Electronic Colloquium on Computational Complexity, Report No. 28 (2006)

On Expected Constant-Round Protocols for Byzantine Agreement

Jonathan Katz*† Chiu-Yuen Koo*‡

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 improve the techniques of Lindell, et al. (PODC '02) for sequential composition of protocols without simultaneous termination (something that is inherent for Byzantine agreement protocols using $o(n)$ rounds). Our approach is simpler and yields more round-efficient protocols.

*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.

‡Supported by NSF Trusted Computing grant #0310751.

ISSN 1433-8092

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* which 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 or when a semi-trusted third party can be assumed), 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 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)[2] [31, 27]; we survey this work in Section 1.2. As discussed there, a seminal result of Feldman and Micali [18] 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 $t < n$ [31, 27, 14], but all existing protocols that assume only a PKI and secure signatures require $\Theta(t)$ rounds.[3] (Recent work of Fitzi and Garay [21] 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 *relying on a broadcast channel*, while a generic construction of this sort for graded VSS seems unlikely. Perhaps more importantly, mVSS provides a conceptually-simpler and more natural 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

[1] Other "abstractions" include the assumptions of private and/or authenticated channels. For 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] In BA, parties begin with an input value, and all honest parties must terminate with identical outputs. Furthermore, if all honest parties initially hold the same input, then they all output this value. It is easy to see (and well known) that in the case of honest majority broadcast can be obtained from BA using one additional round.

[3] The conference version of [18] claims an expected $O(1)$-round solution for $t < n/2$, but we are unaware of any proof of this claim (none appears in Feldman's thesis [16]) and the claim no longer appears in the journal version [18].

is more round-efficient than the Feldman-Micali protocol and which also admits a self-contained proof (included here) that we believe is significantly simpler than that of [18].

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 (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*[4] of the various broadcast sub-protocols arise; see the detailed discussion in Section 4. Parallel composition can be dealt with using existing techniques [4, 21]. As for sequential composition, Lindell, et al. [29] recently showed how to handle the difficulties arising there. As an additional contribution of our paper, we show a different approach for handling sequential composition that is both simpler than the techniques of [29] and also yields more round-efficient protocols. The reader is referred to Section 4 for further discussion.

The above results, in combination with prior work [2, 13], yield the first expected constant-round protocol for secure computation in a point-to-point network which tolerates an adversary corrupting any minority of the parties and is based only on the existence of one-way functions and a PKI. (The expected constant-round protocol of Goldwasser and Lindell [25], which does not assume a PKI, achieves a weaker notion of security which 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 $t < n/3$ [31, 27]; 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 [19]. A protocol with this round complexity — but with exponential message complexity — was shown by Pease, et al. [31, 27]. Following a long sequence of works, Garay and Moses [23] show a fully-polynomial BA protocol with optimal resilience and round complexity.

To circumvent the above-mentioned lower bound, researchers beginning with Rabin [33] and Ben-Or [3] explored the use of randomization to obtain better round complexity. This line of research [6, 10, 17, 15] culminated in the work of Feldman and Micali [18], who show a 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 model is one which assumes digital signatures and a public-key infrastructure (PKI); recall that protocols in this setting are termed *authenticated*. The implicit assumption in this setting is that the adversary is computationally bounded (so that it cannot forge signatures), although information-theoretic "pseudo-signatures" have also been suggested [32]. Pease, et al. [31, 27] show an authenticated broadcast protocol for $t < n$, and a fully-polynomial protocol achieving this resilience was given by Dolev and Strong [14]. 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 [14], and the protocols of [31, 27, 14] meet this bound. Some randomized protocols beating this bound for the case of $n/3 \leq t < n/2$ are known [35, 6, 37], but these are only partial results:

[4] These issues are unrelated to those considered in [28] where the different executions are oblivious of each other. Here, in contrast, 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.

- Toueg [35] gives an expected $O(1)$-round protocol, but assumes a trusted dealer. Also, after the dealing phase the parties can only run the BA protocol a *bounded* number of times.
- A protocol by Bracha [6] implicitly requires a trusted dealer to ensure that parties agree on a "Bracha assignment" in advance (see [17]). Furthermore, the protocol only achieves expected round complexity $O(\log n)$ and tolerates (slightly sub-optimal) $t \leq n/(2+\epsilon)$ for any $\epsilon > 0$.
- Waidner [37], building on [6, 17], 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 $O(\log n)$ to $O(1)$.

The latter two results assume private channels.

Fitzi and Garay [21] (building on [35, 7, 30]) 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 correctly-generated secret key sk_i associated with pk_i. Malicious parties may generate their keys arbitrarily, even dependent on keys of honest parties.

Unless otherwise stated, all our results are in the *point-to-point* model, by which we mean the standard synchronous communication model in which parties communicate using pairwise *authenticated* and *private* channels. Authenticated channels can be realized easily using digital signatures once a PKI is assumed. For static adversaries, private channels can be realized using semantically-secure public-key encryption by having each party P_i send to each party P_j a public key $PK_{i,j}$ for a public-key encryption scheme (using different keys for each sender avoids issues of malleability); this adds only a single round. For adaptive adversaries, more complicated solutions are available [1, 9] but we do not discuss these further. For simplicity, in our proofs we assume *unconditional* authenticated/private pairwise channels with the understanding that these guarantees hold only computationally if the above techniques are used.

When we say a protocol (for broadcast, 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 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 encompass 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 in our proofs 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 information-theoretic pseudo-signatures instead of standard digital signatures, authenticated protocols are secure even against a computationally-unbounded adversary (except with negligible probability).

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 identities of the sender/receiver, (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 protocol (in case multiple sub-protocols are being run [28]). This information is also verified, as appropriate, when the signature is verified.

When we say, e.g., that a protocol is "constant-round," this implies that honest parties terminate in a constant number of rounds regardless of the actions of the adversary.

Byzantine agreement/broadcast. We will focus on constructing protocols for Byzantine agreement, which readily imply protocols for broadcast. The standard definitions 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:

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. $\diamondsuit$

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:

Agreement All honest parties output the same value.

Validity 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 [18]. We develop both protocols in parallel so as to highlight the high-level similarities in each.

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 [18, Def. 11]; we provide a definition which is slightly weaker than theirs 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 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 $g_j \geq 1$. $\diamondsuit$

The following result is due to [18] and proved for completeness in Appendix A.1:

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

In Appendix A.2, we 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.*

Verifiable Secret Sharing (VSS). *VSS* [11] extends the concept of secret sharing [5, 34] to the case of Byzantine faults. Below we provide what is essentially the standard definition.

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' such that all honest parties will output s' at the end of the reconstruction phase. $\diamondsuit$

Note that the value s' referred to in the reconstruction property, above, can be extracted in polynomial time given the view of the honest parties at the end of the sharing phase (this follows easily by considering the adversary who simply aborts during the reconstruction phase).

The first result that follows is well-known (cf. [24, 20]); the second is not explicit in the literature but follows readily from known results. For completeness, proofs appear in Appendices A.3 and A.4.

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

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. 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. 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 (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 a 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, which holds for any $t < n$:

Theorem 5. *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 of all, if the moderator P^{**} is honest during the sharing phase then no honest party P_i 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 the 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 that 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 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.*

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 [21]:

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 $v_i \in [n]$, 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.[5]

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

Our construction of OLE uses a similar high-level approach as the construction of an oblivious common coin from graded VSS [18]. 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 the c_i are 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 values 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.
- Even if P_i is dishonest, as long as 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 was furthermore honest at the end of the sharing phase) with constant probability.

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 We describe an OLE protocol. Each party P_i begins with $\{\mathsf{trust}_{i,j}\}_{j=1}^n$ set to 1.

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):

[5] 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.

All parties execute the sharing phase of a moderated VSS protocol in which P_i acts as the dealer with 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 among all $j \in \mathsf{trust}_k$ (ties are broken arbitrarily).

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

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

Note that 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 $1 \le \ell \le n$, 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 $c'_k = \sum_{\ell:\, P_\ell \text{ is malicious in phase 1}} c_{\ell,k} \bmod n^4$ (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.[6]) 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, with probability at least $\frac{n-t}{n}$ the $j \in \mathsf{trusted}$ minimizing c_j corresponds to an honest party P_j; when this occurs, all honest parties output j. 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 [18] 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 11. *Assume 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.*

[6] Formally, one could define an appropriate ideal functionality within the UC framework [8] and show that any protocol for moderated VSS implements this functionality (see the remark on extraction following Def. 4); security under parallel composition then follows. One could also appeal to a recent result [26] showing security of statistically-secure protocols under parallel self composition when inputs are not chosen adaptively (as is the case here).

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), $\texttt{lock}_i$ (initially set to 0), and $\texttt{accept}_i$ (initially set to `false`).

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 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 $b_{j,i}$ remains 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 $\texttt{lock}_i := 1$.

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 $\texttt{lock}_i := 1$.

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 $\texttt{lock}_i = 0$, 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 $\texttt{lock}_i = 1$, then P_i outputs b_i and terminates; otherwise, P_i goes to step 1.

We refer to an execution of steps 1 through 6 as an *iteration*. We begin by establishing the following: (i) if — immediately prior to any given iteration — there exists a bit b such that $b_i = b$ for all honest P_i, then all honest parties who have not yet terminated will output b and terminate the protocol at the end of that iteration. (ii) If an honest party P_i sets $\texttt{lock}_i := 1$ in some iteration, then all honest parties P_j who have not yet terminated hold $b_j = b_i$ by the end of step 6 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 who have not yet terminated hold $b_j = b_i$ by the end of step 5 of that iteration.

Consider (i) in the case $b_i = b = 0$ for all honest P_i (the case $b = 1$ is analogous). Then[7] $|\mathcal{S}_i^0| \geq n - t$ in step 2 for any honest P_i, and so P_i sets $\texttt{lock}_i := 1$ and holds $b_i = 0$ at the end of this step. In step 3, $|\mathcal{S}_i^1| \leq t$ and so b_i remains unchanged. By a similar argument, each honest P_i sets $\texttt{accept}_i := \texttt{false}$ in step 4, and b_i remains unchanged in step 5. Since $\texttt{accept}_i = \texttt{false}$, the value b_i again remains unchanged in step 6; since $\texttt{lock}_i = 1$, honest party P_i outputs $b_i = b = 0$ and terminates. This proves (i), which also proves validity of the protocol (cf. Def. 1).

[7] This statement is immediate if no honest parties have yet terminated. But even if an honest party P_j had terminated in the previous iteration with output $b_j = b = 0$, then P_i holds $b_{j,i} = 0$ in step 2 of the current iteration (since P_j sent $b_{j,i} = 0$ in step 5 of the previous iteration and sends no value in step 1 of the current iteration).

For (ii), consider the case when an honest P_i sets $\texttt{lock}_i := 1$ in step 2 of some iteration (the case when P_i sets $\texttt{lock}_i := 1$ 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 $b_j = b_i = 0$ at the end of step 2 for any honest P_j. Since this holds for all honest parties, it follows that in step 3 we have $|S_j^1| \leq t$, and so b_j remains 0, for all honest P_j. Arguing now as in the previous paragraph, we see that all honest P_j set $\texttt{accept}_j := \texttt{false}$ in step 4, and b_j remains 0 for all honest P_j at the end of step 5. Since $\texttt{accept}_j = \texttt{false}$, the value b_j remains 0 for all honest P_j at the end of step 6. This proves (ii).

Claim (iii) is proved by an argument exactly analogous to those of the previous paragraphs.

Now consider the first iteration in which an honest party P_i terminates with output $b_i = b$. We show that all honest parties terminate with identical output in either the current iteration or the following one. Since P_i terminated, we must have $\texttt{lock}_i = 1$; but then (ii) shows that every honest party P_j holds $b_j = b_i$ at the end of step 6 of the current iteration. By (i), any honest parties who do not terminate in the current iteration will terminate in the following iteration with output b.

Finally, we show that if an honest leader[8] 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_j holds $\texttt{accept}_j = \texttt{false}$. By (iii), $b_\ell = b_j$ at the end of step 5, and hence all honest parties P_i have $b_i = b_j$ by the end of step 6 (regardless of whether $\texttt{accept}_i$ is $\texttt{true}$ or $\texttt{false}$).

If the OLE protocol elects an honest leader with constant probability, it follows that agreement is reached in an expected constant number of iterations. Since each iteration uses only a constant number of rounds, this completes the proof. ■

When $t < n/3$, any binary Byzantine agreement protocol can be transformed into a (multi-valued) Byzantine agreement protocol using two additional rounds [36]. 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 [21] 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 how to achieve termination. In the proof below, we assume a gradecast channel for simplicity of exposition; in Appendix B, we show how to improve the round complexity of our protocol by working directly in the point-to-point model.

Theorem 13. *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 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:

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

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 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 $v_i := v$; 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 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). 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 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 $\mathcal{S}_i^{v_i} \subseteq \tilde{\mathcal{S}}_j^{v_i}$. Since P_i set $\texttt{lock}_i := 1$ we know that $|\tilde{\mathcal{S}}_j^{v_i}| \geq |\mathcal{S}_i^{v_i}| > n/2$ and so P_j sets $v_j = v_i$ after step 2. Since this holds for all honest parties, every honest party gradecasts v_i in step 3 and thus $|\mathcal{S}_j^{v_i}| > n/2$ in step 4. 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 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[9] P_ℓ is elected in some iteration then all honest P_i hold the same value v_i 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. 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) $v_j = v_i$.

[9] This implies in particular 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 P_j sets $v_j := v_{\ell,j} = v_\ell = v_i$.

This concludes the proof. ■

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 Appendix C, we compute the exact round complexities of our protocols for BA and authenticated BA (after applying some simple optimizations). We also show some simple ways of reducing the *amortized* round complexities when multiple (sequential) executions of our protocols are run; this is important when our protocols are used as sub-routines to implement a broadcast channel within some larger protocol.

4 Secure Multiparty Computation in Expected Constant Rounds

Beaver, et al. [2] and Damgård and Ishai [13] 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 [13] 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:

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 [4]; their solution is somewhat complicated. For our protocols, however, we may rely on an idea of Fitzi and Garay [21] 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.

Sequential composition. A second issue is that bc does not provide simultaneous termination. (As noted in [29], this is inherent for any $o(t)$-round broadcast protocol.) This may cause problems both in the underlying protocol Π as well as in sequential executions of bc within Π^{bc}.

Lindell, et al. [29] suggest a method for dealing with this issue which is also rather complex. In Section 5 we show a new method for dealing with this issue; besides being (in our opinion) simpler, this approach yields protocols which are more round- and communication-efficient as compared to protocols resulting from the approach of Lindell, et al. [29].

In any case, we have the following result (security without abort is the standard notion of security in the case of honest majority; see [25]):

Theorem 15. *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 network with pairwise private/authenticated channels and 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 [29]). As an example of what can go wrong, assume some protocol Π (that assumes a broadcast channel) requires some 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!

We begin by reviewing the solution from [29]. Let us first define the term *staggering gap*:

Definition 7 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.

Lindell, et al. begin with the following result (cf. [29, Lemma 3.1]) stated informally here:

Lemma 16. *Let Π be a protocol with staggering gap g and expected round complexity r. Then for any constant $c \geq 0$ there exists a protocol $\mathsf{Expand}(\Pi)$ which achieves the same security guarantees as Π as long as all honest parties begin execution of $\mathsf{Expand}(\Pi)$ within c rounds of each other. $\mathsf{Expand}(\Pi)$ has expected round complexity $(2c+1) \cdot r$ and staggering gap $c + g \cdot (2c+1)$.*

Suppose we want to sequentially compose protocols $\mathsf{bc}_1, \ldots, \mathsf{bc}_\ell$, each having staggering gap g. A naïve solution is to apply Lemma 16 sequentially ℓ times, setting c in each case to the staggering gap of the preceding protocol (see [29] for details). However, the staggering gap of $\mathsf{Expand}(\mathsf{bc}_i)$ in this case grows as $\Omega\left((2g)^i\right)$, implying that the expected round complexity of $\mathsf{Expand}(\mathsf{bc}_\ell)$ grows exponentially in ℓ. Even if ℓ is constant, this is clearly prohibitive.

Instead, Lindell, et al. propose a way to reduce the staggering gap of each $\mathsf{Expand}(\mathsf{bc}_i)$ to a constant independent of i. Let SBA denote a BA protocol running in expected constant rounds and with constant staggering gap g. At a high level, the solution of [29] is to run SBA every so often during execution of $\mathsf{Expand}(\mathsf{bc}_i)$, with a party setting its input (in SBA) to 1 iff it has terminated its execution of $\mathsf{Expand}(\mathsf{bc}_i)$. If a party outputs 1 in such an execution of SBA, it terminates its execution of $\mathsf{Expand}(\mathsf{bc}_i)$. In this way, the staggering gap of $\mathsf{Expand}(\mathsf{bc}_i)$ is exactly $\mathsf{gap}(\mathsf{SBA}) = g$.

Besides being a somewhat complicated solution, the message complexity increases due to the multiple executions of SBA. The round complexity also increases: say we start with a protocol Π which uses a broadcast channel in each of its ℓ rounds, and compile it to a protocol Π' (running in a point-to-point network) using the techniques of Lindell, et al. as described above. Letting $\mathsf{rc}(\cdot)$ denote the expected round complexity of a protocol, we have

$$\mathsf{rc}(\Pi') = \ell \cdot \left((2g+1) \cdot \mathsf{rc}(\mathsf{bc}) + \mathsf{rc}(\mathsf{SBA})\right), \tag{1}$$

where it is further required (for technical reasons) that $\mathsf{rc}(\mathsf{SBA}) \geq g \cdot (2g+2)$.

We show a simpler technique in which we reduce the staggering gap of each "expanded" broadcast sub-routine to 1 without having to run an expensive BA protocol "on top of" bc, using the following improvement of Lemma 16 for the case of broadcast (a proof appears in Appendix D):

Lemma 17. *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.

Applying the above lemma to again 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 now obtain

$$\mathsf{rc}(\Pi') = \ell \cdot (3 \cdot \mathsf{rc}(\mathsf{bc}) + 1)\,,$$

an improvement over (1).

References

[1] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology — Eurocrypt '92*.

[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)*, 1983.

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

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

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

[7] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantipole: Practical asynchronous Byzantine agreement using cryptography (extended abstract). In *19th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2000.

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

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

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

[11] 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)*, 1985.

[12] 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*.

[13] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology — Crypto 2005*.

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

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

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

[17] 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)*, 1985.

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

[19] M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Info. Proc. Lett.*, 14(4):183–186, 1982.

[20] M. Fitzi, J. Garay, S. Gollakota, C. P. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. To appear in TCC '06.

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

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

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

[24] 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)*, 2001.

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

[26] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. To appear in STOC 2006.

[27] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

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

[29] 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)*, 2002.

[30] J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology — Crypto 2002*.

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

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

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

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

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

[36] R. Turpin and A. B. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–6, 1984.

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

A Additional Proofs

A.1 Proof of Lemma 1

We show a constant-round gradecast protocol (essentially from [18, 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[10] 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 $g_i := 0$.

Let us now 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 the parties. It follows that any honest party P_i outputs $m_i = M$ and $g_i = 2$.

Before proving the second required property, we claim 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

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

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 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 we have shown in the previous paragraph. So, P_j outputs message m_i as required.

A.2 Proof of Lemma 2

We show a constant-round authenticated gradecast protocol tolerating $t < n/2$ malicious parties. (The following is adapted from work of Fitzi and Maurer [22] in a different setting.)

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 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 from P_j in the previous round. If there exist $\ell \geq n/2$ distinct indices $j_1, \ldots, j_\ell$ and a message 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 4, 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 claim 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).

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 $g_j \geq 1$). Furthermore, since any honest parties who sign a message in round 3 sign the *same* message (as argued in the previous paragraph), it follows that $m_j = m_i$.

A.3 Proof of Lemma 3

The following 4-round VSS protocol, due to [24], tolerates $t < n/3$ malicious parties. (Recent work of Fitzi, et al. [20] improves this to 3 rounds but this is 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. We assume a finite field $\mathbb{F}$ with $s \in \mathbb{F}$, $|\mathbb{F}| > n$, and $[n] \subset \mathbb{F}$. In the description that follows, we also 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 (say, 0) in the reconstruction phase.

Sharing Phase

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

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. We claim that the information the malicious parties have about the dealer's secret s at the end of the sharing phase consists entirely of the polynomials sent to the malicious 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. Now, at the end of the sharing phase the adversary knows (from the information sent by the dealer in round 1) the values $\{F(i, j) \mid P_i \text{ or } P_j \text{ malicious}\}$. To prove the claim, we show that the adversary does not have any information about $F(i, j)$ if both P_i, P_j are honest. In particular, if P_i, P_j are honest then round 2 leaks no information about $g_j(i) = F(i, j)$ or $h_i(j) = F(i, j)$ 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.

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 claim (see, e.g., [18, Lemma 2]):

Claim 18. *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). By Claim 18, 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 18 again) there exists a 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:

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.4 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. [12, Section 5]. At a high level, we introduce two modifications: (1) we replace the "information checking" tool in [12] by digital signatures; and (2) the protocol of [12] uses the broadcast channel during the reconstruction phase, and we avoid this by having parties send certain signed messages to each other. In our proof, we then need to consider the case in which parties send different messages to different parties (this does not arise in [12] due to the use of the broadcast channel there). 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 $1 \leq i \leq n$.

Sharing Phase

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.[11]

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 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.

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

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.

Now, for each P_j which is not yet disqualified, party P_i has values $\vec{b}_i^j \stackrel{\text{def}}{=} (b_{j,1}, \ldots, b_{j,n})$ (each of these values was either received from P_j in the previous round or was broadcast by P_j in round 5 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 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 by P_j (in round 5), 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 $g'_i(j) \stackrel{\text{def}}{=} a_{j,i} = b_{j,i}$ 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 by P_i (in round 5), and so $b_{k,i} = F'(k, i)$. Since this holds for at least $t + 1$ honest parties P_i and $\vec{b}_j^k$ 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 $F'(0, 0)$.

B Improved Round Complexity for Authenticated BA

Below, we construct an authenticated Byzantine agreement protocol directly in the point-to-point model (rather than relying on an underlying gradecast protocol). Essentially, this protocol is obtained from the one described in Theorem 13 by unrolling the gradecast protocol and eliminating redundant steps.

Let V be the domain of possible input values, let the input value for party P_i be $v_i \in V$, let $\phi \in V$ be some default value and let $\perp \notin V$. Each P_i begins with an internal variable $\texttt{lock}_i$ set to ∞. To avoid having to say this every time, we make 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).

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.

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[12] 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.

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 19. *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

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

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. By the observation mentioned earlier, 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 (again using the earlier observation) 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.

This completes the proof. ■

C 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 5 and proved in Appendix D).

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 [16].) Even more, we can run multiple invocations of the first phase of the OLE protocol and "save them up" 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 $\ell \stackrel{\text{def}}{=} \lceil r/6 \rceil$ executions of the first phase of the OLE protocol. These are scheduled so the final 5 rounds coincide with steps 1–5 of the first iteration of the Byzantine agreement

protocol.

2. For the remainder of the Byzantine agreement protocol, continually run ℓ parallel executions of the first phase of the OLE protocol in parallel with the steps of the Byzantine agreement protocol itself. 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 $1/(2/3) = 3/2$, 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:

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

Finally, by applying the transformation from Section 3.2 to the VSS protocol of Fitzi, et al. [20] (using the 3-round gradecast protocol from Appendix A.1), we obtain $r = 13$ for the OLE protocol constructed here. The expected round complexity of the Byzantine agreement protocol constructed in this work, then, is at most $10 + 13 = 23$. An additional round is needed to implement broadcast.

The authenticated case ($t < n/2$). We apply an analogous analysis to the protocol given in Appendix B. Now 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 $1/(1/2) + 2 = 4$. Each iteration requires 7 rounds. Furthermore, we again use an additional $r - 6$ rounds before the first iteration begins to run $\lceil r/7 \rceil$ executions of the first phase of OLE. Hence the expected round complexity is at most:

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

Applying the transformation from Section 3.2 to the authenticated VSS protocol described in Appendix A.4 (using the 4-round gradecast protocol from Appendix A.2), we obtain $r = 34$. The expected round complexity of the authenticated Byzantine agreement protocol constructed in this work, then, is at most $22 + 34 = 56$. An additional round is needed to implement broadcast.

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, before the first iteration) each time. (The reason is that these can be run in parallel with the preceding execution of broadcast.) 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 17 to handle the issue of non-simultaneous termination, we obtain:

- For $t < n/3$, excepting the first broadcast, each additional invocation of broadcast incurs an (expected) cost of $3 \cdot 16 + 1 = 49$ rounds.
- For $t < n/2$, excepting the first broadcast, each additional invocation of authenticated broadcast incurs an (expected) cost of $3 \cdot 29 + 1 = 89$ rounds.

D Proof of Lemma 17

We first prove a more general version of the lemma that applies to arbitrary protocols but has weaker parameters:

Lemma 20. *Let Π be a protocol with staggering gap g and expected round complexity r. Then for any constant $c \geq 0$ there exists a protocol $\mathsf{Expand}'(\Pi)$ which achieves the same security guarantees as Π as long as all honest parties begin execution of $\mathsf{Expand}'(\Pi)$ within c rounds of each other. $\mathsf{Expand}(\Pi)$ has expected round complexity $(2c+1) \cdot r + c + g \cdot (2c+1) + 1$ and staggering gap 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 $t < n/2$.

Proof We describe protocol $\mathsf{Expand}'(\Pi)$. It consists of two phases: in the first phase, each party P_i executes $\mathsf{Expand}(\Pi)$. (Recall that $\mathsf{Expand}(\Pi)$ has expected round complexity $(2c+1)r$ and staggering gap $c + g(2c+1)$; cf. Lemma 16.) When P_i terminates execution of $\mathsf{Expand}(\Pi)$ with output v_i, it then begins the second phase:

Unconditional case: In each round (of the second phase), P_i does the following:

- If $c + g(2c+1)$ rounds have passed since terminating $\mathsf{Expand}(\Pi)$, or P_i has received at least $t+1$ copies of "exit" (from $t+1$ distinct parties), then P_i sends "exit" to all parties.
- If P_i has received at least $2t+1$ copies of "exit" (from $2t+1$ distinct parties), then it terminates $\mathsf{Expand}'(\Pi)$ with output v_i.

Authenticated case: In each round (of the second phase), P_i does the following

- If $c + g(2c+1)$ rounds have passed since terminating $\mathsf{Expand}(\Pi)$, then P_i signs "exit" and sends this to all parties.
- If P_i has received $t+1$ valid signatures (from $t+1$ distinct parties) on the message "exit", then it sends "exit" along with these $t+1$ valid signatures to all parties, and terminates $\mathsf{Expand}'(\Pi)$ with output v_i.

We show that the staggering gap of $\mathsf{Expand}'(\Pi)$ is 1. Let round k be the first round in which some honest party P_i terminates $\mathsf{Expand}'(\Pi)$. Then:

Unconditional case: When P_i terminates $\mathsf{Expand}'(\Pi)$ in round k, it has received $2t+1$ copies of "exit", at least $t+1$ of which are from honest parties. Hence all honest parties have received at least $t+1$ copies of "exit" by round k and have sent "exit" by round $k+1$. Since there are at least $2t+1$ honest parties, it follows that all honest parties receive $2t+1$ copies of "exit", and hence terminate $\mathsf{Expand}'(\Pi)$, by round $k+1$.

Authenticated case: When P_i terminates $\mathsf{Expand}'(\Pi)$, it has received $t+1$ valid signatures on "exit" which it forwards to all parties. Hence all honest parties receive the forwarded signatures in round $k+1$ and terminate by that round.

It is easy to see that no honest party terminates $\mathsf{Expand}'(\Pi)$ until all honest parties have terminated $\mathsf{Expand}(\Pi)$ and so the security guarantees of $\mathsf{Expand}'(\Pi)$ are the same as those of $\mathsf{Expand}(\Pi)$, which are the same as those of Π. The claimed round complexity is immediate. ■

We now prove Lemma 17:

Proof The main difference between here and the proof of the previous lemma is that after the execution of $\mathsf{Expand}(\Pi)$ is completed, a party will not wait for $c + g(2c+1)$ rounds before sending "exit". Instead, it will send its output from $\mathsf{Expand}(\Pi)$ to all parties. A party who is convinced

that this received output value is correct may then terminate $\mathsf{Expand}'(\Pi)$ even if it has not yet terminated $\mathsf{Expand}(\Pi)$.

We now describe protocol $\mathsf{Expand}'(\Pi)$ in more detail. Each party P_i first executes $\mathsf{Expand}(\Pi)$. When P_i terminates execution of $\mathsf{Expand}(\Pi)$ with output v_i, it sends "exit, v_i" to all parties (along with a signature of this message in the authenticated case). Furthermore, at every round (including during execution of $\mathsf{Expand}(\Pi)$), P_i does the following:

Unconditional case: If, for some value v, P_i has received "exit, v" from $t+1$ distinct parties (possibly including itself), then P_i sends "exit, v" to all parties. If a party has received "exit, v" from $2t+1$ distinct parties (possibly including itself), then it terminates $\mathsf{Expand}'(\Pi)$ with output v (even if its execution of $\mathsf{Expand}(\Pi)$ has not completed).

Authenticated case: If, for some value v, P_i has received valid signatures from $t+1$ distinct parties (possibly including itself) on "exit, v", then P_i forwards "exit, v" along with these $t+1$ signatures to all parties, and terminates $\mathsf{Expand}'(\Pi)$ with output v.

The analysis of the staggering gap is the same as in the proof of Lemma 20. As regards the security guarantees of $\mathsf{Expand}'(\Pi)$, the claim now is that that no honest party terminates $\mathsf{Expand}'(\Pi)$ until *some* honest parties has terminated $\mathsf{Expand}(\Pi)$. Still, the security guarantees of $\mathsf{Expand}'(\Pi)$ are the same as those of $\mathsf{Expand}(\Pi)$, which are the same as those of Π. This completes the proof. ■

We remark that the proof of Lemma 17 does not apply to *arbitrary* protocols (rather, we have explicitly stated the lemma only for protocols achieving broadcast) since we use the fact that all honest parties should terminate with *identical* outputs.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de/