On Expected Constant-Round Protocols for Byzantine Agreement

Jonathan Katz[⋆] and Chiu-Yuen Koo

Dept. of Computer Science, University of Maryland, College Park, USA
{jkatz,cykoo}@cs.umd.edu

Abstract. In a seminal paper, Feldman and Micali (STOC '88) show an n-party Byzantine agreement protocol tolerating $t < n/3$ malicious parties that runs in expected constant rounds. Here, 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 a secure signature scheme and a public-key infrastructure (PKI). Combined with existing results, this gives the first expected constant-round protocol for secure computation with honest majority in a point-to-point network assuming only one-way functions and a PKI. Our key technical tool — a new primitive we introduce called *moderated VSS* — also yields a simpler proof of the Feldman-Micali result.

We also show a simple technique for sequential composition of protocols without simultaneous termination (something that is inherent for Byzantine agreement protocols using $o(n)$ rounds) for the case of $t < n/2$.

1 Introduction

When designing cryptographic protocols, it is often convenient to abstract away various details of the underlying communication network. As one noteworthy example,[1] it is often convenient to 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. 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 protocol substantially.

Much work has therefore focused on reducing the round complexity of broadcast or the related task of *Byzantine agreement* (BA) [**?**,**?**]; we survey this work

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

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.

in Section 1.2. As discussed there, a seminal result of Feldman and Micali [?] is a protocol for Byzantine agreement in a network of n parties tolerating $t < n/3$ malicious parties that runs in an expected *constant* number of rounds. This resilience is the best possible — regardless of round complexity — unless additional 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); broadcast or BA protocols in this model are termed *authenticated*. Authenticated broadcast protocols are known for $t < n$ [?,?,?], but all existing protocols that assume only a PKI and secure signatures require $\Theta(t)$ rounds.[2] Recent work of Fitzi and Garay [?] gives an authenticated BA protocol beating this bound, but using 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* readily extend to the case of $t < n/2$. In particular, they rely on a primitive termed *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 new primitive called *moderated VSS* (mVSS) and use this to give an entirely self-contained proof of our result.

We suggest that mVSS is a useful alternative to graded VSS *in general*, even when $t < n/3$. For one, mVSS seems easier to construct: we show a generic construction of mVSS *in the point-to-point model* from any VSS protocol that relies on a broadcast channel, while a generic construction of this sort for graded VSS seems unlikely. Perhaps more importantly, mVSS 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 $t < n/3$ which is more round-efficient than the Feldman-Micali protocol and which also admits a self-contained proof (cf. the full version of this work [?]) that we believe is significantly simpler than that of [?].

Since mVSS is impossible for $t \geq n/2$, our techniques do not extend to the case of dishonest majority. It remains an interesting open question as to whether it is possible to achieve broadcast for $n/2 \leq t < n$ in expected constant rounds.

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

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

broadcast protocol as a sub-routine. If the original (outer) protocol uses multiple invocations of the broadcast channel, and these invocations are each emulated using a *probabilistic* broadcast protocol, subtle issues 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 [?,?]. There are also techniques available for handling sequential composition of protocols without simultaneous termination [?,?]; however, this work applies only to the case $t < n/3$ [?] or else is rather complex [?]. As an additional contribution, we show an extension of previous work [?] that enables sequential composition for $t < n/2$ (assuming digital signatures and a PKI) in a simpler and more round-efficient manner than [?]. See Section 5.

The above results, in combination with prior work [?,?], 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. (The constant-round protocol of Goldwasser and Lindell [?], which does not assume a PKI, achieves a weaker notion of security that does not guarantee output delivery.)

1.2 Prior Work on Broadcast/Byzantine Agreement

In a synchronous network with pairwise authenticated channels and no additional set-up assumptions, BA among n parties is achievable iff the number of corrupted parties t satisfies $t < n/3$ [?,?]; furthermore, in this case the corrupted parties may be computationally unbounded. In this setting, a lower bound of $t + 1$ rounds for any deterministic BA protocol is known [?]. A protocol with this round complexity — but with exponential message complexity — was shown by Pease, et al. [?,?]. Following a long sequence of works, Garay and Moses [?] show a fully-polynomial BA protocol with optimal resilience and round complexity.

To circumvent the above-mentioned lower bound, researchers beginning with Rabin [?] and Ben-Or [?] explored the use of randomization to obtain better round complexity. This line of research [?,?,?,?] culminated in the work of Feldman and Micali [?], who show a randomized BA protocol with optimal resilience $t < n/3$ that runs in an expected *constant* number of rounds. Their protocol requires channels to be both private and authenticated (but see footnote 1 and the next section).

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 digital signatures and a public-key infrastructure (PKI); recall that protocols in this setting are termed *authenticated.* Implicit in this setting is that the adversary is computationally bounded (so it cannot forge signatures), though if information-theoretic "pseudo-signatures" [?] are used security can be obtained

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

even against an unbounded adversary. Pease, et al. [?,?] show an authenticated broadcast protocol for $t < n$, and a fully-polynomial protocol achieving this resilience was given by Dolev and Strong [?]. These works rely only on the existence of digital signature schemes and a PKI, and do not require private channels.

The $(t+1)$-round lower bound for deterministic protocols holds in the authenticated setting as well [?], and the protocols of [?,?,?] meet this bound. Some randomized protocols beating this bound for the case of $n/3 \leq t < n/2$ are known [?,?,?], but these are only partial results:

- Toueg [?] gives an expected $O(1)$-round protocol, but assumes a trusted dealer. After the dealing phase the parties can only run the BA protocol a *bounded* number of times.
- A protocol by Bracha [?] implicitly requires a trusted dealer to ensure that parties agree on a "Bracha assignment" in advance (see [?]). 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 [?], building on [?,?], 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 [?], building on [?,?,?], 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, some appropriately-homomorphic public-key encryption scheme) and not signatures alone. We remark that, 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 Technical Preliminaries

By a *public-key infrastructure* (PKI) in a network of n parties, we mean that prior to any protocol execution 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 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.

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 are necessary for our work, but can be realized using signature schemes if one is willing to assume a PKI. (Note that signatures 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). For adaptive adversaries, more complicated solutions are available [**?**,**?**] 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 also assume that the adversary cannot forge a new valid signature on behalf of any honest party (this is the authenticated case). Using a standard hybrid argument and assuming the existence of one-way functions, 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 often omit for simplicity additional information that must be signed along with the message. Thus, 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; cf. [**?**]).

Byzantine agreement. We focus on Byzantine agreement, which readily implies broadcast. The standard definitions of BA and broadcast follow.

Definition 1. *(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:*

- *All honest parties output the same value.*
- *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. $\diamondsuit$

Definition 2. *(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:*

- *All honest parties output the same value.*
- *If the dealer is honest, then all honest parties output M.* $\diamondsuit$

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 [?]. We develop both protocols in parallel so as to highlight the high-level similarities in each.

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

Gradecast. *Gradecast*, a relaxed version of broadcast, was introduced by Feldman and Micali [?]; below we provide a definition which is slightly weaker than the one in [?, Def. 11], but suffices for our purposes.

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 eventually terminates the protocol and outputs both 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 grade $g_i = 2$, then the output of every honest party P_j satisfies $m_j = m_i$ and $g_j \geq 1$.* $\diamondsuit$

The first result that follows is due to [?]. A proof of the second result can be found in the full version of this paper [?].

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

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

Verifiable Secret Sharing (VSS). *VSS* [?] extends the concept of secret sharing [?,?] to the case of Byzantine faults. Below we provide what is essentially the standard definition except that for technical reasons (namely, security under parallel composition) we explicitly incorporate a notion of "extraction."

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 players is independent of the dealer's input s.*

Extraction *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.* $\diamondsuit$

The first result that follows is well known (see, e.g., [**?**,**?**]). The second result follows readily from existing results; see the full version for a proof [**?**].

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

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

3.1 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. 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 should set this flag to 1. Furthermore, if any honest party sets this flag to 1 then the protocol should achieve all the properties of VSS (cf. Def. 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 (the* sharing phase*), and a value s_i at the end of the second phase (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 players may learn the dealer's secret in the sharing phase.

The main result of this section is the following, which holds for any $t < n$:

Theorem 1. *Assume there exists a constant-round VSS protocol Π, using a broadcast channel in the sharing phase only, which tolerates t malicious parties. Then there exists a constant-round moderated VSS protocol Π', using a gradecast channel, which tolerates 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 "simulated broadcast" 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. First, note 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 simulated broadcast 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 simulated broadcast 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 simulated broadcast subroutine) is honest then $g_i = 2$ and $m_i = m$. 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. ■

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

Corollary 1. *There exists a constant-round protocol for moderated VSS tolerating $t < n/3$ malicious parties.*

Corollary 2. *There exists a constant-round protocol for authenticated moderated VSS tolerating $t < n/2$ malicious parties.*

3.2 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 [**?**]:

Definition 6. *(Oblivious leader election): A two-phase protocol for parties P_1, …, P_n is an* oblivious leader election protocol with fairness δ tolerating t malicious parties *if each honest party P_i outputs a value $v_i \in [n]$, 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.*

If the above event happens, we say an honest leader was elected. ♢

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

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

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

Theorem 2. *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 Each party P_i begins with $\mathsf{trust}_{i,j} = 1$ for $j \in \{1, \ldots, n\}$.

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 $\mathsf{trust}_k \stackrel{\text{def}}{=} \{j : \mathsf{trust}_{k,j} = 1\}$.

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:

$$\mathsf{trusted} = \left\{ k : \begin{array}{c} \text{there exists a } P_i \text{ that was honest at the end of phase 1} \\ \text{for which } k \in \mathsf{trust}_i \end{array} \right\}.$$

If P_i was honest in phase 1, then $i \in \mathsf{trusted}$. Furthermore, by the properties of moderated VSS, if $k \in \mathsf{trusted}$ then for any honest P_i, P_j and any $1 \leq \ell \leq n$, we have $c^i_{\ell,k} = c^j_{\ell,k}$ and hence $c^i_k = c^j_k$; thus, we may freely omit the superscript in this case. We claim that for $k \in \mathsf{trusted}$, the coin c_k is uniformly distributed in $[n^4]$. Let $c'_k = \sum_{\ell \,:\, P_\ell \text{ malicious in phase 1}} c_{\ell,k} \bmod n^4$ (this is the contribution to c_k of the parties that are malicious in phase 1), and let P_i be honest. Since $k \in \mathsf{trusted}$, the properties of VSS hold for all secrets $\{c_{\ell,k}\}_{\ell=1}^{n}$ and thus c'_k is independent of $c_{i,k}$. (If we view moderated VSS as being provided unconditionally, independence holds trivially. When this is instantiated with a protocol for moderated VSS, independence follows from the information-theoretic security of moderated VSS.[4]) It follows that c_k is uniformly distributed in $[n^4]$.

By union bound, with probability at least $1 - \frac{1}{n^2}$, all coins $\{c_k : k \in \mathsf{trusted}\}$ 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 which was honest in phase 1. This concludes the proof. ■

Combining Theorem 2 with Corollaries 1 and 2, we obtain:

Corollary 3. *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 4. *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.)*

3.3 From OLE to Byzantine Agreement

For the unconditional case (i.e., $t < n/3$), Feldman and Micali [?] 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. This also serves as a warmup for the authenticated case.

Theorem 3. *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 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), $\mathtt{lock}_i$ (initially set to ∞), and $\mathtt{accept}_i$ (initially set to $\mathtt{false}$).

Step 1 Each P_i sends b_i to all parties. Let $b_{j,i}$ be the bit P_i receives from P_i. (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 $\mathtt{lock}_i := 0$.
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.

[4] Formally, one could define an appropriate ideal functionality within the UC framework [?] and show that any protocol for moderated VSS implements this functionality (relying on the extraction property in Definition 4); security under parallel composition then follows. One could also appeal to a recent result showing security of statistically-secure protocols under parallel self composition when inputs are not chosen adaptively (as is the case here) [?].

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 $\texttt{lock}_i := 0$.

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 $\texttt{lock}_i = \infty$, then P_i sets $\texttt{accept}_i := \texttt{true}$.

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 $\texttt{accept}_i := \texttt{false}$.

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 $\texttt{accept}_i := \texttt{false}$.

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 $\texttt{accept}_i = \texttt{true}$, then P_i sets $b_i := b_{\ell_i,i}$. If $\texttt{lock}_i = 0$, then P_i outputs b_i and terminates; otherwise, P_i goes to step 1.

We refer to an execution of step 1 through 6 as an *iteration*. First we claim that if an honest P_i sets $\texttt{lock}_i := 0$ 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 $\texttt{lock}_i := 0$ in step 2. (The case when P_i sets $\texttt{lock}_i := 0$ 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 $b_j = 0$. 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 1.

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 $\texttt{lock}_i = 0$. This follows easily (by what we have argued in the previous paragraph) once we show that there exists an honest party who sets $\texttt{lock}_i := 0$ while holding $b_i = b$. 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 $\texttt{lock}_i := 0$ and holds $b_i = 0$ by the end of this step.

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 $b_i = b$, $\texttt{lock}_i = 0$, and $\texttt{accept}_i = \texttt{false}$ (and hence all honest parties output b and terminate the protocol in that iteration). (ii) If an honest party P_i sets $\texttt{lock}_i := 0$ in some iteration, then all honest parties P_j hold $b_j = b_i$ and $\texttt{accept}_j = \texttt{false}$ by the end of step 5 of that iteration. (iii) If an honest party P_i sets $\texttt{accept}_i := \texttt{false}$ in some iteration, then all honest parties P_j hold $b_j = b_i$ by the end of step 5 of that same iteration.

Next, we 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. Note that if P_j fails to receive b_i from P_i, then $P_{i,j}$ is unchanged; thus, if P_i terminates with output $b_i = b$, it can be viewed as if P_i keeps on sending $b_i = b$ in the next iteration. (In particular, note that P_i must have sent

$b_i = b$ in step 5.) Hence it suffices to show that by the end of the (current) iteration, $b_j = b$ for all honest parties P_j. But this is implied by (ii), above.

Finally, we show that if an honest leader[5] 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 $b_i = b_{\ell,i} = b_\ell$ at the end of step 6 of the current iteration. Consider two sub-cases: if all honest P_j hold $\texttt{accept}_j = \texttt{true}$ then this is immediate. Otherwise, say honest P_i holds $\texttt{accept}_i = \texttt{false}$. 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 [**?**]. In Section 4, we show how parallel composition can also be used to achieve the same result without using any additional rounds. Using either approach in combination with the above and Corollary 3, we have:

Corollary 5. *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 [**?**] show a construction of expected constant-round binary Byzantine agreement 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 how to achieve termination. In the proof below, we assume a gradecast channel for simplicity of exposition; in the full version, we improve the round complexity of our protocol by working directly in the point-to-point model.

Theorem 4. *Assume there exist a constant-round authenticated gradecast protocol and a constant-round authenticated OLE protocol with fairness $\delta = \Omega(1)$, both tolerating $t < n/2$ malicious parties. Then there exists an expected constant-round authenticated Byzantine agreement protocol 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 defines the 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 $v_i := v$; otherwise, $v_i := \phi$.
2. If $|\mathcal{S}_i^{v_i}| > n/2$, then $\texttt{lock}_i := 1$.

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

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 $v_i := v$; otherwise, $v_i := \phi$.

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 P_i sets $v_i := v_{\ell_i,i}$.

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 sets $\texttt{lock}_i := 1$, then v_i remains unchanged for the remainder of the protocol and P_i outputs v_i (and terminates) at the end of the next 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, then after step 2 of that iteration $v_i = v$ and $\texttt{lock}_i \neq \infty$ for all honest P_i. To see this, note that in this case all honest parties gradecast v in step 1. Furthermore, 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. Assuming honest majority, it follows that $v_i = v$ and $\texttt{lock}_i \neq \infty$ for all honest P_i after step 2, and the claim follows.

Consider the first iteration in which an honest party P_i sets $\texttt{lock}_i := 1$ (in step 2), and let P_j be a different honest party. We show that $v_j = v_i$ by the end of that same iteration (regardless of the outcome of the OLE protocol). To see this, note that by definition of gradecast $\mathcal{S}_j^{v_i} \subseteq \tilde{\mathcal{S}}_i^{v_i}$ and so $v_j = v_i$ after step 2 (whether or not P_j also sets $\texttt{lock}_j := 1$ at this point). Since this holds for all honest parties, every honest party gradecasts v_i in step 3 and thus $|\mathcal{S}_j^{v_i}| > n/2$. It follows that $v_j = v_i$ 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, then all honest P_j will have $\texttt{lock}_j \neq \infty$ by the end of next iteration, and all honest parties will output an identical value and terminate by the end of the iteration after that.

What remains to show is that if an honest leader[6] P_ℓ is elected in some iteration, then all honest parties P_i will hold the same value v_i by the end of that iteration. By what we have argued already above, we only need to consider the case where $\texttt{lock}_i = \infty$ for all honest parties P_i in step 4. Now, 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 another 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) $v_i = v_j$.

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

- If $|\mathcal{S}_j^{v_j}| \leq n/2$, then P_j sets $v_j := v_{\ell,j}$. But, using 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 $v_{\ell,j} = v_\ell = v_i$.

This concludes the proof. ■

Combining Lemma 2, Corollary 4, and Theorem 4 we obtain our main result:

Corollary 6. *There exists an expected constant-round protocol for authenticated Byzantine agreement tolerating $t < n/2$ malicious parties.*

We refer the reader to the full version of this work [?] for an exact calculation of the expected round complexities of our BA protocols.

4 Secure MPC in Expected Constant Rounds

Beaver, et al. [?] and Damgård and Ishai [?] show computationally-secure constant-round protocols Π for secure multi-party computation tolerating dishonest minority, assuming the existence of one-way functions and a broadcast channel (the solution of [?] is secure against an adaptive adversary). To obtain an expected constant-round protocol Π' in the point-to-point model (assuming one-way functions and a PKI), a natural approach is to replace each invocation of the broadcast channel in Π with an invocation of an expected constant-round (authenticated) broadcast protocol bc; i.e., to set $\Pi' = \Pi^{\mathsf{bc}}$. There are two subtle problems with this approach that we must deal with:

Parallel composition. In protocol Π, all n parties may access the broadcast channel in the same round; this results in n parallel executions of bc in protocol Π^{bc}. Although the expected round complexity of *each* execution of bc is constant, the expected number of rounds for *all* n executions of bc to terminate may no longer be constant.

A general technique for handling this issue is proposed by [?]; their solution is somewhat complicated. In our case, however, we may rely on an idea of Fitzi and Garay [?] that applies to OLE-based protocols such as ours. 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 of these sub-routines. Using this approach, the expected round complexity for n parallel executions will be identical to the expected round complexity of a single execution. We defer to the full version a detailed exposition of their method as applied to our protocol.

Sequential composition. A second issue is that protocol bc does not provide simultaneous termination. (As noted in [?], this is inherent for any expected constant-round broadcast protocol.) This may cause problems both in the underlying protocol Π as well as in sequential executions of bc within Π^{bc}.

Existing methods for dealing with this issue either apply only in the case $t < n/3$ [?] or else are rather complex [?]. In the full version [?], we show a simple way to deal with sequential composition for the case of $t < n/2$ (assuming digital signatures and a PKI); a high-level overview is given in Section 5.

Taking the above into account, we obtain the following (security without abort is the standard notion of security in the case of honest majority; see [?]):

Theorem 5. *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 with a PKI. The protocol is secure* **without** *abort, and tolerates $t < n/2$ adaptive corruptions.*

5 Sequential Composition

We have already noted that certain difficulties arise due to sequential composition of protocols without simultaneous termination (see also the discussion in [?]). As an example of what can go wrong, assume protocol Π (that relies on a broadcast channel) requires a party P_i to broadcast values in rounds 1 and 2. Let $\mathsf{bc}_1, \mathsf{bc}_2$ denote the corresponding invocations of broadcast within the composed protocol Π^{bc} (which runs in a point-to-point network). Then, because honest parties in bc_1 do not terminate in the same round, honest parties may begin execution of bc_2 in different rounds. But security of bc_2 is no longer guaranteed in this case!

At a high level, we can fix this by making sure that bc_2 remains secure as long as all honest parties begin execution of bc_2 within a certain number of rounds. Specifically, if honest parties are guaranteed to terminate bc_1 within g rounds of each other, then bc_2 must remain secure as long as all honest parties start within g rounds. We now show how to achieve this for an arbitrary number of sequential executions of bc, without blowing up the round complexity too much.

Let us first formally define the *staggering gap* of a protocol:

Definition 7. *A protocol Π has* staggering gap *g if any honest parties P_i, P_j are guaranteed to terminate Π within g rounds of each other.*

Let $\mathsf{rc}(\Pi)$ denote the round complexity of a protocol Π. A proof of the following appears in the full version [?]:

Lemma 5. *Let bc be a protocol for (authenticated) broadcast with staggering gap g. Then for any constant $c \geq 0$ there exists a protocol $\mathsf{Expand}'(\mathsf{bc})$ which achieves the same security as bc as long as all honest parties begin execution of $\mathsf{Expand}'(\mathsf{bc})$ within c rounds of each other. Furthermore,*

$$\mathsf{rc}\left(\mathsf{Expand}'(\mathsf{bc})\right) = (2c+1) \cdot \mathsf{rc}(\mathsf{bc}) + 1,$$

and the staggering gap of $\mathsf{Expand}'(\mathsf{bc})$ is 1.

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

To see that the above solves our problem, suppose we want to sequentially compose protocols $\mathsf{bc}_1, \ldots, \mathsf{bc}_\ell$, each having staggering gap g. We simply run ℓ sequential executions of $\mathsf{bc}'_i = \mathsf{Expand}'(\mathsf{bc}_i)$ (setting $c = 1$) instead. Each bc'_i

has staggering gap 1, meaning that honest parties terminate within 1 round of each other. But each bc'_{i+1} is secure as long as honest parties begin execution within 1 round of each other, so things are ok.

Applying this technique to compile a protocol Π (which uses a broadcast channel in each of its ℓ rounds) to a protocol Π' (running in a point-to-point network), we obtain $\mathsf{rc}(\Pi') = \ell \cdot (3 \cdot \mathsf{rc}(\mathsf{bc}) + 1)$. This can be improved slightly [**?**].