Efficient Player-Optimal Protocols for Strong and Differential Consensus

Matthias Fitzi*
Department of Computer Science
University of California, Davis
CA 95616, USA
fitzi@cs.ucdavis.edu

Juan A. Garay
Bell-Labs—Lucent Technologies
600 Mountain Ave., Murray Hill
NJ 07974, USA
garay@research.bell-labs.com

ABSTRACT

In this paper we consider the following two variants of the consensus problem. First, the *strong consensus* problem, where n players attempt to reach agreement on a value initially held by *one of the correct players*, despite the (malicious) behavior of up to t of them. (Recall that in the standard version of the problem, the players are also required to decide on one of the correct players' input values, but only when they all start with the same value; otherwise, they can decide on a default.) Although the problem is closely related to the standard problem, the only known solution with the optimal number of players requires exponential computation and communication in the unconditional setting.

Even though the decision would be a value originally held by a correct player, strong consensus allows for a decision value that is the least common among the correct players. We also formulate the δ*-differential consensus* problem, which specifies that the value agreed on must be of a certain plurality among the correct players — specifically, that the plurality of any other value cannot exceed the plurality of the decision value by more than δ.

In this paper we study these problems, and present *efficient* protocols and tight lower bounds for several standard distributed computation models — unconditional, computational, synchronous, and asynchronous.

1. INTRODUCTION

In the consensus problem (aka Byzantine agreement) [27], n players (parties, processors) attempt to reach agreement on their initial values, despite the possible malicious behavior of up to t of them. In this paper we consider two variants of this problem. First, the *strong consensus* problem [31], where the decision value *must be one of the correct players'* input values. In the standard version of the problem, the correct players are also required to decide on one of the correct players' input values, but only when they all start with the same value (the so-called Validity condition); otherwise, they can decide on a default (say, 0). Thus, in the case of binary decisions, both problems — strong consensus and standard consensus — coincide. When the size of the input domain is larger than two, however, the situations differ. First, since the default value might not be one of the correct players' input, new solutions for the strong problem are required. Second, the complexity and bounds of the strong problem also depend on the size of the input/decision space, as shown below. And third, and perhaps most importantly, even though the standard version of the problem has been well researched and efficient (i.e., polynomial-time) solutions with optimal resiliency have been developed [21, 15, 23], this is not the case for strong consensus. Indeed, the only known solution requiring the optimal number of players, presented by Neiger for the unconditional, synchronous setting, requires exponential computation and communication [31].[1] In this paper we study this problem, and present *efficient* algorithms and tight lower bounds for all (unconditional, computational, synchronous, and asynchronous) settings.

Thus stated, strong consensus does not say anything about the initial plurality that the decision value must have among the correct players — in fact, it may very well be that the decision value was held by just one of them. This is somewhat counterintuitive, as one might expect that choosing the most popular value among the correct players is what consensus protocols ought to be doing. It turns out, however, that depending on the number of corrupted players, this might not always be possible. In this paper we also formulate the δ*-differential consensus* problem, which makes the initial plurality of the decision value explicit; specifically, the problem states that the plurality of any other value held by correct players cannot exceed the plurality of the decision value by more than δ. As for strong consensus, we give efficient protocols and tight lower bounds (for n and δ) for δ-differential consensus in the various settings.

*Work partly done while at ETH Zurich, and partly supported by the Packard Foundation and DIMACS' Visitor Program.

[1] Since consensus and broadcast are fundamental instances of — and basic building blocks for — secure multi-party computation [7, 17], we will be using its terminology to describe our results. In that sense, *unconditional security* means that, for some arbitrarily small, but *a priori* fixed, error probability ε, the probability that the protocol fails is at most ε, independently of the adversary's computational power. When $\varepsilon = 0$, security is said to be *perfect*.

It is easy to see that both strong and differential consensus imply standard consensus. However, strong consensus and differential consensus do not directly compare. Strong consensus allows a player to decide on a value that is only held by one single honest player and thus does not imply differential consensus. On the other hand, if no value is held by more than δ honest players, δ-differential consensus allows a player to decide on a value that is not held by any honest player and thus does not imply strong consensus.

1.1 Prior and related work

The strong consensus problem was first (explicitly) formulated by Neiger in [31] for the unconditional setting. He showed a lower bound in the number of players of $n > \max(3, m)t$, where m is the size of the input/decision domain, and presented two protocols for the problem, one of them using the optimal number of players and optimal number of rounds of communication (i.e., $t+1$, a bound that follows from the standard problem) but which requires exponential computation and communication, and one which is efficient, but at the cost of being sub-optimal in the number of players ($n > 2mt$) and using $2t+2$ rounds of communication. Neiger called the requirement that the decision value be one of the correct players' input "Strong Validity."

We already mentioned the standard version of the consensus problem, formulated in [27] for both the unconditional and computational settings. For the former, optimal ($n > 3t$) probabilistic polynomial-time solutions were obtained in [21, 15] for synchronous and asynchronous networks, respectively, and a deterministic player- and round-optimal polynomial-time solution was found in [23].[2] Consensus in the computational setting, on the other hand, has been less researched. An optimal deterministic polynomial-time solution was found early on, in [19].[3] However, only recently an optimal ($n > 3t$) probabilistic protocol using cryptography has been obtained for asynchronous networks [14].[4] This protocol, however, works in the so-called "random oracle" model [5]. In [32], Nielsen shows how to overcome this requirement by presenting a cryptographic coin protocol based on standard computational assumptions. In some cases, we will be using some of the above protocols as building blocks for the solutions to the problems considered here.

All standard consensus protocols mentioned above concentrate on the *binary* agreement problem, since there exists a one-round transformation that reduces a multi-valued agreement problem to the binary problem [37]. This technique, however, has players decide on a "default" value if initial disagreement is detected, which, as mentioned before, might not be any of the correct players' initial value, and hence does not satisfy our stronger requirements. The approach of running $\lceil \log_2 m \rceil$ binary consensus protocols in parallel for $m > 2$, also yields decision values that might not be any of the correct players' inputs (see, e.g., [1]), at least when performed straightforwardly — see the construction in Theorem 2 for a way to overcome this problem.

Further references to uses and refinements of the Strong Validity condition include [16, 18, 29]. Some applications only need that the decision value be a value proposed by one of the players, but not necessarily a correct one. Such a condition was called "External Validity" by Cachin *et al.* [13], who used it in the design of a randomized atomic broadcast protocol.

Regarding work focusing on the plurality of values held by correct players, perhaps the work most related to ours is that of Berman and Garay [10], who study the difficulty of approximating a solution for binary consensus given by an algorithm that "knows" the *discrepancy* in the input configuration (i.e., the difference between 0's and 1's).

1.2 Our results

In this paper we give full characterizations of when both strong and δ-differential consensus are possible in the presence of up to t arbitrary (Byzantine) failures, for all combinations of network (synchronous, asynchronous) and security (unconditional, computational) models. All our protocols are efficient (i.e., polynomial in n, m, and a security parameter, in the computational case) and player-optimal, as well as δ-optimal in the case of differential consensus. Our results are summarized in Table 1 for strong consensus, and Table 2 for differential consensus.

Network	Synchronous		Async.
Security	Uncond.	Comp.	Any
Players	$n > \max(3, m)t^{(\dagger)}$	$n > mt$	$n > (m+1)t$
Rounds	$t+1$ (det.) $O(1)$ (rand.)	$t+1$ (det.) $O(1)$ (rand.)	— $O(m)$ (rand.)

Table 1: Lower bounds in the no. of players for strong consensus, and protocols satisfying these bounds presented in this paper (†: given in [31]).

Network	Synchronous		Async.
Security	Uncond.	Comp.	Any
Players/δ	$n > 3t \wedge \delta = t$	$n > 2t \wedge \delta = t$	$n > 3t \wedge \delta = 2t$
Rounds	$t+1$ (det.) $O(1)$ (rand.)	$t+1$ (det.) $O(1)$ (rand.)	— $O(m)$ (rand.)

Table 2: Lower bounds in the no. of players and δ for δ-differential consensus, and protocols satisfying these bounds presented in this paper.

We now present a brief overview of the techniques leading to our results, starting with strong consensus. For synchronous networks, we already mentioned the lower bound in the number of players for unconditional strong consensus due to Neiger [31]. It turns out that achieving player- and round-optimal strong consensus in polynomial time in this model is relatively straightforward; we start by presenting such a deterministic protocol. We then present another protocol that is also player-optimal, but more efficient, at the cost of requiring more rounds (i.e., $O(t)$). Besides the efficiency gain, this protocol is interesting for two reasons: First, it sets the ground for the randomized protocols to come, and second, it is the result of the sequential composition of two sub-protocols, the second being a carefully-applied standard consensus protocol. Correctness of this

[2] Recall that, by the result of [22], deterministic asynchronous protocols are not possible.

[3] One should note that, in contrast to the unconditional setting, in the computational setting the bounds on the number of players for the single-sender ("Byzantine generals," broadcast) version of the problem and for consensus are different — $t < n$ and $t < \frac{n}{2}$, respectively.

[4] Here we focus on protocols that generate shared randomness — coin tosses — from scratch. Earlier solutions existed in the "beacon" model [35, 36].

composition follows from the observation that if agreement has not yet been reached by the first sub-protocol, then any value in the current execution is valid (meaning it was initially held by one correct player), and therefore a standard protocol can be applied; on the other hand, if agreement is already reached, then the Validity property of the second sub-protocol will preserve it. The first sub-protocol, in a nutshell, works by having in each phase a distinct leader who takes charge of the computation [9]. By selecting such a leader at random, and making the second sub-protocol also a randomized protocol, we obtain a protocol for strong consensus with constant (expected) number of rounds in the synchronous model.

For asynchronous networks, we first show a lower bound of $n > (m+1)t$ for the number of players in the unconditional model. Thus, the case $m = 2$ can be solved by a standard asynchronous consensus protocol for $n > 3t$ [15], so we focus on $m > 2$. In principle, the approach above of selecting a leader at random (of course, after all the necessary messages have been sent, otherwise it would be impossible for a correct player to figure out whether the leader is faulty, or just being late) would also work here, but at the expense of using somewhat more complicated message dispersion primitives, such as Bracha's "asynchronous broadcast" [12], so as to prevent the basic problems pertinent to these networks. Instead, we toss on the multi-valued domain, but we do not follow the commonly-used random value paradigm (e.g., [14, 15, 21]) where players who are undecided adopt the random value produced by the coin. Instead, we use a more conservative approach in which the outcome of the coin is adopted by the correct players only if there is already evidence of a bias in the current configuration towards that outcome. This approach yields a player-optimal protocol for strong consensus in the asynchronous model, as well as a simplified version of randomized protocols for standard binary consensus where, besides the coin toss, only one communication round is needed. Following this approach, however, results in $O(m)$ expected number of rounds, as opposed to the $O(1)$ that choosing a leader at random would yield; one might consider following the latter approach for large values of m.

Computational strong consensus, as it turns out, is easier to achieve. We extend the lower bound in [31] to show that $n > mt$ (resp., $n > (m+1)t$) players are also necessary in synchronous (resp., asynchronous) networks. The fact that the bounds are the same for both settings, unconditional and computational, is of interest, since typically this is not the case for secure multi-party problems (e.g., $n > 3t$ for unconditional protocols [7] vs. $n > 2t$ for cryptographic protocols [24]). As a consequence, except for special cases (e.g., $m = 2$ in synchronous networks), the same unconditional protocols can be used for the corresponding network — and in fact made more efficient, by using cryptographic coin protocols instead of unconditional ones. The results for the computational setting are presented in Appendix A.

We achieve differential consensus for the optimal values of δ and number of players. In synchronous networks this is done by essentially having each player distribute his input value with a standard broadcast protocol. In asynchronous networks, however, this approach fails since a correct player's broadcast does not necessarily terminate. We obtain our asynchronous protocols by adapting binary protocols for standard consensus [15, 14] to multi-valued domains, thereby involving multi-valued instead of binary coin flipping.

2. MODEL(S), DEFINITIONS AND TOOLS

2.1 Models

We consider a network of $P = \{p_1, \ldots, p_n\}$ players (probabilistic polynomial-time Turing machines), connected by secure pair-wise links.[5] We are interested in security (correctness) against an active adversary who may corrupt up to t of the players, $t < n$, meaning that the adversary may make the corrupted players deviate from the protocol in an arbitrarily malicious way; sometimes we will call such an adversary a "t-adversary."

We consider both *unconditional* security, where no assumptions are made about the adversary's computational power, as well as *computational* security, where the adversary's powers are limited to polynomial-time computation. In the computational model, we will also assume that the players have access to a secure digital signature scheme [25], and that the players share a consistent public-key infrastructure (PKI) with respect to the given scheme, so that messages and their originators can be authenticated by the receiving player, and this proof can be relayed to other players.[6] We will use the short-hand *unconditional* — respectively, *computational* — *strong consensus* to refer to the problem in the respective setting.

We consider both *synchronous* and *asynchronous* networks (see, e.g., [28]). In synchronous networks, it is assumed that the players have access to a global clock, and thus the computation of all players can proceed in a lock-step fashion. It is customary to divide the computation of a synchronous network into *rounds*. In each round, players send messages, receive messages, and perform some local computation. In contrast, in asynchronous networks there is no global clock, and the adversary is also allowed to schedule the arrival time of a message sent to every nonfaulty player; of course, corrupted players may not send any message(s). It is only guaranteed that a message sent by a nonfaulty player will *eventually* arrive at its destination. Although an identification of rounds with absolute time is no longer possible, one can still define asynchronous time based on rounds of message exchange. By an adequate indexing of such rounds, the notion of a global round also follows.

2.2 The problems

In the *consensus* problem, n players attempt to reach agreement on a value from some fixed domain $\mathcal{D}$ ($|\mathcal{D}| = m$) despite the malicious behavior of up to t of them. More specifically, every player p_i starts the consensus protocol with an initial value $v_i \in \mathcal{D}$. Every run of the protocol must satisfy (except for some negligible probability):

— *Termination:* All correct players decide on a value.

[5] Secure links are needed by our randomized protocols; our deterministic protocols only require links to be authenticated.

[6] We note that, with the help of the pseudo-signature scheme of Pfitzmann *et al.* [34], unconditional security can in fact be achieved in this setting. Nonetheless, we shall stick to the traditional term "computational security" since it is being widely used for multi-party computation problems. Alternative names for this setting would be "PKI-dependent security" or, simply, "the authenticated setting."

- *Agreement:* If two correct players decide on v and w, respectively, then $v = w$.
- *Validity:* If all correct players have the same initial value v, then all correct players decide on v.

In the *strong consensus* problem (SC for short), the Validity requirement is strengthened to:

- *Strong Validity:* If the correct players decide on v, then v is the initial value of some correct player.

Let v_i be the input to player p_i, $1 \leq i \leq n$. We will use $\mathcal{I}$ to denote the set of initial values held by correct players in a run of the protocol; i.e., $\mathcal{I} = \{v_i \mid p_i \text{ is correct}\}$. A value $v \in \mathcal{I}$ is called *valid*; *invalid* otherwise. Note that if $\mathcal{I} = \mathcal{D}$ then any value in $\mathcal{D}$ is an allowed decision value.

Furthermore, let $\#_g v$, $v \in \mathcal{D}$, denote the plurality of initial value v among the correct players (alternatively, the number of correct players with initial value v), and let $\delta \geq 0$ be an integer. In the *δ-differential consensus* problem (δ-DC for short), the Validity requirement is strengthened to specify that the value finally agreed on must be of certain plurality among the correct players:

- *δ-Differential Validity:* If the correct players decide on v, then there is no value $w \in \mathcal{D}$ such that $\#_g w > \#_g v + \delta$.

In particular, 0-Differential Validity means that the final value must be one of the input values that is most common among the correct players.

2.3 Shared coins

Our randomized protocols will be making use of a (multi-valued) *shared coin* protocol, i.e., a protocol that allows the correct players to produce, by exchanging messages and in the presence of a t-adversary, a reasonably unpredictable coin toss. More formally,

DEFINITION 1: Let $\mathcal{R}$, $|\mathcal{R}| \geq 2$, be the set of possible outcomes, and C a protocol in which each player p_i is instructed to output a value $r_i \in \mathcal{R}$. Then C is a *p-fair shared coin* protocol iff $\forall x \in \mathcal{R}$ and $\forall$ t-adversaries A, in a random execution of C with A

$$\Pr(\forall \text{ correct players } p_i,\ r_i = x) \geq p.$$

We now review the availability of shared coin protocols for the various settings. Shared coin protocols for the unconditional setting for the classes of adversaries that we are considering here have been developed in [21, 20, 30, 15, 4]. The protocol of [21] works in synchronous networks for any $t < \frac{n}{3}$ and is .35-fair for $|\mathcal{R}| = 2$. Although the main specification and analysis of this protocol is for binary coins, a random leader election protocol (i.e., $\mathcal{R} = P$) is also described, and can be modified to work for other cardinalities of $\mathcal{R}$. In [20], Feldman extends the coin protocol in [21] to work in *asynchronous* networks for $t < \frac{n}{4}$. In [15], Canetti and Rabin also present a shared coin protocol for asynchronous networks, but for the optimal $t < \frac{n}{3}$.

In the computational setting, also several shared coin protocols have been developed that work under different assumptions. In [3], Beaver and So present two shared coin protocols based on the Blum-Blum-Shub generator for synchronous networks and $t < \frac{n}{2}$ that are provably secure under the quadratic residuosity assumption, and the intractability of factoring, respectively. In [14], Cachin *et al.* present two protocols for asynchronous networks and $t < \frac{n}{3}$ in the random oracle model based on RSA and on the Diffie-Hellman problem, respectively. In [32], Nielsen shows how to construct a threshold pseudorandom function based on standard computational assumptions (DDH and RSA), which in turn yields a coin tossing protocol that does not use random oracles. Although presented for asynchronous networks, the protocols above can be readily modified to work on synchronous networks with $t < \frac{n}{2}$.

All the shared coin protocols above execute in a fixed (expected) constant number of rounds. In our protocols, we will use `CoinFlip`($\mathcal{R}$) to denote the corresponding shared coin protocol tossing on domain $\mathcal{R}$.

3. STRONG CONSENSUS

3.1 Synchronous networks

In this section, we describe deterministic and randomized synchronous strong consensus protocols for the unconditional model. All the protocols are optimal in the number of players, efficient (i.e., polynomial-time), and require the minimal (some asymptotically) number of rounds. The lower bound in the number of players has been shown by Neiger:

THEOREM 1. [31] *Unconditional strong consensus is not possible in a synchronous network unless $n > max(3, m)t$.*

We now show that achieving strong consensus efficiently (i.e., in polynomial time) using the optimal number of players and in the optimal number of rounds is possible, and, in fact, relatively straightforward. The complexity of the resulting protocol, although polynomial, is quite high: $\Omega(n^8 \log m)$ bit complexity. We then turn to more efficient solutions to the problem which are still player-optimal, but at the expense of using more rounds.

Our player- and round-optimal protocol for strong consensus will be using as a building block the protocol of [23], which is player- and round-optimal for the standard version of the problem, as well as polynomial-time. In particular, we will be using the broadcast (i.e., the "single-sender") version of the protocol.

PROTOCOL 1.
Perform n copies of the [23] broadcast protocol[7] in parallel — one for each player acting as the sender. Upon completion, decide on the value that was received most frequently. If there is a tie, decide on the lowest of all values.

THEOREM 2. *In a synchronous network, if $n > max(3, m)t$ then Protocol 1 achieves unconditional strong consensus in $t + 1$ rounds with polynomial (in n and m) computation and communication.*

[7] Since the protocol in [23] is for binary decisions, $\lceil \log m \rceil$ of them must be performed in parallel in order to cover the whole domain $\mathcal{D}$ ($|\mathcal{D}| = m$). This ($\log m$) factor can be avoided by "augmenting" the protocol of [23] with the Turpin-Coan transformation [37] that allows for multi-valued standard consensus; each player would need to execute only one protocol instead of $\lceil \log m \rceil$, but at the cost of one extra round.

PROOF. First, observe that since $n > 3t$ all the invoked broadcast protocols work correctly, and therefore all the correct players will decide on the same value for each invocation. Therefore, the protocol satisfies the Agreement condition.

Furthermore, since $n > mt$, there must be a value $v \in \mathcal{D}$ that was received strictly more than t times by each correct player. Hence the decision value was tallied more than t times and hence is an input to a correct player. This gives Strong Validity.

Termination in $t+1$ rounds follows directly from the round optimality of the [23] protocol. $\square$

Note that other, more efficient player-optimal protocols can be "plugged into" Protocol 1 instead of the [23] protocol, at the expense of using more rounds (e.g., [2, 11]). For example, using the player-optimal protocol of [11] would result in $O(n^4 \log m)$ bit complexity and $3t + 1$ rounds.[8]

We now turn to "direct" solutions to the problem, which will yield more efficient protocols for small values of m. We call the next protocol Protocol 2 (the figure shows the "code" for player $p_i, 1 \le i \le n$). We use

$$p_i \to P:\ x_i; \quad \textit{receive}\ x_i^{(1)}, \ldots, x_i^{(n)}$$

to denote that player p_i sends the value stored in his local variable x_i to all players, and stores the value received from every player p_j in his variable $x_i^{(j)}$. Analogously, "$p_k \to P:\ x_k$; *receive* $x_i^{(k)}$" denotes the sending of single player p_k. We use "$\stackrel{R}{\leftarrow}$" as a short-hand for the result of choosing an element uniformly at random from a set.

PROTOCOL 2. *(Deterministic SC for sync. networks)*

1. `for` $k = 1$ `to` $t + 1$ `do`
2. $\quad p_i \to P$: v_i; *receive* $v_i^{(1)}, \ldots, v_i^{(n)}$
3. $\quad p_i$: $L_i := \{v \in \mathcal{D} \mid v$ *received more than* t *times*$\}$
4. $\quad p_i \to P$: L_i; *receive* $L_i^{(1)}, \ldots, L_i^{(n)}$
5. $\quad p_i$: $M_i := \{v \in \mathcal{D} \mid v$ *received more than* t *times*$\}$,
   $\quad N_i := \{v \in \mathcal{D} \mid v$ *received at least* $n - t$ *times*$\}$,
   $\quad$ `if` $N_i \neq \emptyset$ `then` $v_i \stackrel{R}{\leftarrow} N_i$
6. $\quad p_k \to P$: v_k; *receive* $v_i^{(k)}$
7. $\quad p_i$: `if` $v_i^{(k)} \in M_i$ `then` $v_i := v_i^{(k)}$
8. `StandardConsensus`(v_i)

Note that Protocol 2 invokes a standard consensus protocol in its last step (step 8). Thus, the protocol consists of the sequential composition of two sub-protocols. The standard protocol in step 8 is needed only for the case $\mathcal{I} = \mathcal{D}$; otherwise, the first sub-protocol (steps 1–7) alone obtains strong consensus. More specifically, the first sub-protocol consists of $t+1$ *phases*, each consisting of three rounds, and follows the "phase king" paradigm for consensus, in which in each phase a distinguished player (p_k) takes charge of the computation [9]. Specifically, in each phase k, after all players build their sets L_i, M_i and N_i (see figure), p_k distributes a value chosen at random from his N_k set. All others players p_i adopt this value *only if* this value already appears in their M_i set. Upon termination of the first sub-protocol, the players run a standard consensus protocol on their (updated) values.

[8] Also note that by using *randomized* standard protocols [21, 6], the round complexity can be reduced (to expected $O(\log n)$ and $O(1)$, respectively); the communication complexity, however, would remain high. We treat randomized solutions separately later.

We first prove a series of lemmas regarding the first sub-protocol (steps 1–7) of Protocol 2 with respect to threshold $n > \max(3, m)t$.

LEMMA 3.1. *For each phase k, $1 \le k \le t+1$, it holds that every correct player p_i holds a valid value v_i (i.e., $v_i \in \mathcal{I}$) at the end of the phase.*

PROOF. Consider the first phase ($k = 1$). If a correct player p_i adopts p_1's value ($v_i := v_i^{(k)}$), then $v_i \in M_i$ and therefore it was received at least $t + 1$ times during step 4. This implies that at least one correct player p_j sent v_i during step 4 and therefore $v_i \in L_j$. Hence, p_j received the value v_i at least $t + 1$ times during step 2 and v_i must be the input value of at least one correct player, i.e., $v_i \in \mathcal{I}$.

Otherwise, if a correct player p_i ignores the value sent by p_1 during step 7, then either v_i is still p_i's input or $v_i \in N_i$. In the former case v_i is trivially valid. In the latter case v_i is valid along the lines of the previous paragraph since $M_i \supseteq N_i$.

For all other phases, the lemma follows by induction since no correct player p_i ever starts a phase with an invalid value. $\square$

LEMMA 3.2. *Assume $\mathcal{I} \neq \mathcal{D}$ (i.e., $|\mathcal{I}| < |\mathcal{D}|$). Then for each phase k, $1 \le k \le t+1$, and for every correct player p_i, it holds that $N_i \neq \emptyset$ after step 5.*

PROOF. By assumption, it holds that $n > mt$ and $|\mathcal{I}| < |\mathcal{D}|$. It follows from Lemma 3.1 that every correct player starts the phase with a valid value. Hence, some value v is held by at least $t+1$ correct players at the beginning of the phase, which means that $v \in L_j$ for every correct player p_j, and hence $v \in N_i$. $\square$

LEMMA 3.3. *Let p_i and p_j be correct players. Then $v \in N_i$ implies $v \in M_j$.*

PROOF. Since $n > 3t$ and p_i counts v $n - t$ times, p_j counts v at least $n - 2t > t$ times. $\square$

LEMMA 3.4. *If $\mathcal{I} \neq \mathcal{D}$ then Protocol 2 achieves strong consensus.*

PROOF.
Agreement: That all the correct players decide on the same value follows from the Agreement property of the second sub-protocol of Protocol 2, standard consensus.
Strong Validity: Consider the first phase k of the first sub-protocol in which p_k is correct (such a phase exists since there are $t + 1$ phases). By Lemma 3.2, p_k distributes some value $v_k \in N_k$ during step 6. By Lemma 3.3, for every correct player p_i, $v_k \in M_i$. Hence, all correct players adopt p_k's value during step 7 and finish the phase with the same value. By Lemma 3.1, v_k is a valid value. No correct player ever changes his value after this phase. This is guaranteed by Lemma 3.1 with $|\mathcal{I}| = 1$ for the remaining phases of the first sub-protocol, and finally by the Validity property of the standard consensus protocol. $\square$

THEOREM 3. *In a synchronous network, if $n > max(3, m)t$ then Protocol 2 achieves unconditional strong consensus in $O(t)$ rounds and $O(n^3 \log m)$ bit complexity.*

PROOF.

Agreement: It follows from the Agreement property of the second sub-protocol, standard consensus.

Strong Validity: If $\mathcal{I} \neq \mathcal{D}$ then Strong Validity follows from Lemma 3.4 and the Validity property of standard consensus. If $\mathcal{I} = \mathcal{D}$ then any output value is allowed, and Strong Validity is trivially achieved.

Complexity: The round complexity of the first protocol (steps 1 to 7) is $O(t)$ and its message complexity is $O(t \cdot n^2 \cdot \frac{n}{t} \log m) = O(n^3 \log m)$ ($O(t)$ phases of 3 rounds each, times at most n^2 messages per round, times at most the maximal size of $|L_i|$, which can contain at most $\frac{n}{t}$ values of size $\lceil \log m \rceil$). Choosing the phase-king protocol of [11] for the final standard consensus protocol does not change the stated complexities. $\square$

Note that by additionally applying the divide-and-conquer approach in [8], the bit complexity can be further reduced to $O(n^2 m)$.

Randomized phase king and applications. We now complete the treatment of the unconditional synchronous setting by pointing out that by

1. electing a "king" at random in the first sub-protocol of Protocol 2 using a shared coin protocol tossing from P, the set of all players, and running this protocol for a constant number R of phases, and
2. using standard phase-king consensus for the second sub-protocol (step 8) wherein, in the same way, a "king" is selected at random in every phase,

strong consensus can be achieved using $n > \max(3, m)t$ players in $O(R)$ rounds and with exponentially small (in R) probability of error. Furthermore, by using standard techniques to detect agreement, such a protocol can also be modified to terminate in expected $O(1)$ rounds.

Protocol 3 describes the protocol. The first sub-protocol (steps 1–7) of Protocol 2 is only modified such that each phase's king is selected at random. Finally, the phase-king protocol in [11] for standard consensus can be easily modified along the same lines and then be used as the second sub-protocol of Protocol 2.

PROTOCOL 3. *(Randomized SC for sync. networks)*

1. `for` $k = 1$ `to` $t + 1$ `do`
2. $p_i \to P$: v_i; *receive* $v_i^{(1)}, \ldots, v_i^{(n)}$
3. p_i: $L_i := \{v \in \mathcal{D} \mid v \text{ received more than } t \text{ times}\}$
4. $p_i \to P$: L_i; *receive* $L_i^{(1)}, \ldots, L_i^{(n)}$
5. p_i: $M_i := \{v \in \mathcal{D} \mid v \text{ received more than } t \text{ times}\}$,
   $N_i := \{v \in \mathcal{D} \mid v \text{ received at least } n - t \text{ times}\}$,
   `if` $N_i \neq \emptyset$ `then` $v_i \xleftarrow{R} N_i$
6. $p_i \to P$: v_i; *receive* $v_i^{(1)}, \ldots, v_i^{(n)}$
7. P: $r_i :=$ `CoinFlip`$(\{1, \ldots, n\})$
8. p_i: `if` $v_i^{(r_i)} \in M_i$ `then` $v_i := v_i^{(r_i)}$
9. `RandPhaseKing`(v_i)

Note that one phase with a correct king in each sub-protocol is sufficient in order to guarantee the protocol's correctness as can be seen from the analysis of Protocol 2. Thus, running each sub-protocol for a sufficient number R of phases will guarantee, except for some negligible probability, that eventually, some correct player is elected as king. This is so since, by using the leader election protocol of [21] as the coin tossing protocol, a good player is chosen as the king during each phase of the first sub-protocol with constant probability (roughly, $\frac{2}{3}$).

Regarding efficiency, we remark that, modulo a small constant factor, this protocol preserves the complexity of solutions to the standard consensus problem. In the randomized case this is significant, as running multiple randomized standard protocols in parallel *à la* Protocol 1 would require some significant additional "machinery" to guarantee that *all* the protocols terminate in $O(1)$ rounds, since, as pointed out in [6], the mathematical expectation of the maximum of n random variables does not necessarily equal the maximum of their expectations.

Furthermore, "randomized phase-king" for standard consensus also improves over the constant-expected-time solution of Ben-Or and El-Yaniv [6] for the related Interactive Consistency problem [33], which runs multiple instances of randomized consensus protocols in parallel. In a nutshell, any polynomial number of randomized phase-king consensus protocols can be run in parallel by only requiring one coin per phase to determine a unique king to propose default values for all parallel protocol instances. This approach also yields an expected $O(1)$-round protocol for δ-Differential Consensus (§4). We leave further details on these applications for the full version of the paper.

3.2 Asynchronous networks

THEOREM 4. *Unconditional strong consensus is not possible in an asynchronous network unless $n > (m + 1)t$.*

PROOF (SKETCH). If $n \leq 3t$ then the impossibility of strong consensus follows from the impossibility for standard consensus [26]. Thus, let us assume that $n > 3t$, and hence, there are at least $2t + 1$ correct players.

Suppose now that $n \leq (m + 1)t$, and that $\mathcal{I} \subset \mathcal{D}$ with $|\mathcal{I}| = m - 1$. Let $C \subset P$ be the set of all correct players, being partitioned into $A \dot{\cup} B = C$ with $|A| \leq (m - 1)t$ and $|B| = t$. Furthermore, assume that for every value $v \in \mathcal{I}$ there are at most t players $p_i \in A$ with input $v_i = v$.

The adversary now makes the t corrupted players start with the same invalid input $w \in \mathcal{D} \setminus \mathcal{I}$. Additionally, she isolates the players in B by delaying all messages originating from B to any other recipient. The players in A together with the t corrupted players must agree on a valid input value. With non-negligible probability they will agree on the invalid value w proposed by the corrupted players since they behave correctly and hence are not distinguishable from the correct players, thus achieving a contradiction. $\square$

We now turn to efficient protocols for this setting. As mentioned in §1.2, we will base our asynchronous protocols on the standard "random value" paradigm, introduced in [35] and used in subsequent randomized protocols. Based on the "list approach" used above (L_i, M_i, etc.), this paradigm leads to the following new version of a synchronous randomized protocol for *standard* (binary) consensus, along the lines of [35; 21, …], from which we will derive our protocol for asynchronous strong consensus. The protocol is shown below; it uses the optimal number of players, $n > 3t$, and achieves consensus in $O(R)$ rounds with exponentially small (in R) probability of error. The proof of correctness is straightforward. Note that, apart from the shared coin, only one round of communication is needed per phase, as opposed to two in [21].

PROTOCOL 4. *(Simplified rand. binary consensus)*

1. `for` $k=1$ `to` R `do`
2. $p_i \to P$: v_i; receive $v_i^{(1)}, \ldots, v_i^{(n)}$
3. p_i: $M_i := \{v \in \mathcal{D} \mid v$ received more than t times.$\}$
4. P: $w_i :=$ `CoinFlip`$(\{0,1\})$
5. p_i: `if` $w_i \in M_i$ `then` $v_i := w_i$

We now build our player-optimal, asynchronous protocol for strong consensus on the approach of Protocol 4. As the case $m=2$ can be solved by a standard randomized consensus protocol for $n > 3t$ [15], we focus on the case $m > 2$, implying that $n > 4t$.[9]

PROTOCOL 5. *(Randomized SC for async. networks)*

1. `for` $k=1$ `to` Rm `do`
2. p_i: $M_i :=$ `BuildCoreSet`(v_i)
3. P: $v :=$ `CoinFlip`$(\{1,\ldots,m\})$
4. p_i: `if` $v \in M_i$ `then` $v_i := v$

The protocol is denoted Protocol 5 (see figure). The only difference (besides being multi-valued) is the way how the set M_i is built (and, of course, how the coin toss is implemented). In Protocol 4, the single round of message exchange guarantees that the set M_i of a correct player only contains valid values, and moreover, that the lists of all the correct players contain a common value, i.e., $\bigcap_{p_i \text{ correct}} M_i \neq \emptyset$. Achieving the same in an asynchronous environment, however, is not straightforward, but possible, as demonstrated by Protocol 6 (BuildCoreSet).

First, Protocol 6 establishes a set of $2t+1$ correct players such that there is a value v that is detected by all those players to be valid (i.e., their sets L_i have a non-empty intersection). Then, the players distribute all the values that they have detected to be valid (i.e., they distribute their sets L_i — step 5). This results in every correct player p_i eventually receiving sets L_j that contain v from more than t different players. Finally, every player collects in his set M_i (returning to Protocol 5) all values v received in more than t different sets L_j. This results in the required property for M_i.

PROTOCOL 6. (`BuildCoreSet`(v_i))

1. p_i: $A_i := T_i^* := S_i^* := \emptyset$
2. $p_i \to P$: v_i
3. p_i: `on` *receive*(v_j): $S_i^{(v_j)} := S_i^{(v_j)} \cup \{j\}$,
   `on` *receive(“confirm$_j$ v”)*: $T_i^{(v)} := T_i^{(v)} \cup \{j\}$,
   `on` *receive(“accept$_j$”)*: $A_i := A_i \cup \{j\}$,
   $\forall v$: `if` $(|S_i^{(v)}| > t)$ `then` $p_i \to P$: “confirm$_i$ v”,
   `if` $(\exists v : |T_i^{(v)}| \geq n-t) \vee (|A_i| > t)$ `then`
   $p_i \to P$: “accept$_i$”,
   `if` $|A_i| \leq 2t$ `then goto` 3 `else goto` 4
4. p_i: $L_i := \left\{v \;\middle|\; |S_i^{(v)}| > t\right\}$
5. $p_i \to P$: L_i; wait for $n-t$ sets L_j from different p_j's
6. p_i: $M_i := \{v \mid \exists_{t+1} j : v \in L_j\}$
7. p_i: `return`(M_i)

[9] The protocol of [15], however, differs from the protocols presented here in that it is a Las Vegas protocol (i.e., there exist input configurations for which there are non-terminating runs), and only tolerates static adversaries. Obtaining a Monte Carlo, adaptive-adversary protocol for the case $m=2$ remains an open problem.

We first prove a series of facts about Protocol 6.

LEMMA 3.5. *If a correct player p_i terminates (exits) Protocol 6, then*

1. *All correct players terminate;*
2. *there is a value $v \in \mathcal{D}$ and a correct player p_j such that $|T_j^{(v)}| \geq n-t$; and*
3. *there is a value v such that $v \in M_j$ for every correct player p_j.*

PROOF.

1. First note that since $m \geq 3$ it holds that $n > 4t$. If p_i terminates, then he has received more than $2t$ “accept” messages, and has also sent an “accept” message himself. Hence there are more than t accepting correct players and every correct player will eventually receive more than t “accept” messages, and thus send an “accept” message. Since every correct player eventually sends an “accept” message, every correct player eventually receives at least $n-2t > 2t$ such messages and terminates.
2. By way of contradiction, assume that $|T_j^{(v)}| < n-t$ for all correct players p_j and all $v \in \mathcal{D}$. This implies that no correct player ever sends an “accept” message and thus that at most t “accept” messages can be received by any correct player. Hence, no correct player terminates.
3. By claim 2 there is some value v such that $|T_j^{(v)}| \geq n-t$ holds for some correct player p_j; i.e., $n-t$ players claim to have received value v more than t times during step 3. This value must have been sent by at least one correct player during step 2, and for at least $n-2t > 2t$ correct players p_k it holds that $|S_k^{(v)}| > t$. Hence, eventually, $v \in L_k$ for more than $2t$ correct players p_k. Finally, during step 5, every correct player receives at least $n-3t > t$ sets containing v, and therefore $v \in \bigcap_{p_\ell \text{ correct}} M_\ell$. □

LEMMA 3.6. *In Protocol 6, if player p_i is correct then every value $v \in M_i$ is valid.*

PROOF. If $v \in M_i$, for correct player p_i, then there is a correct player p_j with $v \in L_j$, and hence $|S_j^{(v)}| > t$, which in turn implies that there is a correct player p_k who sent v during step 2. □

LEMMA 3.7. *There is a correct player that terminates Protocol 6.*

PROOF. By way of contradiction, assume that no correct player terminates. Then, since all values sent by correct players are eventually delivered, every player p_i eventually gets the values v_j from all $n-t$ correct players p_j. Since $n-t > mt$, there is a value v that is sent by more than t correct players and eventually received more than t times by player p_i. Hence, every correct player eventually confirms v, all correct players accept, and finally terminate by receiving more than $2t$ “accept” messages — in contradiction to the assumption. □

We now turn to Protocol 5.

LEMMA 3.8. *In Protocol 5, no correct player ever holds an invalid value.*

PROOF. Consider the first phase of the protocol, and let v be the outcome of the coin-flip protocol. The following holds for every correct player p_i. If $v \notin M_i$, then v_i is still p_i's input and hence valid. Otherwise ($v \in M_i$), $v_i = v$ at the end of the phase, and by Lemma 3.6, this v is the input from a correct player.

For all other phases, the claim follows by induction since all correct players always start the next phase with a valid value. □

LEMMA 3.9. *In Protocol 5, if all correct players start a phase with the same value v, then they all end the phase with that value.*

PROOF. If all correct players start the phase with the same value v, then, during Protocol 6, for all correct players p_i and all values $w \neq v$, it holds that $|S_i^{(w)}| \leq t$, and therefore $w \notin L_i$ and $w \notin M_i$. Hence, by Lemmas 3.5(3) and 3.7, $M_i = \{v\}$ for all correct players p_i, and $v_i = v$ at the end of the phase. □

THEOREM 5. *Let $m > 2$. Then Protocol 5 achieves unconditional strong consensus in an asynchronous network with $n > (m+1)t$ players, in $O(mR)$ rounds and with an error probability exponentially small in R.*

PROOF (SKETCH).
Termination: By Lemmas 3.7 and 3.5(1), in each phase, all correct players proceed to step 3 of the protocol, and the number of phases is limited to Rm.
Agreement: By Lemma 3.5(3) there is a common value in the sets M_j held by all correct players p_j. Hence, at the end of the first phase where the outcome of the coin toss $v \in \bigcap_{p_j \text{ correct}} M_j$ (since there are mR phases, this event occurs with overwhelming probability — see below), all correct players will adopt v at the end of the phase. By Lemma 3.9, this value remains unchanged through the end of the protocol.
Strong Validity: By Lemma 3.8, the common value above is a valid value.
Error probability: Since the coin protocol produces each value with a probability linear in $\frac{1}{m}$, the error probability gets exponentially small in R. □

As before, by having the players detect agreement, Protocol 5 can be made to terminate in expected $O(m)$ rounds.

4. DIFFERENTIAL CONSENSUS

In this section we describe tight bounds on the achievability of differential consensus in synchronous and asynchronous networks for the unconditional setting; the computational setting is treated in Appendix A. It turns out that Protocol 1 (synchronous strong consensus for $n > \max(3, m)t$ also achieves player-, δ-, and round-optimal differential consensus).

THEOREM 6. *In a synchronous network, unconditional δ-differential consensus is impossible if $n \leq 3t$ or $\delta < t$. On the other hand, Protocol 1 achieves (efficient) t-differential consensus for $n > 3t$ in $t+1$ rounds.*

PROOF (SKETCH).
Lower bounds: For any $\delta \geq 0$, δ-differential consensus implies standard consensus since its conditions are more stringent. Thus, since $n > 3t$ is necessary for standard consensus, it is also necessary for δ-differential consensus, and this for any δ. Assume now that $\delta < t$, and consider any two distinct values v, $w \in \mathcal{D}$ such that $\#_g w = \#_g v - \delta - 1$. The adversary can now make all corrupted players input value w but, besides that, make them behave like correct players. Since now value w is proposed by at least the same number of "honest-looking" players, the correct players cannot detect that w is invalid. Thus, $\delta \geq t$ is necessary.
Protocol: Agreement follows by the same argument as in the proof of Theorem 2. Suppose now that the correct players decide on a value $v \in \mathcal{D}$. Then, for each value $w \neq v$, since at most t corrupted players broadcast value w, it must hold that $\#_g v \geq \#_g w - t$. This gives t-Differential Validity. □

The protocol above achieves t-Differential Validity in $t+1$ rounds. Expected $O(1)$ rounds can be achieved using the randomized phase-king approach described in §3.1, i.e., running n copies of randomized phase-king consensus protocols, and using one coin per phase to determine a unique king to propose values for all the parallel protocol instances. The decision process is the same as in Protocol 1. We leave further details for the full version of the paper.

THEOREM 7. *In an asynchronous network, unconditional δ-differential consensus is impossible if $n \leq 3t$ or $\delta < 2t$. On the other hand, a modification of the protocol in [15] achieves $2t$-differential consensus for $t < n/3$ and expected $O(m)$ rounds.*

PROOF (SKETCH).
Lower bounds: Impossibility follows similarly to the proof of Theorem 6. In contrast to the synchronous case, however, $\delta \geq 2t$ is required instead of $\delta \geq t$ since the values from t correct players might be delayed arbitrarily by the adversary.
Protocol: The binary protocol in [15] naturally extends to the non-binary case ($|\mathcal{D}| > 2$) by replacing the binary coin protocol by a $|\mathcal{D}|$-valued coin protocol (similarly to Protocol 5). The majority votings in the "Vote" protocol of [15] guarantee that, given any two values v and w such that $\#_g v > \#_g w + 2t$, w will never be favored. Thus, Agreement and $2t$-Differential Validity follow. □

Acknowledgements

The authors thank the anonymous reviewers for *PODC'03*, Ran Canetti, Ran El-Yaniv, and Tal Rabin for their helpful comments.

5. REFERENCES

[1] A. Bar-Noy, X. Deng, J. Garay, and T. Kameda. Optimal amortized distributed consensus. In *Proceedings of the 5th International Workshop on Distributed Algorithms (WDAG)*, volume 579 of *LNCS*, pages 95–107, Delphi, Greece, Oct. 1991.

[2] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 42–51, Vancouver, British Columbia, Canada, 10–12 Aug. 1987.

[3] D. Beaver and N. So. Global, unpredictable bit generation without broadcast. In *Advances in Cryptology—EUROCRYPT 93*, volume 765 of *Lecture*

Notes in Computer Science, pages 424–434. Springer-Verlag, 1994, 23–27 May 1993.

[4] M. Bellare, J. Garay, and T. Rabin. Distributed pseudo-random bit generators–a new way to speed-up shared coin tossing. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing*, pages 191–200, Philadelphia, PA, May 1996.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, 1993.

[6] M. Ben-Or and R. El-Yaniv. Optimally-resilient interactive consistency in constant time. *Distributed Computing*, 16(2), May 2003.

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

[8] P. Berman, J. Garay, and K. Perry. Bit optimal distributed consensus. In R. Yaeza-Bates and U. Manber, editors, *Computer Science Research*, pages 313–322. Plenum Publishing Corporation, NY, NY, 1992.

[9] P. Berman and J. A. Garay. Asymptotically optimal distributed consensus. In *Proc. 16th International Colloquium on Automata, Languages and Programming (ICALP'89)*, volume 372 of *Lecture Notes in Computer Science*, pages 80–94. Springer-Verlag, July 1989.

[10] P. Berman and J. A. Garay. Adaptability and the usefulness of hints. In A. P. G. Bilardi, G. Italiano and G. Pucci, editors, *Proc. 6th Annual European Symp. on Algorithms–ESA '98*, volume 1461 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998, 271–282 Aug. 1998.

[11] P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 410–415, Research Triangle Park, North Carolina, 30 Oct.–1 Nov. 1989. IEEE.

[12] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Symposium on Principles of Distributed Systems (PODC '84)*, pages 154–162, New York, USA, Aug. 1984. ACM, Inc.

[13] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer-Verlag, Berlin Germany, 2001.

[14] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantitnople: Practical asynchronous byzantine agreement using cryptography. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, Portland, OR, July 2000.

[15] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience (extended abstract). In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 42–51, San Diego, California, 16–18 May 1993.

[16] S. Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. *IC*, 103(1):132–158, July 1993.

[17] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pages 11–19, 1988.

[18] R. De Prisco, D. Malhki, and M. Reiter. On k-set consensus probmems in asynchronous systems. *PADS*, 12(1):7–21, Jan. 2001.

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

[20] P. Feldman. Asynchronous byzantine agreement in constant expected time. Manuscript, 1989.

[21] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, Aug. 1997. Preliminary version in *STOC'88*.

[22] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty processor. *Journal of the ACM*, 32(2):374–382, 1985.

[23] J. A. Garay and Y. Moses. Fully polynomial Byzantine agreement in $t+1$ rounds. *SIAM Journal of Computing*, 27(1):247–290, 1998. Preliminary version in *STOC'93*.

[24] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on the Theory of Computing (STOC)*, pages 218–229, 1987.

[25] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, Apr. 1988.

[26] A. Karlin and A. C. Yao. Probabilistic lower bounds for the byzantine generals problem. Unpublished manuscript.

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

[28] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1996.

[29] M. Merritt, O. Reingold, G. Taubenfeld, and R. Wright. Tight bounds for shared memory systems accessed by byzantine processes. In D. Malki, editor, *16th International Symp. on Distributed Computing (DISC '02)*, Lecture Notes in Computer Science, Toulousse, France, 28–30 Oct. 2002. Springer.

[30] S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 253–266. Springer-Verlag, 1991, 11–15 Aug. 1990.

[31] G. Neiger. Distributed consensus revisited. *Information Processing Letters*, 49(4):195–201, February 1994.

[32] J. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in*

Cryptology — CRYPTO '02, Lecture Notes in Computer Science. Springer-Verlag, 2002.

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

[34] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t >= n/3$. Research report, IBM Research, Nov. 1996. Submitted for Publication.

[35] M. O. Rabin. Randomized Byzantine Generals. In *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science*, pages 403–409, 1983.

[36] S. Toueg. Randomized Byzantine agreements. In *Symposium on Principles of Distributed Systems (PODC '84)*, pages 163–178, 1984.

[37] R. Turpin and B. A. Coan. Extending binary Byzantine Agreement to multivalued Byzantine Agreement. *Information Processing Letters*, 18(2):73–76, Feb. 1984.

APPENDIX

A. COMPUTATIONAL SECURITY

In this setting, it is assumed that the players have access to a secure digital signature scheme [25]; and that the players share a consistent public-key infrastructure (PKI) with respect to the given scheme. Specifically, it is assumed that each player p_i, holds a private signing key SK_i while every other player p_j holds its corresponding verification key VK_i, as given by the signature scheme's key generation algorithm. To send a message m to player p_j, p_i uses the signing algorithm which takes SK_i and m and produces a signature $\sigma_i(m)$; upon receipt, p_j uses the verification algorithm on VK_i, m and $\sigma_i(m)$ to accept or reject the message. Of relevance in multi-party consistency problems is the fact that signed messages can be relayed to other players, who can in turn also verify their validity. Consensus and broadcast protocols in which players sign their messages are sometimes called "authenticated."

A.1 Strong consensus

THEOREM 8. *In a synchronous network, computational strong consensus is (efficiently) achievable if and only if $n > mt$.*

PROOF (SKETCH).
Lower bound: Essentially the same argument as in the proof of Theorem 1 [31] applies here, since after the adversary has chosen an initial value for the corrupted players, they would sign it and behave correctly, making them indistinguishable from the correct players.
Protocol: Recall that consensus and strong consensus are equivalent if $m = 2$. Thus, for this case, the deterministic protocol of Dolev and Strong [19] does the job for $n > 2t$ in $t + 1$ rounds, or alternatively, the randomized protocol of Toueg [36], using any of the cryptographic coin protocols mentioned in Section 2 for this setting — and conditioning the correctness to the computational assumption the coin protocol is based on.

For the case $m > 2$ any unconditional strong consensus protocol (e.g., Protocol 2) can be applied since, for $m > 2$, the bounds coincide. □

THEOREM 9. *In an asynchronous network, computational strong consensus is (efficiently) achievable if and only if $n > (m + 1)t$.*

PROOF (SKETCH).
Lower bound: As for the case of unconditional security, $n > (m + 1)t$ is also necessary in the computational setting as can be proven in the same way as in the proof of Theorem 1.
Protocol: Since the lower bound is the same as for unconditional security, we can basically run the protocols of Section 3.2. However, by applying a cryptographic coin instead of an unconditional one, these protocols can be made more efficient (less interactive). □

Additionally, there is one major advantage of computational strong consensus in asynchronous networks over its unconditional counterpart for the case $m = 2$ and $n > 3t$: Whereas no Monte-Carlo unconditional protocol is known for this case (the protocol of [15] can have non-terminating runs), the protocol of [14] can easily be made to terminate.

A.2 Differential consensus

THEOREM 10. *In a synchronous network, δ-differential consensus is (efficiently) achievable if and only if $n > 2t$ and $\delta \geq t$.*

PROOF (SKETCH).
Lower bounds: Essentially, the argument for $\delta \geq t$ in the proof of Theorem 6 also holds in the computational setting. Since δ-differential consensus implies consensus, and consensus is impossible if $n \leq 2t$, $n > 2t$ also follows.
Protocol: As in Protocol 1, we can have all the players broadcast their initial value using the protocol of [19]; and have the players decide on the value that they received most frequently. Since the protocol of [19] is computationally secure against $t < n$ bad players, Agreement and t-Differential Validity follow analogously to the proof of Theorem 2. □

THEOREM 11. *In an asynchronous network, δ-differential consensus is (efficiently) achievable if and only if $n > 3t$ and $\delta \geq 2t$.*

PROOF (SKETCH).
Lower bounds: As in the case of unconditional security, $\delta \geq 2t$ is also necessary in the computational setting for the same reason (see proof of Theorem 6). Since δ-differential consensus implies consensus, and consensus is impossible if $n \leq 3t$, $n > 3t$ follows.
Protocol: The binary protocol in [14] extends naturally to the non-binary case ($|\mathcal{D}| > 2$) essentially by replacing their binary coin by a $|\mathcal{D}|$-valued coin. Then, the "pre-vote/main-vote" mechanism in [14] guarantees that, given any two values v and w such that $\#_g v > \#_g w + 2t$, w will never be favored. Thus, Agreement and $2t$-Differential Validity follow. □