls both D and D' (Strategies A, B), they are given random but opposite inputs $s, s' \in \{0,1\}$. Otherwise $\mathcal{A}$ controls only D' (Strategy C). In the latter case, $\mathcal{A}$ makes a random guess about honest D's actual input, and assigns the opposite s' to D'.

The 10 individual (simulated) nodes in each network behave honestly according to their view, and so the joint distribution of their 10 views remains identical whether in (a), (b), or (c).

Broadcast round. In the broadcast round, $\mathcal{A}$ always has corrupt nodes broadcast in such a way that D ends up in conflict with P_3 and P_4 (if anyone). We have chosen the labels so that $\mathcal{A}$ always gives the honest broadcast for the *un*primed copy of a corrupt node. So we have:
Strategy A: $\mathcal{A}$ commits to D, disavows D', broadcasts honestly with P_1;
Strategy B: $\mathcal{A}$ commits to D, disavows D', broadcasts with P_2;
Strategy C: $\mathcal{A}$ commits to P_3, P_4, disavows P'_3, P'_4.

Of course, $\mathcal{A}$ will also send any point-to-point messages as called for by the protocol.

Post-broadcast. Now that there are public conflicts between D and P_3, and between D and P_4 (potentially), the adversary henceforth blocks/disregards all messages between those pairs. Figure 4(a)–(c) illustrate the simulated network which the adversary creates in the post-broadcast segment, depending on strategy (and omitting the blocked channels). The dots at left and right indicate that the connection pattern extends indefinitely in either direction.

In contrast with the situation pre-broadcast, where it was enough to assign random coins (and dealer input) to each node in the simulated network and then let the network evolve "honestly," $\mathcal{A}$ must now assign each node a simulated view of the protocol execution all the way up to the broadcast round (before, again, letting the newly sampled network evolve honestly). For details, see full proof.

Reconstruction. In reconstruction, $\mathcal{A}$ continues to simulate the post-broadcast network of Figure 4.

Now Correctness requires that copies of P_1 and P_2 must both output their dealer's input with high probability (otherwise Correctness is violated in Strategy C). It follows that with similarly high probability the two copies of P_1 and P_2 which P_3 and P_4 are connected to *disagree* (since they have different dealers). But P_3 and P_4, even jointly, cannot distinguish which of P_1, P_2 is dishonest. Therefore with constant probability they will "side" with the dishonest one and output a different value than the honest player, which itself violates Correctness. □

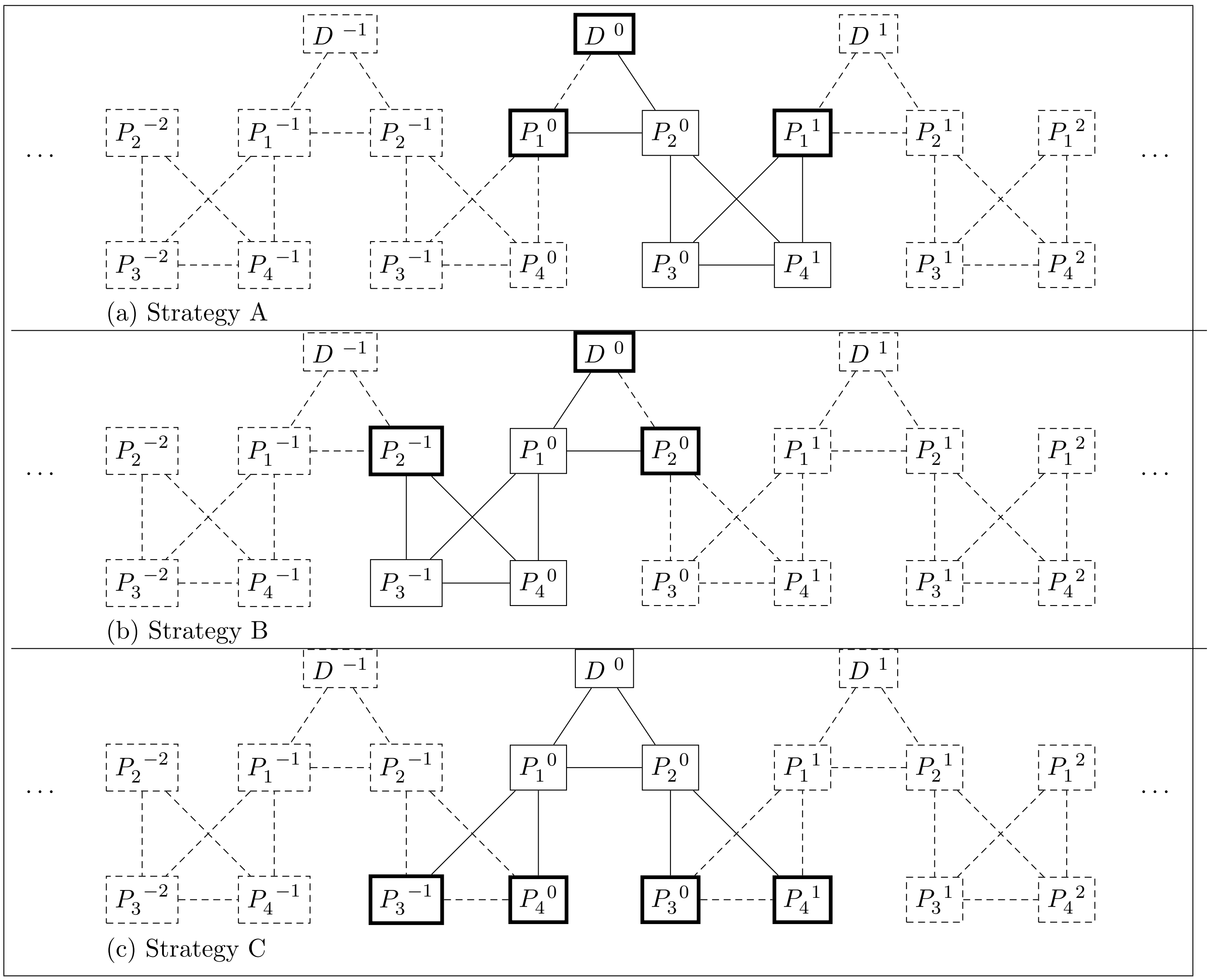


Figure 4: *Post-broadcast segment for Strategies A, B, and C.*

References

[Bea91a] D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1991.

[Bea91b] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *J. Cryptology*, 4(2):75–122, 1991.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th Annual ACM Symposium of the Theory of Computation*, pages 1–10, May 1988.

[BH06] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer Berlin / Heidelberg, 2006.

[BPW91] Birgit Baum-Waidner, Birgit Pfitzmann, and Michael Waidner. Unconditional byzantine agreement with good majority. In *STACS*, pages 285–295, 1991.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings 20th Annual Symposium on Theory of Computing, STOC*. Association for Computing Machinery, May 1988.

[CDD+01] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology–EUROCRYPT'01*, Lecture Notes in Computer Science. Springer Verlag, May 2001.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of Twenty Sixth IEEE Symposium in Foundations of Computer Science*, pages 383–395, 1985.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[Chv79] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285 – 287, 1979.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369, 1986.

[CR90] D. Chaum and S. Roijakkers. Unconditionally secure digital signatures. In A. Menezes and S. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 206–214. Springer, 1990.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of ACM*, 1(40):17–47, 1993.

[DS83] D. Dolev and H. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[FGH+02] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith. Detectable byzantine agreement secure against faulty majorities. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, PODC '02, pages 118–126, New York, NY, USA, 2002. ACM.

[FGMR02] M. Fitzi, N. Gisin, U. Maurer, and O. von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '02, pages 482–501, London, UK, 2002. Springer-Verlag.

[GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *J. Cryptology*, 18(3):247–287, 2005.

[Hag91] T. Hagerup. Fast parallel generation of random permutations. In J. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *ICALP*, volume 510 of *Lecture Notes in Computer Science*, pages 405–416. Springer, 1991.

[KK07] J. Katz and C.-Y. Koo. Round-efficient secure computation in point-to-point networks. In *Proceedings of the 26th annual international conference on Advances in Cryptology*, EUROCRYPT '07, pages 311–328, Berlin, Heidelberg, 2007. Springer-Verlag.

[KKK08] J. Katz, C.-Y. Koo, and R. Kumaresan. Improving the round complexity of VSS in point-to-point networks. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, pages 499–510, Berlin, Heidelberg, 2008. Springer-Verlag.

[Koo07] C.-Y. Koo. *Studies on Fault-Tolerant Broadcast and Secure Computation.* PhD thesis, 2007.

[KPC10] R. Kumaresan, A. Patra, and C. Pandu Rangan. The round complexity of verifiable secret sharing: The statistical case. In *ASIACRYPT*, pages 431–447, 2010.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, July 1982.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM, JACM*, 27(2), April 1980.

[PW96] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$. Technical Report RZ 2882 (#90830), IBM Research, 1996.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st ACM Symposium on the Theory of Computing*, pages 73–85, 1989.

A Information Checking Protocol

Here we give the information checking subprotocol used in our WSS and VSS constructions. It is based on that in [CDD+01] with some minor adjustments to increase broadcast efficiency. The three protocols ICSetup, ICValidate, and ICReveal are given in Figure A.

Definition A.1. Let $s, y, z, \alpha \in \mathbb{F}$. We say that the triple (s, y, z) is 1_α-*consistent* provided that the three points $(0, s)$, $(1, y)$, and (α, z) are colinear over $\mathbb{F}$.[8] One easily verifies that if (s, y, z) and (s', y', z') are 1_α-consistent, then linear combinations of these two vectors are as well.

Remark. Since we ultimately want to run many invocations of the IC protocol in parallel, some of the protocol steps allow events in parallel instances to affect the current instance. Such instructions are set off in square brackets (and can be ignored when considering the scheme as a stand-alone protocol).

Theorem 18. IC = (ICSetup, ICValidate, ICReveal) *is an IC scheme which remains secure when polynomially many instances of the* ICSetup*, and then* ICValidate*, phases are run in parallel. Additionally, it is linear with respect to such invocations.*

The proof is similar to that in [CDD+01] with some minor adjustments. We include it for completeness.

Proof. Correctness. It is easy to see that if D, I, and R are all honest, then they will be satisfied in step 3 of ICValidate, and R will accept s in ICReveal.

Non-Forgery. Since D and R are honest, they will never be in conflict. We claim that I gains no information on α during the ICSetup and ICValidate phases. The values he receives in ICSetup are independent of α. After choosing d in step 1 of ICValidate, I knows exactly what D and R will broadcast in step 3 and so learns nothing. Moreover if D and I conflict, then D broadcasts the values s, y which I again already possessed.

Now consider the situation I is in upon invoking ICReveal for the first time with value s. He knows that if he sends the proper values (s, y), R will accept s, and if he alters just one of them, R will reject. Either way I learns nothing about α, which still appears uniform in $\mathbb{F} - \{0, 1\}$ to him. If he alters both values, to (s^*, y^*), R will accept only if (s^*, y^*, z) is 1_α-consistent. If that is the case then, since (s, y, z) is also 1_α-consistent, it follows that $(s - s^*, y - y^*, 0)$ is as well. From this fact I can deduce the value of α. In other words, in order to get R to accept a false value, I must guess the value of α, and if R does

[8] That is, for some line $L(x) = bx + c$ over $\mathbb{F}$, we have $L(0) = s$, $L(1) = y$, and $L(\alpha) = z$.

Protocol ICSetup(D, I, R, s)**:**

1. Dealer D chooses a random value $\alpha \in \mathbb{F} - \{0,1\}$ and additional values $y, z \in \mathbb{F}$ such that (s, y, z) is 1_α-consistent. [D uses the same α for all parallel instances.] Also he chooses random values $s', y', z' \in \mathbb{F}$ such that (s', y', z') is 1_α-consistent. D sends (s, s', y, y') to the intermediary I, and (α, z, z') to recipient R.

Protocol ICValidate(D, I, R, s)

1. I chooses a random value $d \in \mathbb{F}$ and sends it to D.
2. D sends the triple $(d, s' + ds, y' + dy)$ to R.
3. **BROADCAST:** Each player broadcasts the values he sent and received in the previous two rounds. I broadcasts his view of the triple $(d, s' + ds, y' + dy)$. Additionally, R checks that $(s' + ds, y' + dy, z' + dz)$ is 1_α-consistent; if not R broadcasts "reject values."

   Based on these broadcasts D may be *in conflict* with I and/or R:

   (1) D and I are in conflict if they disagree about the value of the triple $(d, s' + ds, y' + dy)$. [Or if they conflict in a parallel instance.]

   (2) D and R are in conflict if they disagree about what D sent in step 2, or if D is *not* in conflict with I, and R broadcast "reject values." [Or if they conflict in a parallel instance.]

   If no such conflicts arise, then all parties are satisfied and the phase ends here. Otherwise continue to step 4.
4. **BROADCAST:** If D, I are in conflict, then D broadcasts (s, y) and R adjusts z if necessary so that (s, y, z) is 1_α-consistent. This is done regardless whether D, R are in conflict or not, and the phase ends here.

   Otherwise, it must be that D, R are in conflict, but D, I are not. In this case D broadcasts (z, α) and I adjusts y if necessary so that (s, y, z) is 1_α-consistent.

Protocol ICReveal(I, R, s)

1. I sends (s, y) to R, who accepts s if and only if (s, y, z) is 1_α-consistent.

Figure 5: *Information Checking protocol based on [CDD+01].*

not accept, I learns only that this particular α was incorrect. Thus if I makes ℓ attempts, he has at best an $\ell/(|\mathbb{F}| - \ell - 2)$ chance of ever making R accept a false value, which is negligible for polynomial ℓ.

Commitment. Here I and R are honest. If D conflicts with either one of them (or both), the property is trivial. Otherwise, D, I, and R all agree on the values $(d, s' + ds, y' + dy)$, and R accepted $(s' + ds, y' + dy, z' + dz)$ as 1_α-consistent. If for some $e \neq d$, $(s' + es, y' + ey, z' + ez)$ is also 1_α-consistent, then their difference $((d - e)s, (d - e)y, (d - e)z)$ is as well, and hence (s, y, z). Taking the contrapositive: if (s, y, z) distributed by D were *not* 1_α-consistent, there is at most one d which would have led R to accept the values. Since I chooses d randomly, an inconsistent (s, y, z) is detected (and a conflict occurs) except with negligible probability $1/|\mathbb{F}|$.

Privacy. Finally, assume D and I are honest. In the course of ICSetup and ICValidate, a cheating R learns the values $\alpha, z, z', d, s' + ds, y' + dy$. However, he knows that the values $(s' + ds, y' + dy, z' + dz)$ are 1_α-consistent since D and I are honest. This implies that the value of $y' + dy$ can be computed based on $\alpha, z, z', d, s' + ds$, hence it can be removed from the view. This leaves only $\alpha, z, z', d, s' + ds$. Now fixing α, z, z' still leaves s' uniformly random (since y' may be any value), and since s' is used to blind ds, R learns no information on s.

Linearity. The protocol guarantees that R uses the same α value in all instances (for which the first two phases are parallel). The property follows immediately from the prior observation that linear combinations preserve 1_α-consistency. □

B WSS Requires Two Broadcast Rounds (cont'd)

Before presenting the full proof, we gather some additional remarks which we omitted from the main body for brevity.

Remark. When $n = 2t + 1$ exactly, it may be impossible for a WSS scheme to guarantee that an honest D's input remain private *even from the other honest players* until reconstruction. As Rabin and Ben-Or [RB89] note regarding their Theorem 4 (a construction of MPC for honest majority): "As in all the previous results, [our protocol] cannot prevent the bad players from sending all the information they are holding to some honest player. Likewise, if among three players one of the players is faulty and stops sending messages, no further secrecy conditions are imposed on the other two honest players left."

In light of this subtlety, we are careful to give a proof which does *not* assume that honest players gain no information on an honest dealer's secret.

Remark. One natural approach to constructing broadcast-efficient protocols is to start with protocols which use physical broadcast many times, and replace each broadcast with a detectable broadcast protocol [FGMR02, FGH$^+$02] (defined in Related Work section). If ever one of the detectable broadcasts fails, then the parties fall back to a physical broadcast to reconcile their views and proceed.[9]

Such a (hypothetical) protocol has the property that the round number in which physical broadcast is invoked is not fixed in advance, but instead adaptive to the execution. In order to capture the impossibility of this kind of protocol as well, our lower bound does *not* assume a fixed round number for broadcast, but only assumes that at the end of any given round all honest players agree whether the next round will be a broadcast (as would be the case when a detectable broadcast fails).

Remark. Since (in the absence of trusted setup such as public-key infrastructure) the impossibility of Byzantine agreement for $t \geq n/3$ holds even under cryptographic assumptions, it follows that at least one broadcast round is necessary to achieve MPC when $n/3 \leq t < n/2$ regardless of our assumptions on the adversary's computational power. On the other hand, if one-way functions exist then one broadcast round suffices to achieve computationally secure MPC. Indeed: the existence of one-way functions implies that of secure digital signatures. Thus a single broadcast round allows each party to broadcast his public key vk for a signature scheme; from this point on future broadcasts can be simulated using an authenticated broadcast protocol.

Theorem 19. *Let* $\Pi = (\mathsf{WSS\text{-}Share}, \mathsf{WSS\text{-}Rec})$ *be a perfectly private WSS protocol for* $n = 5$ *players, which tolerates a static, computationally unbounded, non-rushing adversary who corrupts at most* $t = 2$ *parties. If* Π *uses at most a single physical broadcast during* WSS-Share *and zero broadcasts in* WSS-Rec*, then* Π *will fail to satisfy the Correctness property with probability* $\geq 1/20$.

Proof. First we establish some notation. We consider the player set $\mathcal{P} = \{D, P_1, P_2, P_3, P_4\}$, where the dealer D holds input $s \in \{0, 1\}$. An *execution* of Π in the presence of a given adversary $\mathcal{A}$ may be identified with a sample of the random coins for all players and for the adversary. Let $\mathsf{View}^{\ell}(S)$ denote the *joint view* of the players in set S at the end of round $\ell \geq 0$, where an individual player's view consists of his random coins and all messages sent and received in rounds 1 to ℓ. As noted above, we allow the broadcast round to be determined dynamically during execution; let bc be the random variable which records this round number, so that $\mathsf{View}^{\mathsf{bc}}(S)$ indicates the joint view of players in S at the end of the broadcast round.

Now let Π be a WSS protocol as in the statement of the theorem; we must show Commitment is violated with constant probability. For each execution, we divide the rounds of Π's sharing phase WSS-Share into three segments: all *pre-broadcast* rounds; the *broadcast* round itself (which may also include point-to-point communication); and all *post-broadcast* rounds.

At the beginning of the protocol, the adversary randomly selects one of three strategies, call them A, B, and C, which entail the following corruptions:

- Strategy A: corrupt D, P_1 (probability 1/4)
- Strategy B: corrupt D, P_2 (probability 1/4)
- Strategy C: corrupt P_3, P_4 (probability 1/2)

[9] Of course the hope is to somehow leverage information gained from detectable broadcast failures to ensure that physical broadcast occurs only once, or at most a few times, rather than for every broadcast in the original protocol!

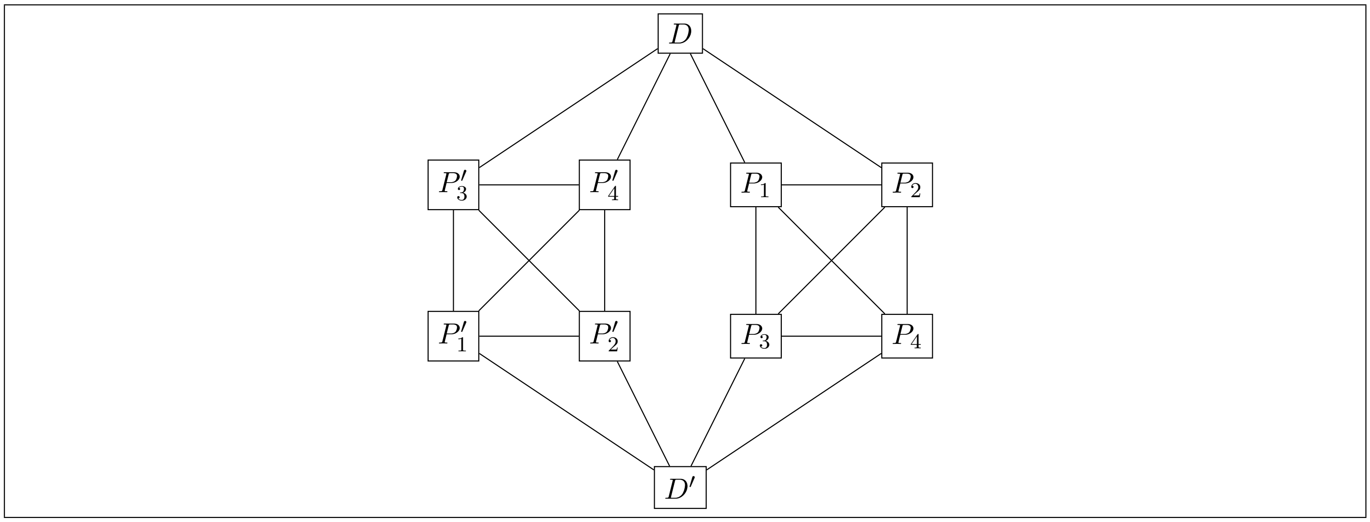


Figure 6: *Simulated network for pre-broadcast segment.*

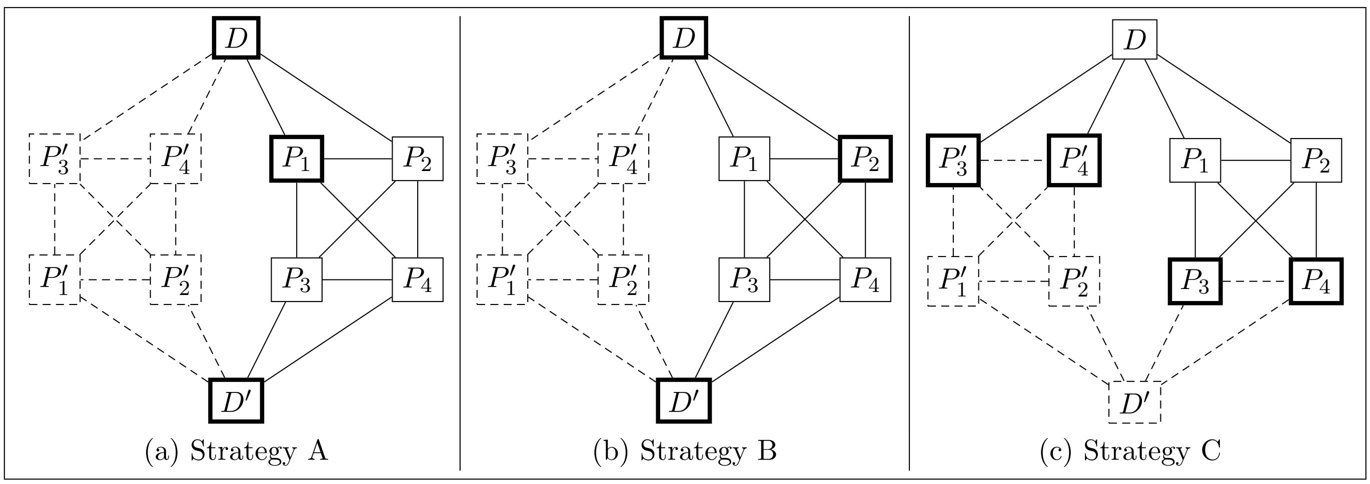


Figure 7: *Pre-broadcast segment for Strategies A, B, and C.*

Pre-broadcast. In the pre-broadcast segment of the protocol, $\mathcal{A}$ internally simulates additional players, and has the cheating players act in such a manner, so as to simulate a distributed computation running on the network shown in Figure 6. . In this network, each node is correctly connected to a copy of each other node in $\mathcal{P}$. Therefore in this simulated network, each (simulated) node simply follows Π honestly.

Although the virtual nodes of Figure 6 each follow Π honestly, in simulating this network $\mathcal{A}$ has one or both of the *actual* corrupted nodes cheat. Diagrams (a)–(c) in Figure 7 illustrate each pre-broadcast strategy. In these diagrams, the bold nodes are corrupted, while dashed nodes and edges represent players and channels which are entirely internally simulated by $\mathcal{A}$. Corrupt nodes which appear twice, in primed and unprimed versions—viz., D/D' in Strategies A and B, and P_3/P_3' and P_4/P_4' in Strategy C—behave as two completely independent, "honest" copies of themselves which happen to be in different positions in the simulated network.

When $\mathcal{A}$ controls both D and D' (Strategies A, B), they are given random but opposite inputs $s, s' \in \{0, 1\}$. Otherwise $\mathcal{A}$ controls only D' (Strategy C). In the latter case, $\mathcal{A}$ makes a random guess about honest D's actual input, and assigns the opposite s' to D'.

Broadcast round. In the broadcast round, there may be no single broadcast message which is consistent with the private messages of both copies of a corrupted node. Therefore, each corrupt node must select just one of its copies, and broadcast a message consistent with the honest view of that copy. We will say informally that $\mathcal{A}$ "commits" to the copy whose broadcast message is chosen, and "disavows" the other copy. As a result, the broadcast round allows honest players who were connected to disavowed copies, to learn that those nodes are dishonest; it also allows other honest players to learn that there is a conflict on these network edges (but crucially, not which of the endpoints are dishonest).[10]

By design, $\mathcal{A}$ always commits in such a way that D ends up in conflict with P_3 and P_4. We have chosen the labels so that $\mathcal{A}$ always commits to the *un*primed copy. So we have:

- Strategy A: $\mathcal{A}$ commits to D, disavows D' (and broadcasts honestly with P_1).
- Strategy B: $\mathcal{A}$ commits to D, disavows D' (and broadcasts honestly with P_2).
- Strategy C: $\mathcal{A}$ commits to P_3 and P_4, disavows P_3' and P_4'.

Of course, $\mathcal{A}$ will also send any point-to-point messages as called for by the protocol.

Before describing the post-broadcast segment, we collect some simple lemmas regarding the pre-broadcast and broadcast segments.

Lemma 20. *Consider the random variables* $\mathsf{View}^{\ell}(all)\,|\mathrm{A}, \{\ell < \mathsf{bc}\}$, $\mathsf{View}^{\ell}(all)\,|\mathrm{B}, \{\ell < \mathsf{bc}\}$, *and* $\mathsf{View}^{\ell}(all)\,|\mathrm{C}, \{\ell < \mathsf{bc}\}, \{s' \neq s\}$, *which record the joint view of all 10 nodes (including the natural "honest" view associated with each corrupt and simulated copy) in the simulated network at the end of round ℓ, conditioned on the adversary's choice of Strategy A, B, and C respectively, and on the broadcast round not having occurred yet. Additionally in Strategy C we condition on the event that $\mathcal{A}$ guesses D's input correctly, so that the two copies of D have unequal inputs. (This happens automatically in Strategies A and B, by construction.) Then the distributions of these random variables are identical, i.e.*

$$\mathsf{View}^{\ell}(all)\,|\mathrm{A}, \{\ell < \mathsf{bc}\} \equiv \mathsf{View}^{\ell}(all)\,|\mathrm{B}, \{\ell < \mathsf{bc}\} \equiv \mathsf{View}^{\ell}(all)\,|\mathrm{C}, \{\ell < \mathsf{bc}\}, \{s' \neq s\}$$

Proof. Prior to the first communication round, the view of each node P (in any of the three strategies) consists only of his random coins (and the random-but-unequal secrets s and s', if $P \in \{D, D'\}$). Hence the claim holds for $\ell = 0$. Then it remains only to observe that (prior to broadcast) the next-message function for each node—whether in Strategy A, B, or C—is simply the honest next-message function of Π applied to that node's current view. Since each experiment begins identically distributed and then evolves according to identical, deterministic rules (pre-broadcast), the distribution of views remains identical conditioned on broadcast not having yet occurred. □

The following lemma establishes that the joint view remains identically distributed regardless of the adversary's strategy, even after the broadcast round itself, when restricted to those nodes from the simulated network which actually send a broadcast (as opposed to fully simulated nodes and disavowed copies).

Lemma 21. *Consider the random variables* $\mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{A}$, $\mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{B}$, *and* $\mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{C}, \{s' \neq s\}$, *which represent the joint view of all 5 players in $\mathcal{P}$ at the end of the broadcast round, conditioned on $\mathcal{A}$'s strategy. (Here the view of a corrupted player is the "honest" view associated with the copy $\mathcal{A}$ committed to during broadcast.) Then they are identical:*

$$\mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{A} \equiv \mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{B} \equiv \mathsf{View}^{\mathsf{bc}}(all)\,|\mathrm{C}, \{s' \neq s\}$$

Proof. By construction, $\mathcal{A}$ always "commits" in such a way that, in terms of Figure 6, the broadcasting nodes will be D, P_1, P_2, P_3, and P_4 (as opposed to any of their primed versions). By the previous lemma, the joint view of these 5 nodes in the larger hypothetical/simulated network is identically distributed (for each fixed round ℓ), irrespective of $\mathcal{A}$'s choice of Strategy. Then when the broadcast round occurs, each of the broadcasting nodes simply sends its broadcasts and private messages honestly (according to its view). Hence the distribution remains independent of $\mathcal{A}$'s strategy when we take the broadcast round into account. □

Corollary 22. *Condition on the event that the sharing phase of Π terminates before a single broadcast round occurs. Suppose that in Strategy A, with probability $\geq 1 - \epsilon$ the adversary is committed to a value s^* at the end of sharing phase. (Here commitment means that during reconstruction, with probability $\geq 1 - \epsilon$, honest parties will output either s^* or $\perp$ [with agreement]). Then in Strategy C,*

[10] Of course whether or not potential conflicts are always detected, by the honest parties involved or by the others, depends on the details of Π. Nevertheless, it is good intuition, and we shall continue to speak as though the broadcast round necessarily introduced explicit conflicts, although the proof does not rely on this.

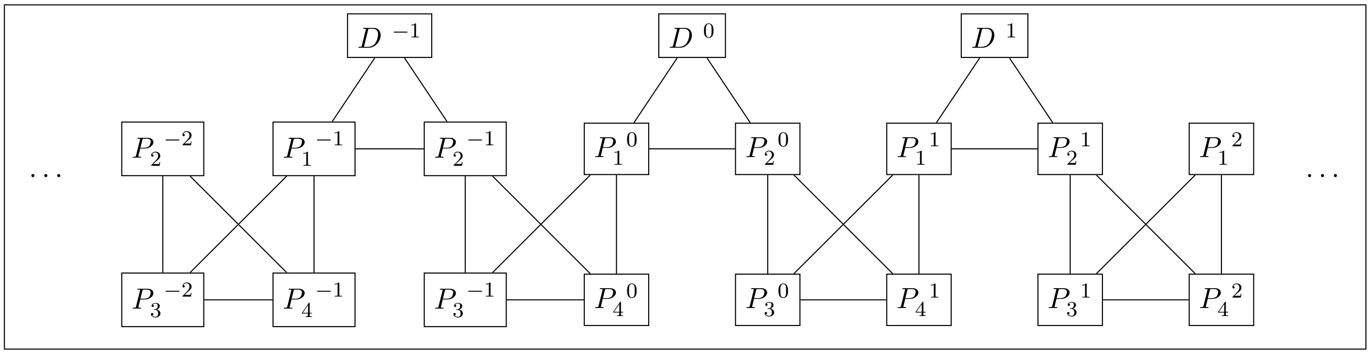


Figure 8: *Simulated network for post-broadcast segment.*

Post-broadcast. Now that there are public conflicts between D and P_3, and between D and P_4, the adversary will block all messages between those conflicting pairs. That is to say: in Strategies A and B, $\mathcal{A}$ sends no messages $D \to P_3$ or $D \to P_4$, and alters the view of D by omitting all post-broadcast messages from them. Similarly, in Strategy C, $\mathcal{A}$ sends no messages $P_3 \to D$ or $P_4 \to D$, and alters the views of P_3 and P_4 to exclude post-broadcast messages from D, if any.

Figure 8 illustrates the simulated network which the adversary creates in the post-broadcast segment; for clarity, it omits blocked private channels (those of type $D \leftrightarrow P_3$ and $D \leftrightarrow P_4$). The dots at left and right indicate that the connection pattern extends infinitely in either direction. The infinity is merely a formal convenience, since the length of chain which must actually be constructed by $\mathcal{A}$ is finite and proportional to the number of post-broadcast and reconstruction rounds. The superscripts, which range over $\mathbb{Z}$, distinguish different translated copies of the same underlying player in $\mathcal{P}$. Although it is not indicated in the diagram, the adversary's aim is that alternate copies of the dealer have opposite input values; i.e. $D^{\,k}$ holds $s_k = 0$ for k even and $s_k = 1$ for k odd, or vice versa. In Strategy A and B, this will be ensured directly since all copies of the dealer are corrupt/simulated. In Strategy C, the adversary uses his previous guess for the dealer's bit in order to determine which value should be given to simulated copies of D.

For the adversary's behavior in each particular strategy, refer to Figure 9, in which again dashed edges and nodes are internally simulated by $\mathcal{A}$, and bold nodes are (copies of) corrupt nodes.

For sets of nodes S and T, define

$$\mathsf{Common}^{\ell}(S|T) := \mathsf{View}^{\ell}(S) \cap \mathsf{View}^{\ell}(T)$$

to be the partial transcript shared between S and T, up to round ℓ. For example, $\mathsf{Common}^{\mathsf{bc}}(D, P_1|P_2)$ consists of:

- The broadcasts of *all* parties;
- private messages $D \leftrightarrow P_2$;
- private messages $P_1 \leftrightarrow P_2$.

Note that $\mathsf{Common}^{\mathsf{bc}}(D, P_1|P_2)$ does *not* include private messages $D \leftrightarrow P_1$, since those messages are internal to the set $\{D, P_1\}$, rather than shared between the two sets.

At this time the reader may wish to refer to diagrams (a)–(c) in Figure 9, which identify the honest, corrupt, and simulated nodes for each Strategy as it continues post-broadcast. In contrast with the situation pre-broadcast, where it was enough to assign random coins (and dealer input) to each node in the simulated network and then let the network evolve "honestly," $\mathcal{A}$ must now assign each node a simulated view of the protocol execution all the way up to the broadcast round.

$\mathcal{A}$ has six "moves" at his disposal to extrapolate the chain leftward or rightward. For simplicity, we omit superscripts. We do allow $\mathcal{A}$, when sampling a new copy of a dealer (moves (L1) and (R1)), to choose that copy's input $s^* \in \{0, 1\}$. Recall this is necessary since $\mathcal{A}$ wishes to alternate the inputs assigned to consecutive dealer copies. That this is possible is a consequence of Lemma 25, which states that the individual view $\mathsf{View}^{\mathsf{bc}}(P_1)$ (resp., $\mathsf{View}^{\mathsf{bc}}(P_2)$) is independent of the actual secret s.

Leftward moves

(L1) $(P_1, D(s^*)) \dashleftarrow P_2$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_2)$. To extend leftward, he obtains views for P_1 and D by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(P_1, D)\big|\mathsf{Common}^{\mathsf{bc}}(P_1, D|P_2), \{s = s^*\}$.

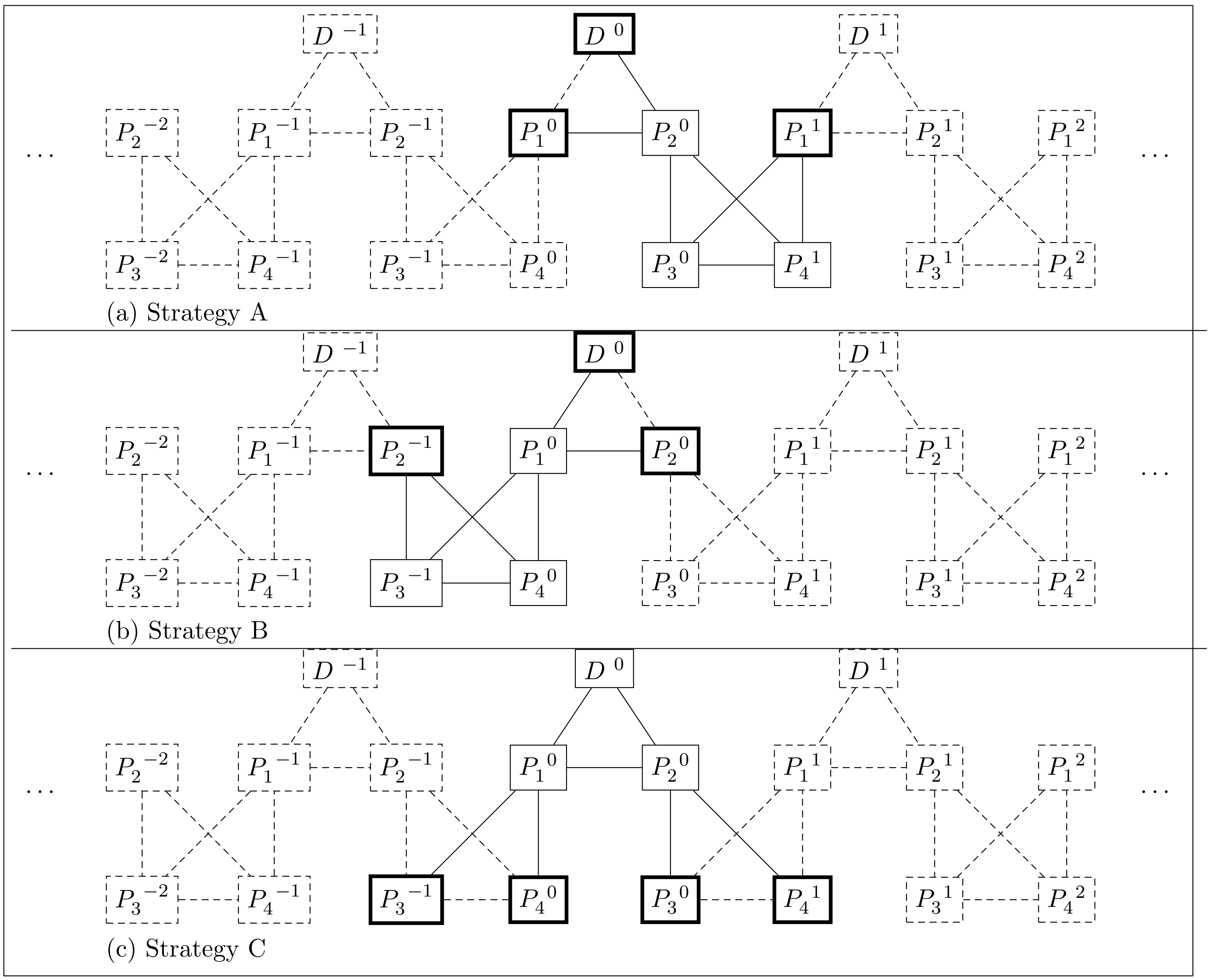


Figure 9: *Post-broadcast segment for Strategies A, B, and C.*

(L2) $(P_3, P_4) \dashleftarrow P_1$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_1)$. To extend leftward, he obtains views for P_3 and P_4 by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(P_3, P_4)\big|\mathsf{Common}^{\mathsf{bc}}(P_3, P_4|P_1)$.

(L3) $P_2 \dashleftarrow (P_3, P_4)$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_3, P_4)$. To extend leftward, he obtains a view for P_2 by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(P_2)\big|\mathsf{Common}^{\mathsf{bc}}(P_2|P_3, P_4)$.

Rightward moves

(R1) $P_1 \dashrightarrow (D(s^*), P_2)$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_1)$. To extend rightward, he obtains views for D and P_2 by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(D, P_2)\big|\mathsf{Common}^{\mathsf{bc}}(P_1|D, P_2), \{s = s^*\}$.

(R2) $P_2 \dashrightarrow (P_3, P_4)$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_2)$. To extend rightward, he obtains views for P_3 and P_4 by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(P_3, P_4)\big|\mathsf{Common}^{\mathsf{bc}}(P_2|P_3, P_4)$.

(R3) $(P_3, P_4) \dashrightarrow P_1$: $\mathcal{A}$ knows $\mathsf{View}^{\mathsf{bc}}(P_3, P_4)$. To extend rightward, he obtains a view for P_1 by sampling from the conditional distribution $\mathsf{View}^{\mathsf{bc}}(P_1)\big|\mathsf{Common}^{\mathsf{bc}}(P_3, P_4|P_1)$.

With these moves in hand, it is a straightforward matter to see how $\mathcal{A}$ generates the chains for each of the three strategies.

Strategy A. To extend leftward, first re-sample a joint view $\mathsf{View}^{\mathsf{bc}}({P_1}^0, D^{\ 0})$ by simulating move (L1). (This first move is not *technically* (L1) as we described it because in this case the adversary actually knows the joint view $\mathsf{View}^{\mathsf{bc}}(P_1, D)$ rather than the view $\mathsf{View}^{\mathsf{bc}}(P_2)$, but the effect is identical since either one determines the shared view $\mathsf{Common}^{\mathsf{bc}}(P_1, D|P_2)$.) Then continue leftward from ${P_1}^0$ by repeating the moves (L2), (L3), (L1), ...

To extend rightward, re-sample a view $\mathsf{View}^{\mathsf{bc}}({P_1}^1)$ by simulating move (R3) (as before, this is not *technically* (R3) since $\mathcal{A}$ uses knowledge of $\mathsf{View}^{\mathsf{bc}}(P_1)$ rather than of $\mathsf{View}^{\mathsf{bc}}(P_3, P_4)$; we do not belabor the point in Strategy B and C). Then continue rightward from ${P_1}^1$ by repeating the moves (R1), (R2), (R3), ...

Strategy B. To extend leftward, first re-sample a view $\mathsf{View}^{\mathsf{bc}}({P_2}^{-1})$ by simulating move (L3). Then continue leftward by repeating the moves (L1), (L2), (L3), ...

To extend rightward, first re-sample a joint view $\mathsf{View}^{\mathsf{bc}}(D^{\ 0}, {\mathcal{P}_2}^0)$ by simulating move (R1). Then continue rightward by repeating the moves (R2), (R3), (R1), ...

Strategy C. To extend leftward, first re-sample a joint view $\mathsf{View}^{\mathsf{bc}}({P_3}^{-1}, {P_4}^0)$ by simulating move (L2). Then continue leftward by repeating the moves (L3), (L1), (L2), ...

To extend rightward, first re-sample a joint view $\mathsf{View}^{\mathsf{bc}}({P_3}^0, {P_4}^1)$ by simulating move (R2). Then continue rightward by repeating the moves (R3), (R1), (R2), ...

Having sampled appropriate views for all necessary copies of nodes, $\mathcal{A}$ generates all subsequent messages for corrupt and simulated nodes according to Π's next-message function for that node's underlying player, given that node's current view (which, of course, evolves each round as messages from its neighbors are received). We note in passing that it is not necessary for $\mathcal{A}$ to know in advance any bound whatsoever on the number of post-broadcast rounds, since he can simply extend the chain diagram one move in each direction per round.

To analyze the chains so constructed, consider the types of triangles which appear in them—namely, (P_2, P_3, P_4), (P_1, P_3, P_4), and (D, P_1, P_2). Given $k \in \mathbb{Z}$, we define the *kth translate* of each of these triples to be the specific triangle of players in Figure 8 indexed by k—to wit:

- $T_{234}(k) := ({P_2}^k, {P_3}^k, {P_4}^{k+1})$
- $T_{134}(k) := ({P_1}^k, {P_3}^{k-1}, {P_4}^k)$
- $T_{D12}(k) := (D^{\ k}, {P_1}^k, {P_2}^k)$

The reader will observe (Figure 9(a)–(c)) that in Strategy A, the honest players are exactly $T_{234}(0)$; in Strategy B, the honest players are $T_{134}(0)$; and in Strategy C, they are $T_{D12}(0)$. The next lemma asserts that *every* triangle in the chain (whether for Strategy A, B, or C) has its joint view distributed identically with the appropriate "honest" triangle.

Lemma 23. *Fix $k \in \mathbb{Z}$, and consider the triangles of (possibly corrupt/simulated) nodes $T_{234}(k)$, $T_{134}(k)$, and $T_{D12}(k)$, whose views are defined by the adversary's sampling process described above (in any of Strategy A, B, or C).*

Then the following pairs of joint views are, for each pair, identically distributed, provided that in case (3), we further condition on the adversary's guess for the dealer's bit being correct when the adversary uses Strategy C:

$$\begin{aligned}
\mathsf{View}^{\mathsf{bc}}\left(T_{234}(k)\right) &\equiv \mathsf{View}^{\mathsf{bc}}\left(T_{234}(0)\right) && (1)\\
\mathsf{View}^{\mathsf{bc}}\left(T_{134}(k)\right) &\equiv \mathsf{View}^{\mathsf{bc}}\left(T_{134}(0)\right) && (2)\\
\mathsf{View}^{\mathsf{bc}}\left(T_{D12}(k)\right) &\equiv \mathsf{View}^{\mathsf{bc}}\left(T_{D12}(0)\right) \quad \textit{if } k \equiv 0 \pmod 2 && (3)\\
\mathsf{View}^{\mathsf{bc}}\left(T_{D12}(k)\right) &\equiv \mathsf{View}^{\mathsf{bc}}\left(T_{D12}(1)\right) \quad \textit{if } k \equiv 1 \pmod 2 && (4)
\end{aligned}$$

Proof. (Sketch.) The proof is inductive, and specifically is an immediate consequence of the fact that each of the six moves described above, when applied to an initial (joint) view with "honest" marginal distribution, correctly samples an "honest" joint distribution on the triangle of views. Here by an honest marginal/joint distribution, we mean precisely (the appropriate restriction of) the one guaranteed by Lemma 21 to be independent of Strategy. In particular the chain for each Strategy (A, B, or C) has as its "base case" for leftward or rightward extrapolation, an honest triangle of nodes ($T_{234}(0)$, $T_{134}(0)$, or $T_{D12}(0)$ respectively) which will have this honest distribution. We again suppress the superscripts for ease of understanding.

For example, consider (L1), a.k.a. $(P_1, D(s^*)) \dashleftarrow P_2$, and suppose the view $\mathsf{View}^{\mathsf{bc}}(P_2)$ which the adversary uses to extend leftward has correct marginal distribution (i.e., the joint distribution guaranteed by Lemma 21 to be independent of adversary strategy, restricted to node P_2's view). Then the experiment of (L1)—in which a view for P_2 is chosen and then, conditioning on that view, extended to a joint view for $P_1, D(s^*), P_2$—is identical to an experiment in which the joint view is sampled *directly*. The distribution of the latter experiment is once again exactly that appearing in Lemma 21.

The argument for each of the other five moves follows in like manner. □

Reconstruction. During reconstruction, $\mathcal{A}$ simply continues to simulate the post-broadcast network of Figure 8 continuing to use it to determine what messages to send to the honest players in each round of Reconstruction. (Recall we assume that the reconstruction phase does not include broadcast.)

Lemma 24. *Consider the same pairs of joint views as in Lemma 23, but extend the views to the* entire WSS protocol. *Then the identities still hold (again conditioned on adversary's correct guess if in Strategy C), namely:*

$$\begin{aligned}
\mathsf{View}^{\mathsf{WSS}}\left(T_{234}(k)\right) &\equiv \mathsf{View}^{\mathsf{WSS}}\left(T_{234}(0)\right) && (1)\\
\mathsf{View}^{\mathsf{WSS}}\left(T_{134}(k)\right) &\equiv \mathsf{View}^{\mathsf{WSS}}\left(T_{134}(0)\right) && (2)\\
\mathsf{View}^{\mathsf{WSS}}\left(T_{D12}(k)\right) &\equiv \mathsf{View}^{\mathsf{WSS}}\left(T_{D12}(0)\right) \quad \textit{if } k \equiv 0 \pmod 2 && (3)\\
\mathsf{View}^{\mathsf{WSS}}\left(T_{D12}(k)\right) &\equiv \mathsf{View}^{\mathsf{WSS}}\left(T_{D12}(1)\right) \quad \textit{if } k \equiv 1 \pmod 2 && (4)
\end{aligned}$$

Proof. This follows immediately from Lemma 23, the symmetry of the chain diagrams, and the fact that, conditioned on the views sampled up to the end of the broadcast round, the protocol evolves deterministically until the end of reconstruction (since there is no further broadcast round). □

Now based on these indistinguishability results we make the following observations, using also the $(1-\epsilon)$-correctness of the WSS.

(1) Copies of P_1 and P_2 in the same triangle as a given dealer both output the input held by that dealer with probability $\geq \frac{1}{-}2\epsilon$ (otherwise correctness is violated since $\mathcal{A}$ chooses Strategy C, and guesses correctly, with probability 1/2);

Hence with probability $\geq 1 - 4\epsilon$ in Strategy A (or B), the copies of P_1 and P_2 which (P_3, P_4) are connected to both agree with their dealers and hence output *different values* (since dealers have alternating values).

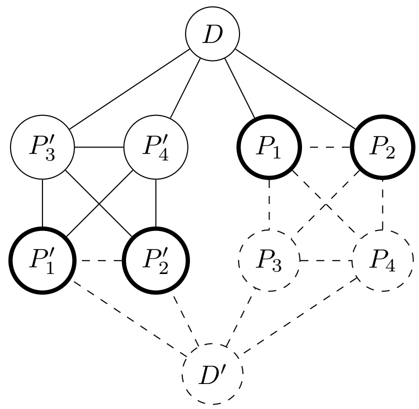
Since (P_3, P_4) cannot distinguish which scenario they are in (hence which of P_1, P_2 to agree with), then (conditioned on the event that P_1, P_2 disagree), in at least one of the strategies they will fail to agree with the correct player with probability at least 1/2. Without loss of generality this occurs in Strategy A. Since the probability of Strategy A is 1/4, we have that the failure probability is $\epsilon \geq (1-4\epsilon)(1/2)(1/4)$. Solving yields $\epsilon \geq 1/20$. This completes the proof of the lower bound.

□

It remains only to prove the lemma which we deferred from earlier.

Lemma 25. *The individual view* $\mathsf{View}^{\mathsf{bc}}(P_1)$ *(resp.,* $\mathsf{View}^{\mathsf{bc}}(P_2)$*) in any of the adversary strategies A,B,C is independent of the secret s held by the dealer D.*

Proof. Even though $\mathcal{A}$ never corrupts P_1 and P_2 at the same time (nor either one individually without corrupting D), we can describe an adversary $\mathcal{A}'$ who *does* corrupt P_1 and P_2 and simulates the same network as follows:



Now the distribution of players' views in the diagram above is identical with their distribution in diagrams 7(a)–(c) in the pre-broadcast segment (Lemma 20). But in the diagram above, P_1 and P_2 are corrupt, and so their joint view should remain independent of s. This holds also when we take into account the broadcast round itself: for if not, then the broadcasts and private messages of D, and of the *simulated* P_3 and P_4 in the above diagram (corresponding to the actual nodes P_3 and P_4 in an execution under Strategy A, B, or C), would reveal information about s. But P_1 and P_2 see D's broadcasts and private messages to them, and $\mathcal{A}$ can simulate the broadcasts and private messages from P_3, P_4 to P_1, P_2, so that if these revealed information about s, then $\mathcal{A}'$ would gain information about s in the diagram above, contradicting WSS privacy. □