On Expected Constant-Round Protocols for Byzantine Agreement[*]

Jonathan Katz[†‡] Chiu-Yuen Koo[†]

Abstract

In a seminal paper, Feldman and Micali show an n-party Byzantine agreement protocol in the plain model that tolerates $t < n/3$ malicious parties and runs in expected constant rounds. Here, resolving a question that had been open since their work, we show an expected constant-round protocol for *authenticated* Byzantine agreement assuming *honest majority* (i.e., $t < n/2$), and relying only on the existence of signature schemes and a public-key infrastructure. Combined with existing results, this gives the first expected constant-round protocol for secure computation with honest majority in a point-to-point network under the same assumptions. Our key technical tool — a new primitive we introduce called *moderated VSS* — also yields a simpler proof of the Feldman-Micali result.

In addition, we show a simple technique for sequential composition of Byzantine agreement protocols that do not achieve simultaneous termination, something that is inherent for protocols using $o(t)$ rounds.

[*]A preliminary version of this work appeared in *Advances in Cryptology — Crypto 2006*.

[†]Dept. of Computer Science, University of Maryland. `{jkatz,cykoo}@cs.umd.edu`.

[‡]This research was supported by NSF Trusted Computing grant #0310751, NSF CAREER award #0447075, and US-Israel Binational Science Foundation grant #2004240.

1 Introduction

When designing cryptographic protocols, it is often convenient to abstract away various details of the underlying communication network. As one noteworthy example, protocol designers frequently assume the existence of a *broadcast channel* that allows any party to send the same message to all other parties (and all parties to be assured they have received identical messages) in a single round.[1] With limited exceptions (e.g., in a small-scale wireless network), it is understood that the protocol will be run in a network where only point-to-point communication is available and the parties will have to "emulate" the broadcast channel by running a broadcast sub-protocol. Unfortunately, this "emulation" typically increases the round complexity of the original protocol substantially.

Much work has therefore focused on reducing the round complexity of protocols for broadcast or the related task of *Byzantine agreement* (BA) [39, 35]; we survey this work in Section 1.2. As discussed there, a seminal result of Feldman and Micali [20] is a protocol for n-party Byzantine agreement tolerating $t < n/3$ malicious parties and running in expected *constant* rounds. This resilience is the best possible — regardless of round complexity — unless some set-up assumptions are made. The most common assumption is the existence of a public-key infrastructure (PKI) such that each party P_i has a public key pk_i for a digital signature scheme that is known to all other parties (a more formal definition is given in Section 2); protocols in this model are termed *authenticated.* Authenticated broadcast protocols are known for $t < n$ [39, 35, 16], but all previously existing protocols[2] that assume only a PKI and digital signatures require $\Theta(t)$ rounds. Fitzi and Garay [22] give an authenticated BA protocol that beats this bound for $t < n/2$, but their protocol relies on specific number-theoretic assumptions; see Section 1.2 for further discussion.

1.1 Our Contributions

As our main result, we extend the work of Feldman and Micali and show an authenticated BA protocol tolerating $t<n/2$ malicious parties and running in expected constant rounds. Our protocol assumes only the existence of signature schemes and a PKI, and is secure against a rushing adversary who adaptively corrupts up to t parties. For those unfamiliar with the specifics of the Feldman-Micali protocol, we stress that their approach does *not* extend to the case of $t < n/2$. In particular, they rely on a primitive termed *graded verifiable secret sharing* (graded VSS) and construct this primitive using in an essential way the fact that $t < n/3$. We take a different approach: we introduce a primitive called *moderated VSS* and use this to give an entirely self-contained proof of our result.

We suggest that moderated VSS is a useful primitive *in general*, even when $t < n/3$. For one, moderated VSS seems easier to construct: we show a generic construction of moderated VSS *in the point-to-point model* from any VSS protocol that relies on a broadcast channel, while a direct construction of this sort for graded VSS is unknown. Perhaps more importantly, moderated VSS provides what we believe to be a conceptually-simpler approach to the problem at hand: in addition to our authenticated BA protocol for $t < n/2$, our techniques give a BA protocol (in the plain model) for $t < n/3$ whose concrete round complexity is better than that of the Feldman-Micali protocol and whose proof is, in our opinion, significantly simpler than that of [20].

Since moderated VSS is impossible for $t \geq n/2$, our techniques do not extend to the case when there is no honest majority. It remains an interesting open question as to whether it is possible to

[1]Other "abstractions" include the assumptions of private and/or authenticated channels. For non-adaptive, computationally-bounded adversaries these can be realized without affecting the round complexity using public-key encryption and digital signatures, respectively. See also Section 2.

[2]Although Feldman and Micali claim an expected $O(1)$-round solution for $t < n/2$ in the conference version of their paper, this claim no longer appears in either the journal version of their work or Feldman's thesis [18].

achieve broadcast for $n/2 \leq t < n$ in expected constant rounds. See the recent work of Garay et al. [25] for some partial progress in this direction.

As mentioned earlier, cryptographic protocols are often designed under the assumption that a broadcast channel is available; when run in a point-to-point network, these protocols must "emulate" the broadcast channel by running a broadcast protocol as a sub-routine. If the original protocol uses multiple invocations of the broadcast channel, and these invocations are each emulated using an $o(t)$-round broadcast protocol, difficulties related to the parallel and sequential *composition*[3] of the various broadcast sub-protocols arise; see the detailed discussion in Section 4. Parallel composition can be dealt with using existing techniques [4, 22]. There are also techniques available for handling sequential composition [4, 37]; however, these techniques apply only to the case $t < n/3$ [4] or are rather complex [37]. As an additional contribution of this work, we show a method for sequential composition that applies when $t < n/2$ (assuming digital signatures and a PKI) and is simpler and more round efficient than prior work.

The above results, together with prior work [2, 15], yield the first expected constant-round protocol for secure computation in a point-to-point network that tolerates an adversary corrupting a minority of the parties and is based only on the existence of one-way functions and a PKI. (A constant-round protocol of Goldwasser and Lindell [29], which does not require a PKI, achieves a weaker notion of security that does not guarantee complete fairness or output delivery; see further discussion in Section 4.3.)

1.2 Prior Work on Broadcast/Byzantine Agreement

Definitions for broadcast and Byzantine agreement are given in Section 2. The problem of achieving broadcast makes sense for any number of corrupted players, whereas Byzantine agreement is only well-defined when $t < n/2$. It is easy to see that for $t < n/2$ any Byzantine agreement protocol implies a broadcast protocol using one additional round: in the first round the dealer sends its input to all players, who then run a Byzantine agreement protocol on the values they received.

In a synchronous network with pairwise private, authenticated channels and no additional setup assumptions, BA among n parties is achievable if and only if the number of corrupted parties t satisfies $t < n/3$ [39, 35]. (Unless otherwise stated, impossibility results hold even for computationally bounded adversaries while feasibility results hold even for computationally unbounded adversaries.) Concerning round complexity, a lower bound of $t + 1$ rounds for any deterministic BA protocol is known in this setting [21]. A protocol with this round complexity — but with exponential message complexity — was shown by Pease, et al. [39, 35]. Following a long sequence of works, Garay and Moses [26] show a fully-polynomial, deterministic BA protocol with optimal resilience and round complexity. Private channels are not assumed in these results.

To circumvent the above-mentioned lower bound on the round complexity, researchers beginning with Ben-Or [3] and Rabin [41] explored the use of randomization. This line of research [8, 12, 19, 17] culminated in the work of Feldman and Micali [20], who show a randomized BA protocol with optimal resilience $t < n/3$ that runs in expected *constant* rounds. Their protocol, as well as most work in this direction, assumes private channels (but see footnote 1 and the next section); see [6, 30] for recent work showing $o(t)$-round randomized BA protocols without this assumption.

To achieve resilience $t \geq n/3$, additional assumptions are needed even if randomization is used. The most widely-used assumptions are the existence of signatures and a public-key infrastructure

[3]These issues are unrelated to those considered in [36] where the different executions are oblivious of each other: here, there is a single "outer" protocol scheduling all the broadcast sub-protocols. We do not consider *concurrent* composition since we are interested only in "stand-alone" security of the outer protocol.

(PKI); recall that protocols in this setting are termed *authenticated.* When analyzing protocols in this model, signatures are assumed to be ideal in the sense that it is simply assumed to be *impossible* for the adversary to forge signatures on behalf of honest parties. (For this reason, the adversary may still be taken to be unbounded in the analysis and perfect security can be obtained.) These signatures can then be instantiated using either cryptographically-secure digital signatures [42] to guarantee security against a computationally-bounded adversary, or using information-theoretic "pseudo-signatures" [40] to obtain statistical[4] security even against an unbounded adversary. (As in the case of a PKI, the keying material needed for implementing pseudo-signatures is assumed to be provided to the parties before the protocol begins.)

Pease et al. [39, 35] show an authenticated broadcast protocol for $t < n$ with exponential complexity. (Since signatures are treated as ideal, it is meaningful to discuss protocols having exponential complexity. A secure realization of the protocol then requires either information-theoretic pseudo-signatures, or cryptographic signatures secure against an exponential-time adversary.) A fully-polynomial protocol with resilience $t < n$ was given by Dolev and Strong [16]. These works do not require private channels.

The $(t + 1)$-round lower bound for deterministic protocols holds in the authenticated setting as well [16], and the protocols of [39, 35, 16] meet this bound. (Although the setup phase and signature generation may be probabilistic, recall that in the analysis signatures are treated as ideal and it is therefore still meaningful to talk of "deterministic protocols" in the authenticated setting.) Some randomized protocols beating this bound for the case of $n/3 \leq t < n/2$ are known [44, 8, 46], but these are only partial results:

- Toueg [44] gives an expected $O(1)$-round protocol, but assumes a trusted dealer. Furthermore, after the dealing phase the parties can only run the BA protocol a *bounded* number of times.
- A protocol by Bracha [8] implicitly requires a trusted dealer to ensure that parties agree on a "Bracha assignment" in advance (see [19]). Furthermore, the protocol only achieves expected round complexity $\Theta(\log n)$ and tolerates (slightly) sub-optimal $t \leq n/(2 + \epsilon)$ for any $\epsilon > 0$.
- Waidner [46], building on [8, 19], shows that the dealer in Bracha's protocol can be replaced by an $\Omega(t)$-round pre-processing phase during which a broadcast channel is assumed. The expected round complexity (after the pre-processing) is also improved from $\Theta(\log n)$ to $\Theta(1)$.

The latter two results assume private channels.

Fitzi and Garay [22], building on [44, 9, 38], give the first full solution to this problem: that is, they show the first authenticated BA protocol with optimal resilience $t < n/2$ and expected constant round complexity that does not require any trusted dealer or pre-processing (other than a PKI). Even assuming private channels, however, their protocol requires specific number-theoretic assumptions (essentially, an appropriate type of homomorphic public-key encryption scheme) and cannot be based on signatures alone. Because of its reliance on additional assumptions, the Fitzi-Garay protocol cannot be adapted to the information-theoretic setting using pseudo-signatures.

2 Model and Defintions

By a *public-key infrastructure* (PKI) in a network of n parties, we mean that all parties hold the same vector $(pk_1, \ldots, pk_n)$ of public keys for a digital signature scheme, and each honest party P_i

[4]Even if an authenticated protocol is proved to be perfectly secure when ideal signatures are available, it will in general only be statistically secure when pseudo-signatures are used as there is always a non-zero probability that an adversary can forge an honest party's signature.

holds the honestly-generated secret key sk_i associated with pk_i. Malicious parties may generate their keys arbitrarily, even dependent on keys of honest parties. We stress that the vector of public keys is assumed to be fixed before any execution of the protocol begins.

Our results are in the *point-to-point* model (unless otherwise stated), by which we mean the standard model in which parties communicate in synchronous rounds using pairwise *private* and *authenticated* channels. Authenticated channels can be realized using signature schemes and a PKI. (Note that signatures and a PKI are stronger than authenticated channels since they allow third-party verifiability.) For static adversaries, private channels can be realized using one additional round by having each party P_i send to each party P_j a public key $PK_{i,j}$ for a semantically-secure public-key encryption scheme (using a different key for each sender avoids issues of malleability). Of course, in the authenticated case such public keys could also be included as part of the PKI. For adaptive adversaries, more complicated solutions are available [1, 11] but we do not discuss these further. For simplicity, we assume *unconditional* private/authenticated channels with the understanding that these guarantees hold only computationally if the above techniques are used.

When we say a protocol (for Byzantine agreement, VSS, etc.) tolerates t malicious parties, we always mean that it is secure against a *rushing* adversary who may *adaptively* corrupt up to t parties during execution of the protocol and coordinate the actions of these parties as they deviate from the protocol in an arbitrary manner. Parties not corrupted by the adversary are called *honest*. Our definitions always implicitly cover both the "unconditional" and "authenticated" cases, in the following way: For $t < n/3$ we allow a computationally-unbounded adversary (this is the unconditional case). For $t < n/2$ we assume a PKI and an adversary who is computationally unbounded but is unable to forge signatures on behalf of any honest party (this is the authenticated case). Using a standard hybrid argument and assuming the existence of digital signatures, this implies that authenticated protocols are secure against computationally-bounded adversaries. Using pseudo-signatures, authenticated protocols can be made secure even against a computationally-unbounded adversary (though in a statistical, rather than perfect, sense).

When we describe signature computation in authenticated protocols we omit for simplicity additional information that should be signed along with the message. That is, when we say that party P_i signs message m and sends it to P_j, we implicitly mean that P_i signs the concatenation of m with additional information such as: (1) the identity of the recipient P_j, (2) the current round number, (3) an identifier for the message (in case multiple messages are sent to P_j in the same round); and (4) an identifier for the (sub-)protocol (in case multiple sub-protocols are being run). This information is also verified, as appropriate, when the signature is verified.

It can be verified by inspection that all protocols constructed in this work remain secure even when run in parallel (in the authenticated case, we rely on the existence of distinct identifiers for each sub-protocol being run so as to avoid the problems highlighted in [36]). One could also prove this formally by showing that each of our protocols realizes the appropriate functionality within the UC framework [10]. Alternately, one could appeal to a recent result showing security of statistically-secure protocols under parallel self composition when inputs are not chosen adaptively [34] (as will always be the case in our work).

When we say that a party sends a message "to all other parties" we also treat this as if it sends the message to itself.

2.1 Broadcast and Byzantine Agreement

Standard definitions of broadcast and Byzantine agreement (BA) follow.

Definition 1 (Broadcast): A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer

$P^* \in \mathcal{P}$ holds an initial input M, is a *broadcast protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Agreement All honest parties output the same value.

Validity If the dealer is honest, then all honest parties output M. $\diamondsuit$

A *broadcast channel* refers to a physical implementation of broadcast; it can be viewed formally as a trusted party to which all parties have access that implements the broadcast functionality as defined above. A broadcast protocol can then be viewed as a protocol that realizes a broadcast channel. These notions can be formalized in the setting of secure multi-party computation by casting broadcast as a functionality (in the natural way) and then proving that any protocol realizing the above definition serves as a secure implementation of the broadcast functionality. When a broadcast channel is used by multiple parties (acting as dealers) in the same round, it is implicitly understood that *rushing* is always allowed and so the message used by a dishonest dealer may depend on the messages used by honest dealers in the same round.

In this paper, we construct protocols for Byzantine agreement which readily imply protocols for broadcast.

Definition 2 (Byzantine agreement): A protocol for parties $P_1, \ldots, P_n$, where each party P_i holds initial input v_i, is a *Byzantine agreement protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Agreement All honest parties output the same value.

Validity If all honest parties begin with the same input value v, then all honest parties output v.

If the $\{v_i\}$ are restricted to binary values, the protocol achieves *binary* Byzantine agreement. $\diamondsuit$

In this work we construct randomized broadcast/BA protocols where termination is not guaranteed in any fixed number of rounds; i.e., for any r there is a non-zero probability that the protocol does not terminate in r rounds. Our protocols do, however, eventually terminate with probability 1, in the sense that the probability of not terminating in r rounds approaches 0 as r tends to infinity. We use the *expected* number of rounds to termination as our measure of round complexity, but remark that our protocols (as in all previous work) achieve something stronger: they terminate in r rounds except with probability $2^{-\Theta(r)}$. In particular, the probability that an execution of the protocol exceeds $\omega(\log k)$ rounds is negligible in k.

3 Byzantine Agreement in Expected Constant Rounds

In this section, we construct expected constant-round protocols for Byzantine agreement in both the unconditional ($t < n/3$) and authenticated ($t < n/2$) settings. Our main result is the protocol for the case $t < n/2$ (which is the first such construction assuming only a PKI and digital signatures); however, we believe our result for the case $t < n/3$ is also interesting as an illustration that our techniques yield a conceptually-simpler and more efficient protocol in that setting as compared to [20]. We develop both protocols in parallel so as to highlight the high-level similarities in each.

Notation: We use "=" to denote a test for equality, and ":=" to denote variable assignment. When we refer to an "honest party" in our proofs and definitions, we mean a party that remains honest until the end of the protocol (or the corresponding phase of the protocol, if the protocol concerned has multiple phases). This distinction is important to keep in mind since we consider adaptive adversaries who may corrupt players in the middle of the protocol execution.

3.1 Basic Primitives

We begin by reviewing the notions of *gradecast* and *VSS*:

Gradecast. *Gradecast*, a relaxed version of broadcast, was introduced by Feldman and Micali [20]; we provide a definition which is weaker than theirs but suffices for our purposes. (In contrast to [20, Def. 11], our definition provides no guarantees even if every honest party P_i outputs $g_i = 1$.)

Definition 3 (Gradecast): A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $P^* \in \mathcal{P}$ holds an initial input M, is a *gradecast protocol tolerating* t *malicious parties* if the following conditions hold for any adversary controlling at most t parties:

- Each honest party P_i outputs a *message* m_i and a *grade* $g_i \in \{0, 1, 2\}$.
- If the dealer is honest, then the output of every honest party P_i satisfies $m_i = M$ and $g_i = 2$.
- If there exists an honest party P_i who outputs a message m_i and the grade $g_i = 2$, then the output of every honest party P_j satisfies $m_j = m_i$ and $g_j \geq 1$. $\diamondsuit$

When we refer to a *gradecast channel* we mean a trusted party that implements gradecast as defined above. (Formally, a dishonest dealer is allowed to specify the outputs of all the honest parties in an arbitrary manner, subject to the restrictions of the above definition; an honest dealer would always elect to have all parties output $(M, 2)$.) When a gradecast channel is used by multiple parties (acting as dealers) in the same round, it is implicitly understood that *rushing* is always allowed and so the message used by a dishonest dealer may depend on the messages used by honest dealers in the same round.

The following result is due to [20] and proved for completeness in Appendix A.1:

Lemma 1 *There exists a constant-round gradecast protocol tolerating $t < n/3$ malicious parties.*

We next prove an analogue of the above for the case of *authenticated* gradecast.

Lemma 2 *There exists a constant-round authenticated gradecast protocol tolerating $t < n/2$ malicious parties.*

Proof The following protocol is adapted from the work of Fitzi and Maurer [24], who were concerned with a different setting than we are here.

Round 1 The dealer computes a signature σ of M and sends (M, σ) to all parties.

Round 2 Let (M_i, σ_i) be the message received by party P_i (from the dealer) in the previous round. If σ_i is a valid signature of M_i (with respect to the dealer's public key), then P_i sends (M_i, σ_i) to all other parties; otherwise P_i sets $M_i := \perp$ and sends nothing.

Round 3 Let $(M_{j,i}, \sigma_{j,i})$ be the message received by P_i (if any) from P_j in the previous round. If there exists a j such that $M_{j,i} \neq M_i$ but $\sigma_{j,i}$ is a valid signature of $M_{j,i}$ (with respect to the dealer's public key), then P_i sets $M_i := \perp$.

If $M_i \neq \perp$, then P_i computes a signature σ'_i of M_i and sends (M_i, σ'_i) to all parties. (If $M_i = \perp$, then P_i sends nothing.)

Round 4 Let $(M'_{j,i}, \sigma'_{j,i})$ be the message received by P_i (if any) from P_j in the previous round. If there exist $\ell \geq n/2$ distinct indices $j_1, \ldots, j_\ell$ and an M^* such that $M'_{j_1,i} = \cdots = M'_{j_\ell,i} = M^*$ and $\sigma'_{j_k,i}$ is a valid signature of M^* (with respect to the public key of P_{j_k}) for $1 \leq k \leq \ell$, then P_i sends $(M^*, j_1, \sigma'_{j_1,i}, \ldots, j_\ell, \sigma'_{j_\ell,i})$ to all other parties and outputs $m_i := M^*$, $g_i := 2$.

Output determination Assuming P_i has not decided on its output, it proceeds as follows: If in the previous round P_i received any message $(M^*, j_1, \sigma'_1, \ldots, j_\ell, \sigma'_\ell)$ for which $\ell \geq n/2$, the $\{j_k\}_{k=1}^{\ell}$ are distinct, and σ'_k is a valid signature of M^* with respect to the public key of party P_{j_k} for $1 \leq k \leq \ell$, then P_i outputs $m_i := M^*$, $g_i := 1$. Otherwise, P_i outputs $m_i :=\perp$, $g_i := 0$.

We show the above protocol satisfies Definition 3. If the dealer is honest, then in round 3 every honest party P_i computes a signature σ'_i of the dealer's message M and sends (M, σ'_i) to all other parties. Thus, all honest parties will receive at least $n/2$ correct signatures on M in round 3, and every honest party P_i will output $m_i = M, g_i = 2$ in round 4.

Before proving the second required property, we first show that no two honest parties P_i, P_j send messages (M_i, σ'_i) and (M_j, σ'_j) in round 3 with $M_i \neq M_j$. To see this, note that in round 3 M_i (resp., M_j) is either equal to $\perp$ or to the message sent by the dealer to P_i (resp., P_j) in the first round. So if the dealer sent a valid signature on the same message to parties P_i, P_j in the first round, the claim is obviously true. On the other hand, in any other case at least one of P_i, P_j will not send any message at all in round 3 (as at least one of $M_i =\perp$ or $M_j =\perp$ will then hold).

Say a value M^* is *certified* if an honest player holds $(M^*, j_1, \sigma'_1, \ldots, j_\ell, \sigma'_\ell)$ with $\ell \geq n/2$, distinct $\{j_k\}_{k=1}^{\ell}$, and σ'_k a valid signature of M^* with respect to the public key of party P_{j_k} for $1 \leq k \leq \ell$. Note that any certified value was signed by at least one honest party. Since any honest parties who sign a message in round 3 sign the *same* message, as argued in the previous paragraph, it follows that at most one value is certified.

Now, say there is an honest party P_i who outputs some message m_i and $g_i = 2$. It follows easily that any honest party P_j who did not output $g_j = 2$ immediately in round 4 will output $g_j = 1$ (and hence we have $g_j \geq 1$). Since, as we have just argued, at most one value can be certified, it follows that all honest parties output m_i. ■

Verifiable Secret Sharing (VSS). *VSS* [13] extends the concept of secret sharing [7, 43] in the sense that it considers the presence of malicious, rather than just honest-but-curious, parties.

Definition 4 (Verifiable secret sharing): A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $P^* \in \mathcal{P}$ holds initial input s, is a *VSS protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Validity Each honest party P_i outputs a value s_i at the end of the second phase (the *reconstruction phase*). Furthermore, if the dealer is honest then $s_i = s$.

Secrecy If the dealer is honest during the first phase (the *sharing phase*), then at the end of this phase the joint view of the malicious parties is independent of the dealer's input s.

Reconstruction At the end of the sharing phase the joint view of the honest parties defines a value s' (which can be computed in polynomial time from this view) such that all honest parties will output s' at the end of the reconstruction phase. ♢

Note that, by definition, VSS is not possible when $t \geq n/2$.

The first result that follows is well-known; the second is not explicit in the literature but follows readily from known results. For completeness, proofs appear in Appendices A.2 and A.3.

Lemma 3 *There exists a constant-round VSS protocol tolerating $t < n/3$ malicious parties, and relying on a broadcast channel during the sharing phase only.*

Lemma 4 *There exists a constant-round authenticated VSS protocol tolerating $t < n/2$ malicious parties, and relying on a broadcast channel during the sharing phase only.*

3.2 Moderated VSS

We introduce a variant of VSS called *moderated VSS*, in which there is a distinguished party (who may be identical to the dealer) called the *moderator*. Roughly speaking, the moderator "simulates" a broadcast channel for the other parties during the sharing phase. At the end of the sharing phase, parties output a boolean flag indicating whether or not they trust the moderator. If the moderator is honest, all honest parties set this flag to 1. Furthermore, if any honest party sets this flag to 1 then the protocol achieves all the properties of VSS (cf. Definition 4). A formal definition follows.

Definition 5 (Moderated VSS): A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where there is a distinguished dealer $P^* \in \mathcal{P}$ who holds an initial input s and a moderator $P^{**} \in \mathcal{P}$ (who may possibly be the dealer), is a *moderated VSS protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

- Each honest party P_i outputs a bit f_i at the end of the first phase (called the *sharing phase*), and a value s_i at the end of the second phase (called the *reconstruction phase*).
- If the moderator is honest during the sharing phase, then each honest party P_i outputs $f_i = 1$ at the end of this phase.
- If there exists an honest party P_i who outputs $f_i = 1$ at the end of the sharing phase, then the protocol achieves VSS; specifically: (1) if the dealer is honest then all honest parties output s at the end of the reconstruction phase, and the joint view of all the malicious parties at the end of the sharing phase is independent of s, and (2) the joint view of the honest parties at the end of the sharing phase defines an efficiently-computable value s' such that all honest parties output s' at the end of the reconstruction phase. $\diamondsuit$

We stress that if all honest parties P_i output $f_i = 0$ at the end of the sharing phase, then no guarantees are provided; e.g., honest parties may output different values at the end of the reconstruction phase, or the malicious parties may learn the dealer's secret in the sharing phase.

The main result of this section is the following:

Theorem 5 *Assume there exists a constant-round VSS protocol Π, using a broadcast channel in the sharing phase only, tolerating $t < n/2$ malicious parties. Then there exists a constant-round moderated VSS protocol Π', using a gradecast channel, tolerating t malicious parties.*

Proof We show how to "compile" Π so as to obtain the desired Π'. Essentially, Π' is constructed by replacing each broadcast in Π with two invocations of gradecast: one by the party who is supposed to broadcast the message, and one by the moderator P^{**}. In more detail, Π' is defined as follows: At the beginning of the protocol, all parties set their flag f to 1. The parties then run an execution of Π. When a party P is directed by Π to send message m to P', it simply sends this message. When a party P is directed by Π to broadcast a message m, the parties run the following "broadcast emulation" subroutine:

1. P gradecasts the message m.
2. The moderator P^{**} gradecasts the message it output in the previous step.
3. Let (m_i, g_i) and (m'_i, g'_i) be the outputs of party P_i in steps 1 and 2, respectively. Within the underlying execution of Π, party P_i will use m'_i as the message "broadcast" by P.
4. Furthermore, P_i sets $f_i := 0$ if either (or both) of the following conditions hold: (1) $g'_i \neq 2$, or (2) $m'_i \neq m_i$ and $g_i = 2$.

Party P_i outputs f_i at the end of the sharing phase, and outputs whatever it is directed to output by Π at the end of the reconstruction phase.

We now prove that Π' is a moderated VSS protocol tolerating t malicious parties. We first show that if the moderator is honest during the sharing phase then no honest party P_i ever sets $f_i := 0$. To see this, note that if P^{**} is honest then $g'_i = 2$ each time the broadcast emulation subroutine is executed. Furthermore, if P_i outputs some m_i and $g_i = 2$ in step 1 of that subroutine then, by definition of gradecast, P^{**} also outputs m_i in step 1. Hence $m'_i = m_i$ and f_i remains 1.

To show the second required property of moderated VSS, consider any execution of the broadcast emulation subroutine. We show that if there exists an honest party P_i who holds $f_i = 1$ upon completion of that subroutine, then the functionality of broadcast was achieved (in that execution of the subroutine). It follows that if P_i holds $f_i = 1$ at the end of the sharing phase, then Π' provided a faithful execution of all broadcasts in Π and so the functionality of VSS is achieved.

If P_i holds $f_i = 1$, then $g'_i = 2$. (For the remainder of this paragraph, all variables are local to a particular execution of the broadcast subroutine.) Since $g'_i = 2$, the properties of gradecast imply that any honest party P_j holds $m'_j = m'_i$ and so all honest parties agree on the message that was "broadcast." Furthermore, if the "dealer" P (in the broadcast emulation subroutine) is honest then $g_i = 2$ and m_i is equal to the dealer's intended message m. So the fact that $f_i = 1$ means that $m'_i = m_i = m$, and so all honest parties use the message m "broadcast" by P in their underlying execution of Π. ■

The transformation used in the proof of the preceding theorem does not apply to VSS protocols that use broadcast in the reconstruction phase. We did not attempt to extend the theorem to such protocols since we do not need such an extension for our results.

By applying the above theorem to the VSS protocol of Lemma 3 (resp., the authenticated VSS protocol of Lemma 4) and then using the gradecast protocol of Lemma 1 (resp., the authenticated gradecast protocol of Lemma 2) to instantiate[5] the gradecast channel, we obtain:

Corollary 6 *There exists a constant-round protocol for moderated VSS (in the point-to-point model) tolerating $t < n/3$ malicious parties.*

Corollary 7 *There exists a constant-round protocol for authenticated moderated VSS (in the point-to-point model) tolerating $t < n/2$ malicious parties.*

Moderated VSS vs. graded VSS. We briefly discuss the differences between moderated VSS and graded VSS (the key component in the Feldman-Micali protocol [20]). We first quickly recall the definition of graded VSS. Like VSS, graded VSS has two phases: the sharing phase and the reconstruction phase. At the end of the sharing phase, each party outputs a grade in $\{0, 1, 2\}$. A graded VSS protocol satisfies the following three requirements:

- If the dealer is honest, then all honest parties output grade 2.
- If there exists an honest party that outputs a grade ≥ 1, then graded VSS achieves VSS.
- If there exists an honest party that outputs grade 2, then all honest parties output a grade ≥ 1.

Essentially, the grade output by a party in graded VSS indicates the extent to which the party "trusts" the dealer. In moderated VSS we shift this determination of "trust" in the dealer to a determination of "trust" in a (possibly) *different* party, the moderator. This conceptual separation

[5]It is important here for the claimed round complexity that multiple executions of gradecast, with different parties acting as dealer, can be run in parallel. One can easily verify that this is the case for the protocols presented here.

is the key to our simpler construction of a BA protocol for $t < n/3$, as well as our construction of an authenticated BA protocol for $t < n/2$.

Feldman and Micali construct a constant-round graded VSS protocol for $t < n/3$ based on the VSS protocol of Ben-Or et al. [5] by replacing each invocation of broadcast in the underlying VSS protocol by a gradecast. The grade output in the graded VSS protocol by each honest party is then determined as a (complicated) function of the grades received in each individual gradecast. It is not clear how a constant-round protocol for graded VSS could be constructed directly for $t < n/2$ (though it appears that one could be constructed from a moderated VSS protocol with the same threshold), or how such a protocol could be used to construct a constant-round BA protocol for the same threshold of malicious parties.

3.3 From Moderated VSS to Oblivious Leader Election

In this section, we construct an oblivious leader election (OLE) protocol based on any moderated VSS protocol. The following definition of oblivious leader election is adapted from [22]:

Definition 6 (Oblivious leader election): A two-phase protocol for parties $P_1, \ldots, P_n$ is an *oblivious leader election protocol with fairness δ tolerating t malicious parties* if each honest party P_i outputs a value $v_i \in [n]$ at the end of the second phase, and the following condition holds with probability at least δ (over random coins of the honest parties) for any adversary controlling at most t parties:

> There exists a $j \in [n]$ such that (1) each honest party P_i outputs $v_i = j$, and (2) P_j was honest at the end of the first phase.[6]

If the above event happens, then we say *an honest leader was elected.* $\diamondsuit$

It may seem counter-intuitive that OLE can be useful in the presence of an adaptive adversary who can anyway corrupt the leader as soon as the leader is elected. In our application of OLE, however, each party will send some value to all other parties *prior to* the second phase of OLE. Then, after completion of the second phase of OLE, all parties use the value that was sent previously by the elected leader. Since all parties have already sent their values prior to completion of the first phase, it does not matter whether the leader is subsequently corrupted after completion of the second phase.

Our construction of OLE uses a similar high-level approach as in the construction of an oblivious common coin from graded VSS [20]. However, we introduce different machinery and start from *moderated* VSS. Intuitively, a random coin $c_j \in [n^4]$ is generated for each party P_j. This is done by having each party P_i select a random contribution $c_{i,j} \in [n^4]$, and then this value is shared using moderated VSS with P_j acting as moderator. The $c_{i,j}$ are later reconstructed and c_j is computed as $c_j := \sum_i c_{i,j} \bmod n^4$. An honest party then outputs j minimizing c_j. Since moderated VSS (instead of VSS) is used, each party P_k may have a different view regarding the $\{c_j\}$. However:

- If P_j is honest then (by the properties of moderated VSS) all honest parties reconstruct the same values $c_{i,j}$ (for any i) and hence compute an identical value for c_j.
- If P_j is dishonest but there exists an honest party P_i such that P_i outputs $f_i = 1$ in all invocations of moderated VSS where P_j acts as the moderator, then (by the properties of moderated VSS) all honest parties compute an identical value for c_j.

Relying on the above observations, we devise a way such that all honest parties output the same j (such that P_j is furthermore honest) with constant probability.

[6]Note that we cannot simply require that P_j is honest since an adaptive adversary can always corrupt the leader once it has been elected. It is also for this reason that we separate the protocol into two phases (if a non-adaptive adversary is considered, the conceptual separation of the protocol into two phases is not necessary).

Theorem 8 *Assume there exists a constant-round moderated VSS protocol tolerating t malicious parties. Then there exists a constant-round OLE protocol with fairness $\delta = \frac{n-t}{n} - \frac{1}{n^2}$ tolerating t malicious parties. Specifically, if $n \geq 3$ and $t < n/2$ then $\delta \geq 1/2$.*

Proof An OLE protocol follows. Each P_i begins with $\mathsf{trust}_{i,j} = 1$ for $j \in \{1, \ldots, n\}$. Then:

Phase 1 Each party P_i chooses random $c_{i,j} \in [n^4]$ for $1 \leq j \leq n$. The following is executed n^2 times in parallel for each ordered pair (i, j):

> All parties execute the sharing phase of a moderated VSS protocol in which P_i acts as the dealer with initial input $c_{i,j}$, and P_j acts as the moderator. If a party P_k outputs $f_k = 0$ in this execution, then P_k sets $\mathsf{trust}_{k,j} := 0$.

Upon completion of the above, let $\mathsf{trust}_k \overset{\text{def}}{=} \{j : \mathsf{trust}_{k,j} = 1\}$.

Phase 2 The reconstruction phase of the moderated VSS protocol is run n^2 times in parallel to reconstruct the secrets previously shared. Let $c^k_{i,j}$ denote P_k's view of the value of $c_{i,j}$. (If a reconstructed value lies outside $[n^4]$, then $c^k_{i,j}$ is assigned some default value in the correct range.) Each party P_k sets $c^k_j := \sum_{i=1}^n c^k_{i,j} \bmod n^4$, and outputs $j \in \mathsf{trust}_k$ that minimizes c^k_j.

We prove that the protocol satisfies Definition 6. Following execution of the above, define:

$$\mathsf{trusted} \overset{\text{def}}{=} \left\{ j : \begin{array}{c} \text{there exists a } P_k \text{ that was honest at the end of phase 1} \\ \text{for which } j \in \mathsf{trust}_k \end{array} \right\}.$$

It is immediate that if P_j was honest in phase 1, then $j \in \mathsf{trust}_k$ for all honest P_k (and hence $j \in \mathsf{trusted}$). Furthermore, the following hold by the properties of moderated VSS:

1. If $j \in \mathsf{trusted}$, then for any honest $P_k, P_{k'}$ and any $1 \leq \ell \leq n$, we have $c^k_{\ell,j} = c^{k'}_{\ell,j}$ and hence $c^k_j = c^{k'}_j$; thus, we may freely omit the superscript in this case.
2. If $j \in \mathsf{trusted}$ and P_i is honest, then $c_{i,j}$ is independent of the view of the malicious parties at the end of phase 1.

We next claim that for $j \in \mathsf{trusted}$ the coin c_j is uniformly distributed in $[n^4]$. To see this, let $c'_j = \sum_{\ell : P_\ell \text{ malicious in phase 1}} c_{\ell,j} \bmod n^4$ and let $c''_j = \sum_{\ell : P_\ell \text{ honest in phase 1}} c_{\ell,j} \bmod n^4$. (The former is the contribution to c_j by the parties that are malicious in phase 1; the latter is the contribution to c_j by the parties that are honest in phase 1.) Since $j \in \mathsf{trusted}$, the properties of VSS hold for the $\{c_{\ell,k}\}_{\ell=1}^n$ and thus c'_j is well-defined at the end of phase 1; moreover, by the second property noted earlier, c''_j is independent of the view of the malicious parties at the end of phase 1. Because c''_j is uniform in $[n^4]$, it follows that $c_j = c'_j + c''_j \bmod n^4$ is uniformly distributed in $[n^4]$.

By the union bound, with probability at least $1 - \frac{1}{n^2}$ all coins $\{c_j : j \in \mathsf{trusted}\}$ are distinct. Conditioned on this event, with probability at least $\frac{n-t}{n}$ the party with the minimum c_j among the set $\mathsf{trusted}$ corresponds to a party that was honest in phase 1. This concludes the proof. ■

Combining Theorem 8 with Corollaries 6 and 7, we obtain:

Corollary 9 *There exists a constant-round protocol for OLE with fairness $2/3$ tolerating $t < n/3$ malicious parties. (Note that when $n < 4$ the result is trivially true.)*

Corollary 10 *There exists a constant-round protocol for authenticated OLE with fairness $1/2$ tolerating $t < n/2$ malicious parties. (Note that when $n < 3$ the result is trivially true.)*

3.4 From OLE to Byzantine Agreement

For the unauthenticated case (i.e., $t < n/3$), Feldman and Micali [20] show how to construct an expected constant-round binary Byzantine agreement protocol based on any constant-round *oblivious common coin* protocol. We construct a more round-efficient protocol based on oblivious leader election. (Another advantage is that this approach allows us to handle parallel composition more easily; see Section 4.) This also serves as a warmup for the authenticated case.

Theorem 11 *If there exists a constant-round OLE protocol with fairness $\delta = \Omega(1)$ tolerating $t < n/3$ malicious parties, then there exists an expected constant-round binary Byzantine agreement protocol tolerating t malicious parties.*

Proof We describe a protocol for binary Byzantine agreement, assuming the existence of an OLE protocol with constant fairness tolerating $t < n/3$ malicious parties. Each party P_i uses local variables $b_i \in \{0, 1\}$ (which is initially P_i's input), $\texttt{update}_i$ (initially set to $\texttt{true}$), and $\texttt{exitBA}_i$ (initially set to $\texttt{false}$). At a high level, the parties run the following six steps, called an *iteration*, until agreement is reached. The variable $\texttt{update}_i$ indicates whether P_i should update its value of b_i to the value sent by the leader elected in that iteration, and the variable $\texttt{exitBA}_i$ indicates whether P_i should terminate at the end of the current iteration.

Step 1 Each P_i sends b_i to all parties. Let $b_{j,i}$ be the bit P_i receives from P_j. (When this step is run at the outset of the protocol, a default value is used if P_i does not receive anything from P_j. In subsequent iterations, if P_i does not receive anything from P_j then P_i leaves $b_{j,i}$ unchanged.)

Step 2 Each party P_i sets $\mathcal{S}_i^b := \{j : b_{j,i} = b\}$ for $b \in \{0, 1\}$. If $|\mathcal{S}_i^0| \geq t + 1$, then P_i sets $b_i := 0$. If $|\mathcal{S}_i^0| \geq n - t$, then P_i sets $\texttt{exitBA}_i := \texttt{true}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 3 Each party P_i defines $\mathcal{S}_i^b$ as in step 2. If $|\mathcal{S}_i^1| \geq t + 1$, then P_i sets $b_i := 1$. If $|\mathcal{S}_i^1| \geq n - t$, then P_i sets $\texttt{exitBA}_i := \texttt{true}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

If $\texttt{exitBA}_i = \texttt{true}$, then P_i sets $\texttt{update}_i := \texttt{false}$.

Step 4 Each party P_i defines $\mathcal{S}_i^b$ as in step 2. If $|\mathcal{S}_i^0| \geq t + 1$, then P_i sets $b_i := 0$. If $|\mathcal{S}_i^0| \geq n - t$, then P_i sets $\texttt{update}_i := \texttt{false}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 5 Each party P_i defines $\mathcal{S}_i^b$ as in step 2. If $|\mathcal{S}_i^1| \geq t + 1$, then P_i sets $b_i := 1$. If $|\mathcal{S}_i^1| \geq n - t$, then P_i sets $\texttt{update}_i := \texttt{false}$.

Each P_i sends b_i to all parties. If P_i receives a bit from P_j, then P_i sets $b_{j,i}$ to that value; otherwise, $b_{j,i}$ remains unchanged.

Step 6 All parties execute the OLE protocol; let ℓ_i be the output of P_i. Each P_i does the following: if $\texttt{update}_i = \texttt{true}$, then P_i sets $b_i := b_{\ell_i,i}$. If $\texttt{exitBA}_i = \texttt{true}$, then P_i outputs b_i and terminates; otherwise, P_i goes to step 1.

First we claim that if an honest P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 2 or 3 of some iteration, then all honest parties P_j hold $b_j = b_i$ by the end of step 3 of that same iteration. Consider the case when P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 2. (The case when P_i sets $\texttt{exitBA}_i := \texttt{true}$ in step 3 is exactly analogous.) This implies that $|S_i^0| \geq n - t$ and hence $|S_j^0| \geq n - 2t \geq t + 1$ and $b_j = 0$ by the end of step 2. Since this holds for all honest players P_j, it follows that in step 3 we have $|S_j^1| \leq t$ and so b_j remains 0.

Next, we show that if — immediately prior to any given iteration — no honest parties have terminated and there exists a bit b such that $b_i = b$ for all honest P_i, then by the end of step 3 of that iteration all honest parties P_i hold $b_i = b$ and $\texttt{exitBA}_i = \texttt{true}$. Consider the case $b = 0$ (the case $b = 1$ is exactly analogous). In this case $|\mathcal{S}_i^0| \geq n - t$ in step 2 for any honest P_i. Thus, any honest P_i sets $\texttt{exitBA}_i := \texttt{true}$ and holds $b_i = 0$ by the end of this step. As in the previous paragraph, in step 3 the value of b_i remains 0.

Arguing exactly as in the previous two paragraphs, one can similarly show: (i) if — immediately prior to any given iteration — there exists a bit b such that $b_i = b$ for all honest P_i, then by the end of step 5 of that iteration all honest parties P_i hold $b_i = b$, $\texttt{exitBA}_i = \texttt{true}$, and $\texttt{update}_i = \texttt{false}$ (and hence all active, honest parties output b and terminate the protocol in that iteration). (ii) If an honest party P_i sets $\texttt{exitBA}_i := \texttt{true}$ in some iteration, then all honest parties P_j hold $b_j = b_i$ and $\texttt{update}_j = \texttt{false}$ by the end of step 5 of that iteration. (iii) If an honest party P_i sets $\texttt{update}_i := \texttt{false}$ in some iteration, then all honest parties P_j hold $b_j = b_i$ by the end of step 5 of that same iteration. Note that validity is implied by (i).

We next show that if an honest party P_i outputs $b_i = b$ (and terminates) in some iteration, then all honest parties output b and terminate by the end of the next iteration. To see this, note that if P_i terminates in some iteration then it must have set $\texttt{exitBA} := \texttt{true}$ in that same iteration; by (ii) every honest party P_j holds $b_j = b$ at the end of that iteration, and sent $b_j = b$ to all other parties in step 5. (Thus, even if P_j terminates in that iteration, P_j can be viewed as continuing to send $b_j = b$ in the following iteration.) So by (i), all honest parties who have not yet terminated will output b at the end of the next iteration.

Finally, we show that if an honest leader[7] P_ℓ is elected in step 6 of some iteration, then all honest parties P_i terminate by the end of the next iteration. By (i), it is sufficient to show that, by the end of that iteration, there exists a b with $b_i = b$ for all honest P_i. If all honest P_j hold $\texttt{update}_j = \texttt{true}$ then this is immediate (since all honest parties then set $b_i := b_{\ell,i} = b_\ell$, using the fact that P_ℓ was honest in step 5). Otherwise, say honest P_i holds $\texttt{update}_i = \texttt{false}$. By (iii), $b_\ell = b_i$ at the end of step 5, and hence all honest parties P_j have $b_j = b_i$ by the end of step 6.

If the OLE protocol elects an honest leader with constant probability, it follows that the above protocol is an expected constant-round binary Byzantine agreement protocol. ■

When $t < n/3$, any binary Byzantine agreement protocol can be transformed into a (multi-valued) Byzantine agreement protocol using two additional rounds [45]. Parallel composition (see Section 4) can also be used to achieve the same result without using any additional rounds. Using either approach in combination with the above and Corollary 9, we have:

Corollary 12 *There exists an expected constant-round protocol for Byzantine agreement (and hence also broadcast) tolerating $t < n/3$ malicious parties.*

For the authenticated case (i.e., $t < n/2$), Fitzi and Garay [22] construct a binary Byzantine agreement protocol based on OLE; however, they do not explicitly describe how to achieve termination. We construct a *multi-valued* Byzantine agreement protocol based on OLE and explicitly show

[7]This implies that P_ℓ was uncorrupted in step 5 of the iteration in question.

how to achieve termination. In the proof below, we assume the existence of a gradecast channel for simplicity of exposition; in Section 5.2, we improve the round complexity of our protocol by working directly in the point-to-point model.

Theorem 13 *If there exists a constant-round authenticated OLE protocol with fairness $\delta = \Omega(1)$ tolerating $t < n/2$ malicious parties, then there exists an expected constant-round authenticated Byzantine agreement protocol, assuming a gradecast channel, tolerating t malicious parties.*

Proof Let V be the domain of possible input values, let the input value for party P_i be $v_i \in V$, and let $\phi \in V$ be some default value. Each P_i begins with an internal variable $\texttt{lock}_i$ set to ∞.

Step 1 Each P_i gradecasts v_i. Let $(v_{j,i}, g_{j,i})$ be the output of P_i in the gradecast by P_j.

Step 2 For any v such that $v = v_{j,i}$ for some j, party P_i sets $\mathcal{S}_i^v := \{j : v_{j,i} = v \wedge g_{j,i} = 2\}$ and $\tilde{\mathcal{S}}_i^v := \{j : v_{j,i} = v \wedge g_{j,i} \geq 1\}$. If $\texttt{lock}_i = \infty$, then:

1. If there exists a v such that $|\tilde{\mathcal{S}}_i^v| > n/2$, then set $v_i := v$; otherwise, set $v_i := \phi$.
2. If $|\mathcal{S}_i^{v_i}| > n/2$, then set $\texttt{lock}_i := 1$.

Step 3 Each P_i gradecasts v_i. Let $(v_{j,i}, g_{j,i})$ be the output of P_i in the gradecast by P_j.

Step 4 For any v such that $v = v_{j,i}$ for some j, party P_i defines $\mathcal{S}_i^v$ and $\tilde{\mathcal{S}}_i^v$ as in step 2. If $\texttt{lock}_i = \infty$, then:

- If there exists a v such that $|\tilde{\mathcal{S}}_i^v| > n/2$, then set $v_i := v$; otherwise, set $v_i := \phi$.

P_i sends v_i to all parties. Let $v_{j,i}^*$ be the value P_i receives from P_j.

Step 5 All parties execute the OLE protocol; let ℓ_i be the output of P_i.

Step 6 Each P_i does the following: if $\texttt{lock}_i = \infty$ and $|\mathcal{S}_i^{v_i}| \leq n/2$, then set $v_i := v_{\ell_i,i}^*$.

Step 7 If $\texttt{lock}_i = 0$, then P_i outputs v_i and terminates the protocol. If $\texttt{lock}_i = 1$, then P_i sets $\texttt{lock}_i := 0$ and goes to step 1. If $\texttt{lock}_i = \infty$, then P_i goes to step 1.

We refer an execution of steps 1 through 7 as an *iteration*. An easy observation is that once an honest party P_i holds $\texttt{lock}_i \neq \infty$, then (assuming P_i remains honest) v_i is unchanged for the remainder of the protocol and P_i terminates with output v_i by (at latest) the end of the following iteration. We first claim that if — immediately prior to any given iteration — there exists a value v such that $v_i = v$ for all honest P_i and no honest parties have yet terminated, then all honest parties will terminate and output v by the end of the following iteration. (This in particular proves validity.) To see this, note that in this case all honest parties gradecast v in step 1. By the properties of gradecast, all honest parties will be in $\mathcal{S}_i^v$ and $\tilde{\mathcal{S}}_i^v$ for any honest P_i. It follows that $v_i = v$ and $\texttt{lock}_i \neq \infty$ for all honest P_i after step 2. The observation mentioned earlier shows that all honest parties will terminate within at most one additional iteration with output v, as desired.

Now consider the first iteration in which an honest party P_i sets $\texttt{lock}_i := 1$ (in step 2), and let $v \stackrel{\text{def}}{=} v_i$. We claim that, by the end of that iteration, $v_j = v$ for all honest P_j and no honest parties will have yet terminated. The claim regarding termination is immediate since honest parties do not terminate until the iteration following the one in which they set $\texttt{lock} := 1$ (and we are considering the first such iteration). As for the first property, note that by the properties of gradecast $\mathcal{S}_i^v \subseteq \tilde{\mathcal{S}}_j^v$. Since P_i set $\texttt{lock}_i := 1$ we know that $|\tilde{\mathcal{S}}_j^v| \geq |\mathcal{S}_i^v| > n/2$ and so P_j sets $v_j = v$ after step 2. Since this holds for all honest parties, every honest party gradecasts v in step 3 and thus $|\mathcal{S}_j^v| > n/2$ in step 4. It follows that $v_j = v$ at the end of that iteration.

Combining the above arguments, it follows that if an honest P_i sets $\texttt{lock}_i := 1$ in some iteration I, then $v_j = v_i$ for all honest P_j at the end of iteration I; all honest P_j have $\texttt{lock}_j \neq \infty$ by the end of iteration $I + 1$; and all honest parties will have terminated with identical outputs by the end of iteration $I + 2$.

We next show that if an honest leader[8] P_ℓ is elected in some iteration then all honest P_i hold the same value v by the end of that iteration. By what we have argued above, it suffices to consider the case where $\texttt{lock}_i = \infty$ for all honest parties P_i after step 2. If in step 6 all honest P_i have $|\mathcal{S}_i^{v_i}| \leq n/2$, it follows easily that each honest P_i sets $v_i := v^*_{\ell,i} = v_\ell$ (using the fact that P_ℓ was honest in step 4) and so all honest parties hold the same value v_i at the end of step 6. So, say there exists an honest party P_i such that $|\mathcal{S}_i^{v_i}| > n/2$ in step 6. Consider any other honest party P_j:

- If $|\mathcal{S}_j^{v_j}| > n/2$, then $\mathcal{S}_i^{v_i} \cap \mathcal{S}_j^{v_j} \neq \emptyset$ and (by properties of gradecast) $v_j = v_i$.
- If $|\mathcal{S}_j^{v_j}| \leq n/2$, then P_j sets $v_j := v^*_{\ell,j}$. But, by properties of gradecast, $\mathcal{S}_i^{v_i} \subseteq \tilde{\mathcal{S}}_\ell^{v_i}$ and so P_ℓ set $v_\ell := v_i$ in step 4 (since P_ℓ was honest at that point). Hence P_j sets $v_j := v^*_{\ell,j} = v_\ell = v_i$.

If the OLE protocol elects an honest leader with constant probability, it follows that the above protocol terminates in expected constant rounds. ■

Combining Lemma 2, Corollary 10, and Theorem 13 we obtain our main result:

Corollary 14 *There exists an expected constant-round protocol for authenticated Byzantine agreement (and hence also for authenticated broadcast) tolerating $t < n/2$ malicious parties.*

Exact round complexities. In Section 5, we compute the exact round complexities of our protocols for BA and authenticated BA (after applying some optimizations). We also show ways of reducing the *amortized* round complexities when multiple (sequential) executions of our protocols are run; this is important when they are used as sub-routines to implement a broadcast channel within some larger protocol (cf. the discussion on sequential composition in the following section).

Communication complexity and computational overhead. In what follows, we estimate the communication complexities of our protocols, as well as the number of signature computations required in the authenticated case. As the focus of this work was on round complexity, we did not make any attempt to improve or optimize these results.

Unconditional case The gradecast protocol given in Appendix A.1 has total communication complexity $O(n^2|M|)$ to gradecast a message of length $|M|$. The communication complexity of moderated VSS, as given by applying Theorem 5 to the VSS protocol from Appendix A.2, is $O(n^4|s|)$ to share a secret of length $|s|$. Oblivious leader election requires $O(n^2)$ invocations of moderated VSS, where a secret of length $O(\log n)$ is shared, for a total communication complexity of $O(n^6 \log n)$. This dominates the cost of each iteration of the binary Byzantine agreement protocol described in Theorem 11; since an expected constant number of iterations are run, the expected communication complexity is $O(n^6 \log n)$. Multi-valued Byzantine agreement on a value of length $|M|$ can be obtained using only $O(n^2|M|)$ additional communication [45].

Authenticated case Let k denote the length of a signature. The gradecast protocol given in the proof of Lemma 2 has total communication complexity $O(n^2|M| + n^3(k + \log n))$ to gradecast a message of length $|M|$. Furthermore, in total $O(n)$ signatures are generated and $O(n^2)$ signatures are verified. The communication complexity of moderated VSS, as given by applying

[8]This implies in particular that P_ℓ was uncorrupted in step 4 of the iteration in question.

Theorem 5 to the VSS protocol from Appendix A.3, is $O(n^4 \cdot (|s| + k + \log n))$ for sharing a secret of length $|s|$. Furthermore, a total of $O(n^2)$ signatures are generated and $O(n^3)$ signatures are verified. Oblivious leader election requires $O(n^2)$ invocations of moderated VSS, where a secret of length $O(\log n)$ is shared, for a total communication complexity of $O(n^6 \cdot (k + \log n))$, along with $O(n^4)$ signature computations and $O(n^5)$ signature verifications.

The cost of each iteration of the Byzantine agreement protocol described in Theorem 13 is dominated by the oblivious leader election and gradecast sub-routines. Thus, the expected communication complexity is $O(n^3|M| + n^6 \cdot (k + \log n))$ for agreement on a value of length $|M|$, while there are $O(n^4)$ signature computations and $O(n^5)$ signature verifications.

In Section 5.2 we show a Byzantine agreement protocol with expected communication complexity $O(n^2|M| + n^6 \cdot (k + \log n))$ (and the same number of signature computations as above).

4 Parallel and Sequential Composition

4.1 Composition of Broadcast Sub-Routines

Suppose we are given an n-party protocol Π that is designed under the assumption that a broadcast channel exists. Without loss of generality, assume that every party broadcasts in every round of Π. (We stress, however, that rushing is allowed and so the message broadcast by a dishonest party in a given round may depend on the messages broadcast by the honest parties in that round.) We would like to compile Π so as to obtain a protocol Π' running in a point-to-point network, with only an expected constant multiplicative increase in the round complexity. The naive way to do this is to simply replace every execution of broadcast in Π with an execution of an expected constant-round broadcast protocol bc. There are, however, two subtle problems with this approach:

Parallel composition Although the expected round complexity of any single execution of bc is constant, the expected number of rounds needed for n parallel executions of bc to all terminate may no longer be constant.

Sequential composition Protocol bc does not provide simultaneous termination. (As noted by Lindell et al. [37], this is inherent for any broadcast protocol using $o(t)$ rounds.) The lack of simultaneous termination may cause problems for subsequent executions of bc, since all honest parties may not begin running some subsequent execution of bc in the same round. It may also cause problems for messages sent as part of Π itself.

The problem of parallel composition is rather easy to deal with. A general technique for handling this issue is proposed by [4], though their solution is somewhat complicated. For the specific broadcast protocol developed here, however, we may rely on an idea of Fitzi and Garay [22] that applies to any broadcast protocol based on oblivious leader election (as ours are). The main idea is that when multiple broadcast sub-routines are run in parallel, only a *single* leader election (per iteration) is required for all these sub-routines. Using this approach, the expected round complexity for n parallel executions is identical to the expected round complexity of a single execution.

In what follows, then, we will assume we have an expected constant-round protocol pbc achieving *parallel broadcast*, defined as follows:

Definition 7 (Parallel broadcast): A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each party P_i holds an initial input M_i, is a *parallel broadcast protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

Functionality Each honest party P_j outputs a vector of values $(m_1^j, \ldots, m_n^j)$.

Agreement All honest parties output the same vector.

Validity If P_i and P_j are any two honest parties, then $m_i^j = M_i$. $\diamondsuit$

We stress that "parallel broadcast"is *not* the same as "simultaneous broadcast" [31], since the latter requires independence of the messages used by the different parties.

4.2 Sequential Composition

We have already noted that certain difficulties arise due to sequential composition of protocols without simultaneous termination (see also [37]). As an example of what can go wrong, assume some protocol Π (that relies on a broadcast channel) requires all parties to broadcast messages in rounds 1 and 2. Let $\mathsf{pbc}_1, \mathsf{pbc}_2$ denote the corresponding invocations of parallel broadcast within the composed protocol Π^{pbc} (which runs in a point-to-point network, and in which each invocation of parallel broadcast is emulated by running a parallel broadcast protocol pbc). Then, because honest parties in pbc_1 do not terminate in the same round, honest parties may begin execution of pbc_2 in different rounds. But security of pbc_2 is no longer guaranteed in this case!

Of course, one way to deal with this issue is to design protocols that use broadcast in only a *single* round; see [32, 33] for work in this direction. In this section, however, we focus on general techniques that can be applied to arbitrary protocols (that may use broadcast in multiple rounds).

Existing methods for dealing with sequential composition of protocols without simultaneous termination either apply only in the case $t < n/3$ [4] or are rather inefficient [37]. We show here a more efficient way to deal with the problem that applies for $t < n/2$.

Let $\mathsf{rc}(\Pi)$ denote the (expected) round complexity of a protocol Π.[9] We also define:

Definition 8 A protocol Π has *staggering gap* $g = \mathsf{gap}(\Pi)$ if any honest parties P_i, P_j are guaranteed to terminate Π within g rounds of each other.

We now prove the following:

Lemma 15 *Let* pbc *be a parallel broadcast protocol tolerating* t *malicious parties. Then for any constant* $c \geq 0$ *there is a parallel broadcast protocol* $\mathsf{Expand}_c(\mathsf{pbc})$ *tolerating* t *malicious parties as long as all honest parties begin execution of* $\mathsf{Expand}_c(\mathsf{pbc})$ *within* c *rounds of each other. The staggering gap of* $\mathsf{Expand}_c(\mathsf{pbc})$ *is 1 (as long as all honest parties begin execution within* c *rounds of each other), and furthermore*

$$\mathsf{rc}\left(\mathsf{Expand}_c(\mathsf{pbc})\right) \leq (2c+1) \cdot \mathsf{rc}(\mathsf{pbc}) + c + 2,$$

where the round complexity (when honest parties begin within c *rounds of each other) is measured from the initiation of the first honest party until the termination of the last honest party.*

This result holds unconditionally for the case of $t < n/3$*, and under the assumption of a PKI and digital signatures for* $t < n/2$*.*

Proof We describe execution of $\mathsf{Expand}_c(\mathsf{pbc})$ from the perspective of party P_k.

1. P_k sends nothing for the first c rounds of its activation.
2. In round $c+1$, party P_k sends its 1st-round message of pbc.

   Define $r_i \overset{\text{def}}{=} c + 1 + (i-1) \cdot (2c+1)$. In general, P_k sends its ith-round message of pbc in round r_i.

[9]The round complexity of a run of a protocol is the round in which the *last* honest player terminates.

3. Round-i messages received from other parties are accepted only if they arrive in the $(2c+1)$-round interval between rounds $r_i - c$ and $r_i + c$, inclusive. If a message is not received from some party in that interval, a default message is used on behalf of that party instead.

   Observe that when P_k is supposed to send its ith-round message of pbc, it has already determined the incoming $(i-1)$th-round messages of all other parties.

4. When P_k terminates execution of pbc with output a vector v, it sends "exit, v" to all parties, along with a signature of this message in the authenticated case. It does not yet terminate execution of $\mathsf{Expand}_c(\mathsf{pbc})$; see the continuation of the protocol below.

(The first 3 steps of the above are similar, but not identical, to the approach of [37, Lemma 3.1].) In addition to the above, the following is done in parallel at every round:

Unconditional case: If there is a v such that P_k has received "exit, v" from $t+1$ distinct parties, then P_k sends "exit, v" to all parties (assuming it has not sent such a message already).

If P_k has received "exit, v" from $2t+1$ distinct parties, then it terminates immediately with output v.

Authenticated case: If there is a v such that P_k has received valid signatures from $t+1$ distinct parties on "exit, v," then P_k forwards "exit, v" along with these $t+1$ signatures to all parties and terminates immediately with output v.

We prove that $\mathsf{Expand}_c(\mathsf{pbc})$ achieves parallel broadcast as long as all honest parties begin within c rounds of each other. As in [37], all honest parties will accept the pbc-messages of all other honest parties. Furthermore, for any $j > i$ it is impossible for the adversary to make any of its round-i messages of pbc depend on the round-j message of pbc sent by any honest party. To see this, say the earliest round-j message of pbc sent by any honest party occurs in round R. Then all honest parties will send their round-j messages by round $R+c$ at the latest, and so send their round-i messages by round $R+c-(2c+1) = R-c-1$ at the latest. But this means that the last round in which any honest party accepts a round-i message on behalf of any other party is round $R-1$. Taken together, this proves that the execution of pbc taking place still achieves the properties of Definition 7 as long as all honest parties start within c rounds of each other.

We are not yet done proving correctness of $\mathsf{Expand}_c(\mathsf{pbc})$ since honest parties may terminate $\mathsf{Expand}_c(\mathsf{pbc})$ without terminating their local copy of pbc. Nevertheless, it is clear that all honest parties output the same value of v. Furthermore, no honest party terminates $\mathsf{Expand}_c(\mathsf{pbc})$ until *some* honest party has terminated its local copy of pbc. By what we have said in the previous paragraph, the output of any honest party who runs pbc to completion will satisfy the validity condition of Definition 7. It follows that the output of all honest parties in $\mathsf{Expand}_c(\mathsf{pbc})$ will satisfy the validity condition. This proves that $\mathsf{Expand}_c(\mathsf{pbc})$ achieves parallel broadcast.

We next prove that the staggering gap of $\mathsf{Expand}_c(\mathsf{pbc})$ is 1.

Unconditional case: Let R denote the first round in which some honest party P_i terminates $\mathsf{Expand}_c(\mathsf{pbc})$ with output v. So by round R, party P_i has received $2t+1$ copies of "exit, v," at least $t+1$ of which are from honest parties. Hence all honest parties have received at least $t+1$ copies of "exit, v" by round R and send (or have already sent) "exit, v" by round $R+1$. Since there are at least $2t+1$ honest parties, it follows that all honest parties receive $2t+1$ copies of "exit, v" (and hence terminate $\mathsf{Expand}_c(\mathsf{pbc})$) by round $R+1$.

Authenticated case: Let R denote the first round by which some honest party P_i has received $t+1$ valid signatures on "exit, v". Then P_i forwards these signatures to all other parties (and terminates) in round $R+1$. It follows that all honest parties will terminate by round $R+2$ at the latest.

An execution of $\mathsf{Expand}_c(\mathsf{pbc})$ terminates at latest by two rounds after the last honest party completes its execution of pbc. If a given execution of pbc takes ℓ rounds, this translates to a round complexity of $(2c+1)\cdot\ell+2$ for the corresponding execution of $\mathsf{Expand}_c(\mathsf{pbc})$. The stated round complexity takes into account the fact one honest party may have begun c rounds after another. ■

Unlike the analogous result in [37], Lemma 15 does not apply to *arbitrary* protocols (rather, we have explicitly stated the lemma only for protocols achieving parallel broadcast) since the proof uses the fact that all honest parties should terminate with *identical* outputs.

To see how Lemma 15 can be applied to the problem of sequential composition, suppose we have a protocol Π (relying on a broadcast channel) that has fixed round complexity $\ell = \mathsf{rc}(\Pi)$ and uses parallel broadcast in each round. Say we want to instantiate the ℓ invocations of parallel broadcast using sequential composition of protocols $\mathsf{pbc}_1, \ldots, \mathsf{pbc}_\ell$, each having staggering gap g. (We also assume that the point-to-point round-i messages of Π are sent as part of the first round pbc_i.) We then run ℓ sequential executions of $\mathsf{pbc}'_i = \mathsf{Expand}_1(\mathsf{pbc}_i)$ instead. (A small optimization is to set $\mathsf{pbc}'_1 = \mathsf{Expand}_0(\mathsf{pbc}_1)$, relying on the fact that honest parties start pbc'_1 in the same round.) Letting Π' denote the resulting protocol, we have:

Claim 16 *Protocol Π' in the point-to-point model achieves the same security guarantees as Π in the broadcast model. (Formally, we mean this as in [37, Lemma 3.1(1)].) The expected round complexity of Π' is $O(\ell\cdot\mathsf{rc}(\mathsf{pbc}))$. In particular, if Π is a constant-round protocol and pbc is an expected constant-round broadcast protocol, then Π' runs in an expected constant number of rounds.*

Proof We argue security of Π' as follows. We first prove by induction that, for all i, all honest parties begin execution of pbc'_i within one round of each other. For $i = 1$ this is immediate. Assuming it is true for i, Lemma 15 guarantees that all honest parties terminate pbc'_i within one round of each other. But then all honest parties will begin pbc'_{i+1} within one round of each other.

Since all honest parties begin execution of pbc'_i within one round of each other, Lemma 15 ensures that each execution of pbc'_i is secure in the sense of Definition 7.

The only thing left to argue is that no dishonest party can cause its messages in the ith round of Π (whether these are broadcast messages or point-to-point messages) to depend on the messages of any honest party in the jth round of Π, for any $j > i$. (It is not a problem if the round-i message of a dishonest party depends on a round-i message of an honest party, since we allow rushing when considering the security of Π.) Assume for simplicity that $i = j - 1$ (the argument extends easily to the general case). Consider an honest party P_k who begins running pbc'_j in some round R. At this point the output of P_k in pbc'_i is determined and hence, since pbc'_i achieves parallel broadcast, the broadcast messages of all the dishonest parties corresponding to the ith round of Π are already fixed. As for point-to-point messages of Π, note that P_k does not send any messages as part of pbc'_j until round $R+1$. Furthermore, since P_k terminated pbc'_i in round $R-1$, all honest parties terminate pbc'_i by round R at the latest. It follows that no honest party will accept any round-i messages in the underlying protocol Π once any honest party sends its round-j message of Π. ■

4.3 Secure Multiparty Computation in Expected Constant Rounds

Beaver, et al. [2] and Damgård and Ishai [15] show computationally-secure constant-round protocols for secure multi-party computation tolerating dishonest minority, assuming the existence of one-way functions, private and authenticated point-to-point channels, and a broadcast channel. (The solution of [15] is secure against an adaptive adversary.) We can obtain an expected constant-round protocol Π' in the point-to-point model (assuming one-way functions and a PKI) using the approach outlined in the previous section. (In fact, we can avoid the issue of sequential composition altogether by relying on subsequent work [32] which shows a constant-round protocol for secure computation that uses broadcast in only a *single* round.)

Theorem 17 *Assuming the existence of one-way functions, for every probabilistic polynomial-time functionality f there exists an expected constant-round protocol for computing f in a point-to-point network of private and authenticated channels assuming a PKI. The protocol is computationally secure* **without** *abort, and tolerates $t < n/2$ adaptive corruptions.*

Security without abort is the standard notion of security in the case of honest majority; see [28]. Roughly speaking, this definition implies the usual security guarantees of privacy and correctness, and additionally guarantees that all honest parties always receive their output. In contrast, a constant-round protocol given by Goldwasser and Lindell [29] guarantees privacy and correctness, but does not guarantee output delivery (namely, some honest parties may not receive any output), fairness (namely, the adversary may receive output even though honest parties do not), or "unanimity" (namely, some honest parties may receive their output while others do not).

5 Exact Round Complexities

In this section, we compute the exact round complexities of our broadcast/Byzantine agreement protocols, applying in the process some optimizations. We also discuss the round complexities for multiple sequential invocations of our protocols, relying on the results stated in Section 4.

5.1 The Unconditional Case ($t < n/3$)

Let us examine the Byzantine agreement protocol $\mathcal{BA}$ given in the proof of Theorem 11. Recall that $\mathcal{BA}$ consists of multiple iterations; steps 1–5 of each iteration require only one round each, while in step 6 of an iteration the two phases of an OLE protocol are run. Taking the OLE protocol from the proof of Theorem 8, note that the second phase of that OLE protocol takes only a single round. Letting r denote the round complexity of the first phase of the underlying OLE protocol, we see that the round complexity of a single iteration of $\mathcal{BA}$ is $5 + r + 1 = 6 + r$. Multiplying this by the expected number of iterations yields the expected round complexity of $\mathcal{BA}$.

We can, however, do better. The key observation is that the first phase of the OLE protocol can be carried out in advance of step 6, and in particular can be carried out in parallel with steps 1–5. (A similar observation was made in [18].) Even more, we can run multiple invocations of the first phase of the OLE protocol and "save them" until needed. Applying these ideas, we obtain the following protocol for Byzantine agreement (r again denotes the round complexity of the first phase of the underlying OLE protocol, and we assume $r \geq 5$):

1. Run $\ell \overset{\text{def}}{=} \lceil r/6 \rceil$ parallel executions of the first phase of the OLE protocol. These are scheduled so that the final 5 rounds coincide with steps 1–5 of the first iteration of $\mathcal{BA}$.
2. For the remainder of the protocol, continually run ℓ parallel executions of the first phase of the OLE protocol in parallel with the "main" protocol. These parallel executions will terminate every r rounds, just as the ℓ previous executions get "used up."

To compute the resulting (expected) round complexity of the Byzantine agreement protocol, recall that an honest leader is elected with probability at least 2/3 in each iteration, and when an honest leader is elected all honest parties terminate by the following iteration. The expected number of iterations until a leader is elected is therefore at most $(2/3)^{-1} = 3/2$, and the expected number of iterations is at most $5/2$. Each iteration requires 6 rounds. Furthermore, we have an additional $r - 5$ rounds during which the initial ℓ executions of the first phase of the OLE protocol are run (recall that the final 5 rounds of these initial ℓ executions coincide with the first 5 rounds of the first iteration). We thus obtain that the expected round complexity of $\mathcal{BA}$ is at most:

$$6 \cdot \frac{5}{2} + (r - 5) = 10 + r.$$

Starting with the VSS protocol of [33], converting it to a moderated VSS protocol using Theorem 5, and then constructing an OLE protocol as described in Theorem 8 (using the 3-round gradecast protocol from Appendix A.1), we obtain an OLE protocol with $r = 8$. The expected round complexity of the Byzantine agreement protocol constructed in this work, then, is at most $10 + 8 = 18$. Broadcast can be implemented in the same number of rounds by having the dealer send its message to all parties during the first round of the initial executions of OLE.

5.2 The Authenticated Case ($t < n/2$)

Improved round complexity for authenticated BA. We start by constructing an authenticated Byzantine agreement protocol directly from OLE in the point-to-point model (rather than relying on an underlying gradecast protocol as in Theorem 13). Essentially, this protocol is obtained from the protocol described in the proof of Theorem 13 by unrolling the gradecast protocol and eliminating redundant steps.

Let V be the domain of possible inputs, let the input value for party P_i be $v_i \in V$, let $\phi \in V$ be some default value and let $\perp \notin V$. We say that P_i has a *(valid) certificate for v* if $v \in V$ and there exist $k > n/2$ distinct indices $j_1, \ldots, j_k$ such that P_i holds $\sigma_{j_1,i}, \ldots, \sigma_{j_k,i}$ which are valid signatures on v with respect to the public keys of $P_{j_1}, \ldots, P_{j_k}$. In this case, we will also call $(v, j_1, \ldots, j_k, \sigma_{j_1,i}, \ldots, \sigma_{j_k,i})$ the *certificate for v*.

Each P_i begins with an internal variable $\texttt{lock}_i$ set to ∞. To avoid having to say this every time, we impose the implicit requirement that if $\texttt{lock}_i \neq \infty$ then the value of v_i is "locked" and remains unchanged (i.e., even if the protocol description below says to change it).

Step 1 Party P_i computes a signature σ_i of v_i and sends (v_i, σ_i) to all parties.

Step 2 Let $(v_{j,i}, \sigma_{j,i})$ be the message received by party P_i from P_j. If these messages yield a certificate for v_i, then P_i sends a certificate for v_i to all parties. Otherwise P_i sends nothing and sets $v_i := \perp$.

Step 3 If in the previous round P_i received a valid certificate for some $v^* \neq v_i$, then P_i sets $v_i := \perp$.
If $v_i \neq \perp$, then P_i computes a signature[10] σ'_i of v_i and sends (v_i, σ'_i) to all parties.

Step 4 Let $(v_{j,i}, \sigma'_{j,i})$ be the message received by party P_i from P_j (if any) in the previous round. If these messages yield a certificate for v_i, then P_i sends a certificate for v_i to all parties and sets $\texttt{lock}_i := 1$; otherwise P_i sends nothing and sets $v_i := \perp$.

Step 5 If in the previous round P_i received a valid certificate on some value v^*, then P_i sends a certificate on v^* to all parties and sets $v_i := v^*$. Otherwise, P_i sends nothing and sets $v_i := \perp$.

Step 6 If in the previous round P_i received a valid certificate on some value v^*, then P_i sends v^* to all parties; otherwise, P_i sends $\perp$ to all parties. Let $v^*_{j,i}$ be the value P_i received from P_j in this round.

[10]The current round number is also signed to distinguish this signature from others.

Step 7 All parties execute the OLE protocol; let ℓ_i be the output of P_i. If $v_i =\perp$ and $v^*_{\ell_i,i} \neq\perp$, then P_i sets $v_i := v^*_{\ell_i,i}$. If $v_i = v^*_{\ell_i,i} =\perp$, then P_i sets $v_i := \phi$. If $\texttt{lock}_i = 0$, then P_i outputs v_i and terminates the protocol. If $\texttt{lock}_i = 1$, then P_i sets $\texttt{lock}_i := 0$ and goes to step 1. If $\texttt{lock}_i = \infty$, then P_i goes to step 1.

Lemma 18 *Assume there exists a constant-round authenticated OLE protocol with fairness $\delta = \Omega(1)$ tolerating $t < n/2$ malicious parties. Then the above protocol is an expected constant-round authenticated Byzantine agreement protocol tolerating t malicious parties.*

Proof The proof is very similar to the proof of Theorem 13. We refer to an execution of steps 1 through 7 as an *iteration*. One can check by inspection that if — immediately prior to any given iteration — there exists a value v such that $v_i = v$ for all honest P_i and no honest parties have yet terminated, then all honest parties will terminate and output v by the end of the following iteration. (This in particular proves validity.)

An easy observation is that at any given step there is at most one value $v \in V$ for which some honest party has a certificate on v (otherwise, some honest party must have signed two different values; but this cannot occur). Now consider the first iteration in which an honest party P_i sets $\texttt{lock}_i := 1$ (in step 4). We claim that, by the end of that iteration, $v_j = v_i$ for all honest P_j and no honest parties will have yet terminated. The claim regarding termination is immediate. From the above observation, any honest party P_j setting $\texttt{lock}_j := 1$ in this iteration must also hold $v_j = v_i$. For an honest party P_j holding $\texttt{lock}_j = \infty$ after step 4, since P_i sends a certificate for v_i to all parties (in step 4) the earlier observation again implies that P_j sets $v_j := v_i$ in step 5. Since $v_j = v_i \neq\perp$ (else P_i would not have set $\texttt{lock}_i := 1$), P_j will not change the value of v_j in step 7. This establishes the claim, and implies that if any honest party terminates then all honest parties terminate with the same output.

To complete the proof, we show that if an honest leader P_ℓ is elected in some iteration then all honest parties will hold the same value v by the end of that iteration. By what we have argued in the previous paragraph, we only need to consider the case where $\texttt{lock}_i = \infty$ for all honest P_i in step 7. If $v_i =\perp$ for all honest P_i by the end of step 5, the claim is immediate since every honest P_i will change their value of v_i to the honest leader's value (or ϕ, as appropriate) in step 7. Otherwise, $v_i \neq\perp$ by the end of step 5 for some honest party P_i. Arguing as before, we see that every honest party P_j holds $v_j \in \{v_i, \perp\}$ by the end of step 5. Furthermore, P_ℓ receives a valid certificate on v_i from P_i in step 5 and so sends $v^*_\ell = v_i$ to all parties in step 6. Hence every honest party P_j holds $v_j = v_i$ by the end of that iteration. ■

Exact round complexity. We now compute the exact round complexity of the above protocol. An honest leader is elected with probability at least 1/2 in each iteration, and all honest parties terminate by two iterations following the one in which an honest leader is elected. The expected number of iterations is therefore at most $(1/2)^{-1}+2 = 4$. Each iteration requires 7 rounds. As in the previous section, we can improve the round complexity by using an additional $r - 6$ rounds before the first iteration begins to run $\lceil r/7 \rceil$ executions of the first phase of OLE (and then continuing to run OLE in parallel with the remainder of the protocol). Doing so, the expected round complexity is at most:

$$7 \cdot 4 + (r - 6) = 22 + r.$$

Starting with the authenticated VSS protocol described in Appendix A.3, converting it to an authenticated moderated VSS protocol using Theorem 5, and then constructing an authenticated OLE protocol as described in Theorem 8 (using the 4-round gradecast protocol from Lemma 2), we obtain an authenticated OLE protocol with $r = 34$. The expected round complexity of the authenticated Byzantine agreement protocol constructed in this work, then, is at most $22+34 = 56$. Broadcast can be implemented in the same number of rounds by having the dealer send its message to all parties during the initial executions of OLE.

5.3 Amortized Round Complexity of Sequential Broadcasts

We focus now on broadcast, rather than Byzantine agreement. The observation here is that when running multiple sequential executions of broadcast, we no longer need to count the rounds for the initial "set-up" phase (in which the first phase of OLE is run) each time, since these can be run in parallel with the preceding execution of broadcast. (Now, however, we require an additional round to implement broadcast, since the dealer cannot send its message to all parties until the preceding execution of broadcast has finished.) Thus, we obtain an amortized (expected) round complexity of $1 + 6 \cdot \frac{5}{2} = 16$ in the unconditional case and $1 + 7 \cdot 4 = 29$ in the authenticated case. Applying Lemma 15 to handle the issue of non-simultaneous termination, we obtain:

- For $t < n/3$, excepting the first invocation of broadcast, each invocation of broadcast incurs an (expected) cost of at most $3 \cdot 16 + 2 = 50$ rounds. (Inspection of Lemma 15 shows that $\mathsf{rc}\left(\mathsf{Expand}_c(\mathsf{pbc})\right) \leq (2c+1) \cdot \mathsf{rc}(\mathsf{pbc}) + c + 1$ in the unauthenticated case.)
- For $t < n/2$, excepting the first invocation of broadcast, each invocation of authenticated broadcast incurs an (expected) cost of at most $3 \cdot 29 + 3 = 90$ rounds.

The round complexity, per broadcast sub-routine, when sequential executions of broadcast are used is higher than the round complexity of a single execution of broadcast. This motivates designing protocols that use only a *single* round of broadcast; see [32, 33] for work in this direction.

Acknowledgments

We thank the anonymous reviewers for several helpful comments.

References

[1] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology — Eurocrypt '92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. Springer-Verlag, 1993.

[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 503–513, 1990.

[3] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *2nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 27–30, 1983.

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

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[6] M. Ben-Or, E. Pavlov, and V. Vaikuntanathan. Byzantine agreement in the full-information model in $O(\log n)$ rounds. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 179–186, 2006.

[7] G. Blakley. Safeguarding cryptographic keys. In *National Computer Conference*, volume 48, pages 313–317. AFIPS Press, 1979.

[8] G. Bracha. An $O(\log n)$ expected rounds randomized Byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.

[9] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. *J. Cryptology*, 18(3):219–246, 2005.

[10] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[11] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648, 1996.

[12] B. Chor and B. Coan. A simple and efficient randomized Byzantine agreement algorithm. *IEEE Trans. Software Engineering*, 11(6):531–539, 1985.

[13] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 383–395, 1985.

[14] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology — Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer-Verlag, 1999.

[15] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology — Crypto 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer-Verlag, 2005.

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

[17] C. Dwork, D. Shmoys, and L. Stockmeyer. Flipping persuasively in constant time. *SIAM J. Computing*, 19(3):472–499, 1990.

[18] P. Feldman. *Optimal Algorithms for Byzantine Agreement.* PhD thesis, Massachusetts Institute of Technology, 1988.

[19] P. Feldman and S. Micali. Byzantine agreement in constant expected time (and trusting no one). In *26th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 267–276, 1985.

[20] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Computing*, 26(4):873–933, 1997.

[21] M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

[22] M. Fitzi and J. Garay. Efficient player-optimal protocols for strong and differential consensus. In *22nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 211–220, 2003.

[23] M. Fitzi, J. Garay, S. Gollakota, C. P. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *3rd Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer-Verlag, 2006.

[24] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2000.

[25] J. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–668, 2007.

[26] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Computing*, 27(1):247–290, 1998.

[27] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 580–589, 2001.

[28] O. Goldreich. *Foundations of Cryptography, Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[29] S. Goldwasser and Y. Lindell. Secure computation without agreement. *J. Cryptology*, 18(3):247–287, 2005.

[30] S. Goldwasser, E. Pavlov, and V. Vaikuntanathan. Fault-tolerant distributed computing in full-information networks. In *47th Anuual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 15–26, 2006.

[31] A. Hevia and D. Micciancio. Simultaneous broadcast revisited. In *24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 324–333, 2005.

[32] J. Katz and C.-Y. Koo. Round-efficient secure computation in point-to-point networks. In *Advances in Cryptology — Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 311–328. Springer-Verlag, 2007.

[33] J. Katz, C.-Y. Koo, and R. Kumaresan. Improving the round complexity of VSS in point-to-point networks. In *35th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510. Springer-Verlag, 2008.

[34] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118, 2006.

[35] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 4(3):382–401, 1982.

[36] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 514–523, 2002.

[37] Y. Lindell, A. Lysyanskaya, and T. Rabin. Sequential composition of protocols without simultaneous termination. In *21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 203–212, 2002.

[38] J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology — Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 401–416. Springer-Verlag, 2002.

[39] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

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

[41] M. Rabin. Randomized Byzantine generals. In *24th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–409, 1983.

[42] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[43] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[44] S. Toueg. Randomized Byzantine agreements. In *3rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 163–178, 1984.

[45] R. Turpin and B. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984. Available at http://publications.csail.mit.edu/lcs/specpub.php?id=883.

[46] M. Waidner. *Byzantinische Verteilung ohne Kryptographische Annahmen trotz Beliebig Vieler Fehler*. PhD thesis, University of Karlsruhe, 1991. (In German).

A Additional Proofs

A.1 Proof of Lemma 1

We show a constant-round gradecast protocol (essentially from [20, Section 5.1]) tolerating $t < n/3$ malicious parties. The protocol proceeds as follows:

Round 1 The dealer sends M to all other parties.

Round 2 Let M_i denote the message received by P_i (from the dealer) in the previous round. P_i sends M_i to all the parties.

Round 3 Let $M_{j,i}$ denote the message received by P_i from P_j in the previous round. Each party P_i does the following: if there exists an M_i^* such that $|\{j : M_{j,i} = M_i^*\}| \geq 2n/3$ then P_i sends this M_i^* to all the parties. Otherwise, P_i sends nothing.

Output determination Let $M^*_{j,i}$ denote the message (if any) received by P_i from P_j in the previous round. Each party P_i determines its output as follows: if there exists an M^{**}_i such that $|\{j : M^*_{j,i} = M^{**}_i\}| \geq 2n/3$, then P_i outputs $m_i := M^{**}_i$ and $g_i := 2$. Otherwise, if there exists[11] an M^{**}_i such that $|\{j : M^*_{j,i} = M^{**}_i\}| \geq n/3$, then P_i outputs $m_i := M^{**}_i$ and $g_i := 1$. Otherwise, P_i outputs $m_i := \perp$ and $g_i := 0$.

We prove that the above protocol satisfies Definition 3. Assume first that the dealer is honest. Then each honest party P_i receives $M_i = M$ in round 1 and sends this to all parties in round 2. So in round 3, for each honest P_i it holds that $M^*_i = M$ and so P_i sends this value to all other parties. It follows that any honest party P_i outputs $m_i = M$ and $g_i = 2$.

Before proving the second required property, we show that if any two honest parties P_i, P_j send a message in round 3 then they in fact send the *same* message. To see this, say P_i sends M^*_i in round 3. Then P_i must have received M^*_i from at least $2n/3$ parties in round 2, and so strictly more than $n/3$ honest parties must have sent M^*_i in round 2. But this means that P_j receives any value $M^*_j \neq M^*_i$ from strictly fewer than $n - n/3 = 2n/3$ parties in round 2, and so P_j either sends M^*_i or nothing in round 3.

Now assume there is an honest party P_i who outputs a message m_i and grade $g_i = 2$, and let P_j be any other honest party. P_i must have received m_i from at least $2n/3$ parties in round 3, and so strictly more than $n/3$ honest parties sent m_i as their round-3 message. It follows that $|\{k : M^*_{k,j} = m_i\}| \geq n/3$ and so P_j outputs grade $g_j \geq 1$. Say there was an $m_j \neq m_i$ for which $|\{k : M^*_{k,j} = m_j\}| \geq n/3$. Then at least one honest party sent $m_j \neq m_i$ as its round-3 message, contradicting what was shown in the previous paragraph. So, P_j outputs m_i as required.

A.2 Proof of Lemma 3

The following 4-round VSS protocol, due to [27], tolerates $t < n/3$ malicious parties. (Recently, 3-round protocols have also been shown [23, 33] but these are not needed to prove the lemma.) The protocol assumes the existence of a broadcast channel in the sharing phase, but not in the reconstruction phase. Let $\mathbb{F}$ be a finite field with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n] \subset \mathbb{F}$. In the description that follows, we implicitly assume that all parties send a properly-formatted message at all times (this is without loss of generality, as we may interpret an improper or missing message as some default message). When the dealer is *disqualified* this means that execution of the protocol halts, and all honest parties output some default value (0, say) in the reconstruction phase.

Sharing Phase

Round 1 The dealer chooses a random bivariate polynomial $F \in \mathbb{F}[x, y]$ of degree at most t in each variable with $F(0, 0) = s$. The dealer sends to P_i the polynomials $g_i(x) \overset{\text{def}}{=} F(x, i)$ and $h_i(y) \overset{\text{def}}{=} F(i, y)$.

In parallel, each P_i sends a random $r_{i,j} \in \mathbb{F}$ to each P_j (including itself).

Round 2 P_i broadcasts $c_{j,i} := g_i(j) + r_{j,i}$ and $d_{i,j} := h_i(j) + r_{i,j}$ for all j.

Round 3 For each $c_{j,i} \neq d_{j,i}$, the following is performed:

- P_i broadcasts $a_{j,i} := g_i(j)$.
- P_j broadcasts $b_{j,i} := h_j(i)$.
- The dealer broadcasts $s_{j,i} := F(j, i)$.

[11]It will follow from the proof below that at most one such M^{**}_i exists in this case.

A party is said to be *unhappy* if any value he broadcasted in this round does not match the corresponding value broadcasted by the dealer, and *happy* otherwise.

Round 4 For each unhappy party P_i, the dealer broadcasts the polynomial $g_i(x)$ and each happy party P_j broadcasts $b'_{j,i} := h_j(i)$.

End of sharing phase A party P_j who was happy at the beginning of round 4 becomes unhappy if the dealer broadcasted $g_i(x)$ in round 4 such that $b'_{j,i} \neq g_i(j)$. If the number of unhappy parties is now more than t, the dealer is disqualified.

Reconstruction Phase

Round 1 If P_i was happy at the beginning of round 4 of the sharing phase, then P_i sends $s_i := g_i(0)$ to all parties; otherwise, P_i sends nothing.

Output determination Party P_i proceeds as follows: if P_j was happy at the beginning of round 4 of the sharing phase, let s_j be the value P_j sent to P_i in the previous round; otherwise, set $s_j := g_j(0)$ (where $g_j(x)$ is the polynomial broadcast by the dealer in round 4 of the sharing phase). Let $g(y)$ be the degree-t polynomial resulting from applying Reed-Solomon error-correction to $(s_1, s_2, \ldots, s_n)$. Output $g(0)$.

We first prove secrecy. Assume the dealer is honest. After round 1, the information the malicious parties have about the dealer's secret s consists of the polynomials sent to the malicious parties by the dealer in round 1 or, equivalently, the values $\{F(i,j) \mid P_i \text{ or } P_j \text{ malicious}\}$. We show that in the remainder of the sharing phase the adversary does not obtain any information about $F(i,j)$ if both P_i, P_j are honest; secrecy follows since F is a degree-t bivariate polynomial and there are at most t malicious parties.

Let P_i, P_j be players who remain honest. Round 2 leaks no information about $g_j(i) = F(i,j) = h_i(j)$ due to the random pads that are used. Furthermore, $c_{i,j} = d_{i,j}$ and so no information about $F(i,j)$ is revealed in round 3. Finally, all honest parties are happy at the end of the sharing phase and so no information about $F(i,j)$ is revealed in round 4. This completes the proof of secrecy.

We continue to assume the dealer is honest, and prove validity. As noted above, all honest parties are happy at the end of the sharing phase and so the dealer is not disqualified. Furthermore, every honest party sends $s_i = g_i(0) = F(0,i)$ to all other parties in the reconstruction phase. Since $n > 3t$ and there are at most t "bad" shares in $\{s_1, s_2, \ldots, s_n\}$, Reed-Solomon error-correction recovers the polynomial $g(y) = F(0,y)$ and hence all honest parties output $g(0) = F(0,0) = s$.

Lastly, we prove the reconstruction property. For the case of an honest dealer, this follows from validity. The reconstruction property is also trivially satisfied if the dealer is disqualified. Thus, in what follows we assume a dishonest dealer who is not disqualified.

For any party P_i who remains honest throughout the entire protocol, its "contribution" s_i in the reconstruction phase (whether sent in round 1 of the reconstruction phase or automatically set equal to $g_i(0)$) is fixed at the end of the sharing phase. Furthermore, the vectors $(s_1^{(j)}, \ldots, s_n^{(j)})$ and $(s_1^{(k)}, \ldots, s_n^{(k)})$ used in the output determination step by two honest parties P_j, P_k agree in at least the $2t+1$ positions corresponding to honest parties. If we can show that the values $\{s_i \mid P_i \text{ is honest at the end of the sharing phase}\}$ lie on a degree-t (univariate) polynomial, then by the properties of Reed-Solomon decoding we see that (1) regardless of the adversary's actions in the reconstruction phase, each honest party will recover the same $g(y)$ and hence output the same $g(0)$; and furthermore (2) this value $g(0)$ is determined by the joint view of the honest parties at the end of the sharing phase, as desired.

We will use the following standard claim:

Claim 19 *Let $x_1, x_2, \ldots, x_{t+1}$ be distinct elements in $\mathbb{F}$, and $h_1(y), \ldots, h_{t+1}(y)$ be polynomials of degree t. Then there exists an unique bivariate polynomial $F'(x, y)$ of degree t in both variables such that $F'(x_i, y) = h_i(y)$ for $i = 1, \ldots, t+1$.*

Since the dealer is not disqualified, there are at least $2t+1$ happy parties at the end of the sharing phase and at least $t+1$ of them are honest. Let $\mathcal{H}$ denote the indices of an arbitrary set of $t+1$ such parties (so $i \in \mathcal{H}$ means P_i was happy and honest at the end of the sharing phase). Let $g_i(x), h_i(y)$ be the polynomials sent to P_i (for $i \in \mathcal{H}$) by the dealer in the first round. By Claim 19, there exists a unique bivariate polynomial $F'(x, y)$ of degree t such that $F'(i, y) = h_i(y)$ for $i \in \mathcal{H}$. We prove below that for all honest P_i, the contribution s_i (in the reconstruction phase) will be equal to $F'(0, i)$ and so the values $\{s_i \mid P_i \text{ honest}\}$ lie on the degree-t polynomial $F'(0, y)$. The reconstruction property follows.

Before continuing, note that (using Claim 19 again) there exists a unique degree-t polynomial $F''(x, y)$ such that $F''(x, i) = g_i(x)$ for $i \in \mathcal{H}$. But then for any $i, j \in \mathcal{H}$, we have $F'(i, j) = h_i(j) = g_j(i) = F''(i, j)$ (using for the second equality the fact that P_i, P_j are happy and honest) and so in fact the bivariate polynomials F', F'' are identical (since $|\mathcal{H}| = t+1$).

We now prove that $s_i = F'(0, i)$ for all honest parties P_i (whether happy or not):

1. If P_i was happy at the beginning of round 4, then $g_i(j) = h_j(i) = F'(j, i)$ for all $j \in \mathcal{H}$. Since $|\mathcal{H}| = t+1$ and the polynomials $g_i(x)$ and $F'(x, i)$ have degree t, they must be identical and so $s_i \stackrel{\text{def}}{=} g_i(0) = F'(0, i)$.
2. If P_i was unhappy at the beginning of round 4, let $d_i(x)$ be the appropriate polynomial broadcasted by the dealer in round 4. We must have $d_i(j) = h_j(i) = F'(j, i)$ for all $j \in \mathcal{H}$ (since P_j is happy at the end of the sharing phase). As before, since $d_i(x)$ and $F'(x, i)$ have degree t and agree on $t+1$ points, they must be identical and so $s_i \stackrel{\text{def}}{=} d_i(0) = F'(0, i)$.

A.3 Proof of Lemma 4

We show a constant-round protocol for authenticated VSS which tolerates $t < n/2$ malicious parties. The protocol assumes a broadcast channel during the sharing phase, but not during the reconstruction phase. Our protocol is adapted from work of Cramer, et al. [14, Section 5] with two modifications: (1) the protocol of [14] uses "information checking" whereas we use digital signatures; and (2) the protocol of [14] uses the broadcast channel during the reconstruction phase, and we avoid this. We now provide the details.

As in the proof of Lemma 3, we assume a finite field $\mathbb{F}$ with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n] \subset \mathbb{F}$. We continue to assume, without loss of generality, that parties send properly-formatted messages. If the dealer is *disqualified* then execution of the protocol halts, and all parties output some default value in the reconstruction phase. Finally, we say an ordered sequence of values $(v_1, \ldots, v_n) \in \mathbb{F}^n$ is *t-consistent* if there exists a degree-t polynomial f such that $f(i) = v_i$ for $1 \leq i \leq n$.

<u>Sharing Phase</u>

Round 1 The dealer chooses a random bivariate polynomial $F \in \mathbb{F}[x, y]$ of degree t in each variable with $F(0, 0) = s$. Let $a_{i,j} = b_{i,j} \stackrel{\text{def}}{=} F(i, j)$. The dealer sends to party P_i the values $a_{1,i}, \ldots, a_{n,i}$ and $b_{i,1}, \ldots, b_{i,n}$, along with a signature on each such value.[12]

Round 2 Let $\vec{a}_i = (a_{1,i}, \ldots, a_{n,i})$ and $\vec{b}_i = (b_{i,1}, \ldots, b_{i,n})$ denote the values received by party P_i in the previous step. If P_i does not receive a valid signature on all these values as specified in the

[12]More precisely, the dealer signs the "message" $(i, j, F(i, j))$.

previous step, then P_i broadcasts a complaint. P_i also checks that $\vec{a}_i, \vec{b}_i$ are each t-consistent. If not, P_i broadcasts these values along with the dealer's signatures on them; upon receiving such a broadcast from any other party (and verifying the dealer's signatures and the fact that the values are not t-consistent), the dealer is disqualified.

Round 3 If P_i broadcasted a complaint in round 2, the dealer broadcasts $\vec{a}_i = (a_{1,i}, \ldots, a_{n,i})$, $\vec{b}_i = (b_{i,1}, \ldots, b_{i,n})$, and signatures on each of these values; P_i uses these values for the remainder of the protocol. If the dealer broadcasts incorrect signatures in response to a complaint, or if any of the broadcasted $\vec{a}_i$, $\vec{b}_i$ are not t-consistent, the dealer is disqualified.

Note that unless the dealer is disqualified at this point, every honest party P_i now has t-consistent vectors $\vec{a}_i, \vec{b}_i$, and valid signatures of the dealer on each of $\{a_{j,i}, b_{i,j}\}_{j=1}^{n}$.

Round 4 Party P_i computes signature $\sigma_{j,i}$ on $(j, i, a_{j,i})$, and sends $(a_{j,i}, \sigma_{j,i})$ to party P_j.

Round 5 P_i compares the value $a_{i,j}$ it received from P_j in the previous step to the value $b_{i,j}$ it received from the dealer. If there is an inconsistency, or if P_j did not send a valid signature, then P_i broadcasts $b_{i,j}$ and the dealer's signature on this value.

Round 6 P_i checks if any party P_j broadcasted a value $b_{j,i}$ which is different from the value $a_{j,i}$ that P_i holds. If so, then P_i broadcasts $a_{j,i}$ and the dealer's signature on this value.

End of sharing phase If there exists a pair (i, j) such that $a_{i,j}$ and $b_{i,j}$ were both broadcast with valid signatures of the dealer and $a_{i,j} \neq b_{i,j}$, the dealer is disqualified.

Reconstruction Phase

Round 1 For every j such that P_i has a valid signature $\sigma_{i,j}$ (with respect to the public key of P_j) on $b_{i,j}$, party P_i sends $(b_{i,j}, \sigma_{i,j})$ to all other parties. Note that for all other j, party P_i has already broadcasted $b_{i,j}$ (with the dealer's signature) in round 5 of the sharing phase.

Output determination For each $1 \leq j \leq n$, party P_i verifies the signatures on the values received from P_j in the previous round, and disqualifies P_j if any of the signatures are invalid.

For each P_j which is not yet disqualified, P_i has values $\vec{b}_i^j \overset{\text{def}}{=} (b_{j,1}, \ldots, b_{j,n})$ (each of these values was either received from P_j in the previous round or was broadcast with a valid dealer signature in round 5 or 6 of the sharing phase). If $\vec{b}_i^j$ is not t-consistent, P_i disqualifies P_j.

Let $\mathcal{H}_i$ be the set of non-disqualified parties, from the perspective of P_i. For each $j \in \mathcal{H}_i$, party P_i interpolates $\vec{b}_i^j$ to obtain a degree-t polynomial $f'_j(y)$ (recall that $\vec{b}_i^j$ is t-consistent). Next, P_i interpolates the $\{f'_j(y)\}_{j \in \mathcal{H}_i}$ to obtain a bivariate polynomial $F'(x, y)$ of degree t in both variables (the proof below will show that this is possible). Output $F'(0, 0)$.

We first prove secrecy. If the dealer is honest, no honest party will complain in round 2. Furthermore, if P_i, P_j are honest then $a_{i,j} = b_{i,j}$ in round 5 and so $b_{i,j}$ is not broadcast. It follows that the information malicious parties have about the dealer's secret s at the end of the sharing phase consists entirely of the values sent to the dishonest parties by the dealer in round 1. Secrecy follows since F is a degree-t bivariate polynomial and there are at most t malicious parties.

We next prove validity. It is easy to see that an honest dealer is never disqualified. Let P_i, P_j be parties that remain honest throughout the entire execution. The vector $\vec{b}_i^j$ (in the reconstruction phase) matches the values sent by the dealer in round 1 and furthermore $P_j \in \mathcal{H}_i$; thus, P_i recovers $f'_j(y) = F(j, y)$ for every honest P_j. For any malicious $P_k \in \mathcal{H}_i$, the value $b_{k,j}$ that P_i holds was either signed by P_j (in round 4) or broadcast with a valid dealer signature (in round 5 or 6), and so $b_{k,j} = F(k, j)$. Since this holds for at least $t + 1$ honest parties P_j and $\vec{b}_i^k$ is t-consistent (else

$k \notin \mathcal{H}_i$), we conclude that P_i recovers $f'_k(y) = F(k, y)$ in this case as well. So interpolating the $\{f'_j(y)\}_{j \in \mathcal{H}_i}$ yields $F(x, y)$ (interpolation can be done since $|\mathcal{H}_i| \geq t + 1$), and the output of P_i is the dealer's secret $F(0, 0)$.

Finally, we prove reconstruction. The case when the dealer is disqualified is obvious, so assume the dealer is not disqualified.

Let $\mathcal{U}$ be the indices of a set of $t+1$ parties who are honest at the end of the sharing phase. For honest P_i, let $\vec{b^i} = (b_{i,1}, \ldots, b_{i,n})$ denote the values that P_i will "effectively" send to other parties in the reconstruction phase (note that some of these values may, in fact, already have been broadcast). Let $f'_i(y)$ be the result of interpolating $\vec{b^i}$ (this is well-defined since $\vec{b^i}$ is t-consistent for honest P_i), and let $F'(x, y)$ be the result of interpolating the $f'_i(y)$ for $i \in \mathcal{U}$. We will show that regardless of the actions of the adversary in the reconstruction phase, each honest party outputs $F'(0, 0)$.

By construction of F', we have $b_{i,k} = F'(i, k)$ for $i \in \mathcal{U}$. We claim that $a_{k,i} = F'(k, i)$ for $i \in \mathcal{U}$. Let $g'_i(x)$ be the result of interpolating $\vec{a^i} = (a_{1,i}, \ldots, a_{n,i})$ (again, this is well-defined since $\vec{a^i}$ is t-consistent for P_i honest). Note that for $j \in \mathcal{U}$ we have $g'_i(j) \overset{\text{def}}{=} a_{j,i} = b_{j,i}$ or else the dealer would have been disqualified. So $g'_i(x)$ agrees with $F'(x, i)$ on $t + 1$ points and hence these polynomials must be identical, proving the claim.

Applying a similar argument (using the fact that, for P_i honest and $j \in \mathcal{U}$, we have $b_{i,j} = a_{i,j} = F'(i, j)$ or else the dealer is disqualified), we see that for any honest P_i the vector $\vec{b^i}$ interpolates to $f'_i(y) = F'(i, y)$. Furthermore, it is easy to see that if P_i, P_j remain honest then $P_i \in \mathcal{H}_j$. For any corrupted $P_k \in \mathcal{H}_j$ and honest P_i, the value $b_{k,i}$ that P_k sends to P_j in the reconstruction phase was either signed by P_i (in round 4) or broadcast with a valid dealer signature (in round 5 or 6), and so $b_{k,i} = F'(k, i)$. Since this holds for at least $t+1$ honest parties P_i and $\vec{b}^k_j$ is t-consistent (else $k \notin \mathcal{H}_j$), we conclude that P_j recovers $f'_k(y) = F'(k, y)$ in this case as well. So interpolating the $\{f'_i(y)\}_{i \in \mathcal{H}_j}$ yields $F'(x, y)$ (interpolation can be done since $|\mathcal{H}_j| \geq t + 1$), and the output of P_j is the dealer's secret $F'(0, 0)$.