Efficient and Player-Optimal Strong Consensus

[Preliminary Draft]

Matthias Fitzi* Juan A. Garay†

Abstract

In the *strong consensus* problem, n players attempt to reach agreement on a value initially held by *one of the good players*, despite the (malicious) behavior of up to t of them. Although the problem is closely related to the standard consensus problem (aka Byzantine agreement), the only known solution with the optimal number of players requires exponential computation and communication in the unconditional setting.

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

1 Introduction

Multi-valued distributed decision problems abound. Consider, for example, a financial board trying to reach a decision from {"hold","buy","sell"} (when some of its members might have conflicting interests!), or a distributed system trying to act consistently on the inputs of many sensors. In this paper we consider *strong consensus* [Nei94], a stronger version of the consensus problem [LSP82], where the decision value *must be one of the good parties'* input values. (In the sensor example above, a decision to "launch" the missile had better be based on the input of a nonfaulty, bug-free component!)

Recall that in "standard" consensus the correct parties (processors, "players") are also required to decide on one of the good 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 2, however, the situations differ. Firstly, since the default value might not be one of the good players' input, new solutions for the strong problem are required. Secondly, the complexity and bounds of the strong problem also depend on the size of the input/decision space, as shown below. Finally, and perhaps most importantly, even though the standard version of the problem has been well researched and efficient (polynomial-time) solutions with optimal resiliency have been developed [FM97, CR93, GM98], this is not the case for strong consensus. Indeed, the only known solution requiring the optimal number of players, presented for the unconditional,[1] synchronous setting, requires exponential computation and communication [Nei94].

In this paper we study this problem, and present *efficient* algorithms and tight lower bounds for all (unconditional, computational, synchronous, and asynchronous) settings. Our techniques for the

*Department of Computer Science, Swiss Federal Institute of Technology (ETH), CH-8092 Zurich, Switzerland. E-mail: `fitzi@inf.ethz.ch`. Work partly supported by DIMACS' 1999-2001 Special Focus on Computational Intractability.

†Bell-Labs—Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, USA. E-mail: `garay@research.bell-labs.com`.

[1] *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, indepentenly of the adversary's computational power (see § 2). When $\varepsilon = 0$, security is said to be *perfect*.

randomized version of the problem also give a simplified version of protocols for the standard consensus problem (e.g., [FM97]).

Prior and related work. The strong consensus problem was first (explicitly) formulated by Neiger in [Nei94] for the unconditional setting. He showed a lower bound in the number of processors 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 with the optimal number of processors, but which requires exponential computation and communication. He 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 [LSP82] for both the unconditional and computational settings. For the former, optimal ($n > 3t$) probabilistic polynomial-time solutions were obtained in [FM97, CR93] for synchronous and asynchronous networks, respectively, and a deterministic processor- and round-optimal polynomial-time solution was found in [GM98].[2] (This problem has a long history, and reviewing the many interesting results leading to the optimal solutions is beyond the scope of this paper.) Consensus in the computational setting, on the other hand, has been less researched. An optimal deterministic polynomial-time solution was found early on, in [DS83].[3] However, only recently an optimal ($n > 3t$) probabilistic protocol has been obtained for asynchronous networks by means of an elegant cryptographic coin [CKS00]. It turns out that our probabilistic cryptographic protocol for *synchronous* networks (§ 3.2) is also optimal ($n > 2t$) for the special case $m = 2$.

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 [TC84]. This technique, however, has processors decide on a "default" value if initial disagreement is detected, which, as mentioned before, might not be any of the correct processors' initial value, and hence does not give Strong Validity. The approach of running $\lceil \log_2 m \rceil$ binary consensus protocols in parallel for $m > 2$ (straightforwardly — see § 3.1) also yields decision values that might not be any of the correct processors' inputs [BDGK91].

Finally, we mention that in some applications it is only needed that the decision value be a value proposed by one of the players (a condition called "external validity" in [CKPS01]), but not necessarily a good one.

Our results. In this paper we give full characterizations of when strong consensus is 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) and processor-optimal. Our results are summarized in Table 1.

Protocols & bounds	Synchronous networks		Asynchronous networks	
	Unconditional	Computational	Unconditional	Computational
Processors	$n > \max(3, m)t^{(\dagger)}$	$n > mt$	$n > (m+1)t$	$n > (m+1)t$
Rounds	$t+1$ (det.) $O(1)$ (rand.)	$t+1$ (det.) $O(1)$ (rand.)	— $O(m)$ (rand.)	— $O(m)$ (rand.)

Table 1: Lower bounds in the no. of processors for strong consensus, and protocols satisfying these bounds presented in this paper (†: given in [Nei94])

[2] Recall that, by the result of [FLP85], asynchronous deterministic 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 processors for the single-sender ("Byzantine generals," reliable broadcast) version of the problem and for consensus are different — $t < n$ and $t < \frac{n}{2}$, respectively.

For synchronous networks, we already mentioned the impossibility due to Neiger for the unconditional case [Nei94]. We start by presenting a deterministic protocol that is both processor-optimal and polynomial-time, and runs in the optimal number of rounds (i.e., $t+1$, a bound that follows from the standard problem). We then present another protocol that is also processor-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 results of the sequential composition of two sub-protocols, the second being a carefully-applied standard consensus protocol. Correctness of this 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 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. 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.

In the computational (synchronous) setting, we first extend the lower bound in [Nei94] to show that $n > mt$ processors are also necessary, even when equipped with a secure digital signature scheme. (For the special case $m = 2$, $n > 3t$ is no longer necessary since computational consensus is already achievable with $n > 2t$.) We obtain the deterministic round- and processor-optimal protocol in a way similar to the unconditional setting, and then present a randomized protocol with constant expected number of rounds. Since for every $m > 2$ the bounds on t coincide with the unconditional setting's, no new protocol is needed since we can apply the randomized protocol above. Thus, we give a randomized protocol for $m = 2$ (standard consensus) that is also processor optimal ($n > 2t$).

For asynchronous networks, we first show that the lower bounds on n for both settings are the same; this is of interest since typically this is not the case for secure multi-party problems — i.e., [BGW88] vs. [GMW87]. Turning to protocols, we first note that the case $m = 2$ can be solved by a standard asynchronous consensus protocol for $n > 3t$ [CR93], so we focus on the case $m > 2$. Since the above approach of selecting a leader at random will not work in asynchronous networks, as it would be impossible for a good processor to figure out whether the leader is faulty, or just being late, we modify the coin tossing procedure so that values are chosen instead of players. However, in our randomized protocol we do not follow the "traditional" random value paradigm, introduced in [Rab83] and used in subsequent randomized protocols, where players who are undecided adopt the random value produced by the coin. Instead, we follow a more "conservative gambling" approach in which the outcome of the coin is adopted by the good players only if there is already evidence of a bias in the current configuration towards that outcome. This yields a player-optimal protocol requiring (expected) $O(m)$ rounds in the asynchronous model. We leave achieving strong consensus in asynchronous networks in expected $O(1)$ rounds as an open problem. The approach above also yields a simplified version of randomized protocols for standard consensus (e.g., [FM97]) where, besides the coin toss, only one communication round is needed.

Organization of the paper. The rest of the paper is organized as follows. In § 2 we describe the models of computation, define the strong consensus problem more formally, and describe the properties of a shared coin protocol, a tool that we will be using. § 3 presents lower bounds and protocols for strong consensus in synchronous networks, as well as the simplified randomized scheme for standard consensus, while § 4 presents impossibilities and protocols for asynchronous networks. Summary and open problems are given in § 5.

2 Model(s), Definitions and Tools

Models. We consider a network of $P = \{p_1, \ldots, p_n\}$ processors (probabilistic polynomial-time Turing machines; "players"), connected by secure pair-wise links.[4] 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 [GMR88], so that messages and their originators can be authenticated by the receiving player, and this proof can be relayed to other players. We will use the short-hands *unconditional* — resp., *computational* — *strong consensus* to refer to the problem in the respective setting.

We consider both *synchronous* and *asynchronous* networks (see, e.g., [Lyn96]). 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 processor; of course, corrupted players may not send any message(s). It is only guaranteed that a message sent by a nonfaulty processor wil *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 exchanges. By an adequate indexing of such rounds, the notion of a global round also follows.

The problem. In the *strong consensus* problem, n players attempt to reach agreement on a value from some fixed domain $\mathcal{D}$ initially held by *one of the good players* despite the (malicious) behavior of up to t of them. Let $m = |\mathcal{D}|$. More specifically, every player p_i starts the strong consensus protocol with an initial value $v_i \in \mathcal{D}$. Every run of the protocol is such that the following conditions are satisfied:

- **Termination:** All good players decide on a value.
- **Agreement:** If two good players decide on v_i and v_j, respectively, then $v_i = v_j$.
- **Strong Validity:** If a good player decides on v, then v is the initial value of some good player.

These conditions are required to hold except with some negligible probability.
In contrast, in the standard consensus problem [LSP82], the third condition is replaced by:

- **Validity:** If all good players have the same initial value v, then all good players decide on v.

Observe that Strong Validity implies Validity, and that in the special case $m = 2$, the two problems are equivalent.

Let v_i be the input to player p_i, $1 \le i \le n$. We will use $\mathcal{I}$ to denote the initial configuration in a run of the protocol, i.e., $\mathcal{I} = \{v_i \,|\, p_i \text{ not corrupted during the protocol}\}$. Note that if $\mathcal{I} = \mathcal{D}$, then any value in $\mathcal{D}$ is an allowed decision value. A value $v \in \mathcal{I}$ is called *valid*; *invalid* otherwise.

Shared coins. Our randomized protocols will be making use of a (multi-valued) *shared coin* protocol, i.e., a protocol that allows the good players to produce, by exchanging messages and in the presence of a t-adversary, a reasonably unpredictable coin toss. More formally, we extend the definition of [FM97] to multi-valued coins:

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

Definition 1: Let $\mathcal{R}$, $|\mathcal{R}| \geq 2$, be the set of possible outcomes, and C a protocol in which each processor 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

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

In the unconditional setting, shared coin protocols have been developed in [FM97, Fel89, MR90, CR93, BGR96]. The protocol of [FM97] works in synchronous networks for any $t < \frac{n}{3}$.[5] In [MR90], the fairness of this protocol for binary coins is improved (to $\frac{1}{2}$). In [Fel89], Feldman extends the [FM97] protocol to work in *asynchronous* networks for $t < \frac{n}{4}$. In [CR93], 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 [BS93], 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 [CKS00], Cachin *et al.* present two protocols for asynchronous networks and $t < \frac{n}{3}$ in the random oracle model[6] based on an RSA threshold-signature scheme, and on the Diffie-Hellman problem, respectively. These last two protocols can be readily adapted 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 (unconditional, computational, synchronous, asynchronous) tossing on domain $\mathcal{R}$.

3 Strong Consensus in Synchronous Networks

In this section, we present deterministic and randomized synchronous strong consensus protocols for both unconditional and computational settings. All protocols are optimal in the number of players, efficient, and require the minimal (some asymptotically) number of rounds. The lower bound in the number of players for the unconditional case has been shown by Neiger [Nei94]; we also present a lower bound in the number of players for the computational setting.

3.1 Unconditional security

We start by re-stating the lower bound found by Neiger for this setting.

Theorem 1 [Nei94] *Unconditional strong consensus is not possible in a synchronous network unless* $n > \max(3, m)t$.

The basic argument behind this result is that if $n = mt$ and the initial configuration of initial values does not include all possible m values, then the t bad players may behave as if their initial value is the missing value, and there is no way for the good players to distinguish this situation from one where the value is valid. Although this impossibility result is proven for deterministic protocols, it also holds when randomization is used. (We leave the details for the full version of the paper.) Also note that the lower bound of $t+1$ rounds for standard consensus [FL82] also holds for strong consensus, as the latter is a special case of the former. We now turn to protocols achieving these bounds.

Theorem 2 *Let* $n > \max(3, m)t$. *Then unconditional strong consensus is achievable in a synchronous network in* $t+1$ *rounds with polynomial (in* n *and* m*) computation and communication.*

[5] Although the main specification and analysis of the [FM97] coin 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 $\mathcal{R}$.

[6] Although the protocols presented in [CKS00] are analyzed and proven secure in the random oracle model, using the threshold pseudorandom function construction of Nielsen [Nie02], a coin tossing protocol can be obtained under standard computational assumptions.

Proof: First note that since when $m = 2$ the two problems – standard consensus and strong consensus – are equivalent, the protocol of [GM98] yields the result for this case. So let $m > 2$ and consider the following protocol:

> **Protocol 1.** Perform n player- and round-optimal multi-valued broadcast protocols in parallel — one for each player as the sender.[7] Upon completion, decide on the value received most frequently.

First, observe that since $n > 3t$ all broadcast protocols work correctly, and therefore all the good players will decide on the same value. Hence 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 good player. Hence the decision value was tallied more than t times and hence is an input to a good player. This gives Strong Validity. ■

Alternatively, by "augmenting" the standard protocol of [GM98] with the Turpin-Coan transformation [TC84] that allows for multi-valued standard consensus, each player needs to execute only one protocol, instead of $\lceil \log m \rceil$, at the cost of one extra round. Although polynomial, the complexity of Protocol 1 is quite high (say, $\Omega(n^8 \log m)$ message complexity) due to the complexity of the standard protocol. Other, more efficient player-optimal protocols can be "plugged in," at the expense of using more rounds (e.g., [BDDS87, BGP89]). As an example, using the protocol of [BGP89] would result in $O(n^4 \log m)$ message complexity and $3t + 1$ rounds.[8]

Protocol 2. (Uncond. deterministic protocol for synchronous networks)

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

We now turn to "direct" solutions to the problem, which will yield a more efficient protocol for small values of m. Besides efficiency, the new protocol and its analysis will illustrate the basic technique and set the tone for the subsequent randomized and asynchronous protocols for strong consensus, as well as for the simplification to standard consensus protocols. We call the new protocol Protocol 2 (the figure shows the "code" for player $p_i, 1 \leq i \leq n$, and should be self-explanatory). Note that Protocol 2 invokes a standard consensus protocol in its last setp (step 8). Thus, effectively, the protocol consists of the sequential composition of two sub-protocols. The first sub-protocol (steps 1–7) consists of $t + 1$ *phases*, each consisting of three rounds, and follows the "phase king" paradigm for consensus [BG89], in which in each phase a distinguished player (p_k) takes charge of the computation. Specifically, in each phase k, after all players build their sets, 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.

[7] Note that the binary consensus protocol in [GM98] can be turned into a broadcast protocol (i.e., the "single sender" version of the problem) that still requires $t + 1$ rounds of communication. Furthermore, this broadcast protocol can be extended to work for any finite domain $\mathcal{D}$ ($|\mathcal{D}| = m$) by running $\lceil \log m \rceil$ binary protocols in parallel.

[8] Also note that by using *randomized* standard protocols [FM97, BE98], 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.

Theorem 3 *Protocol 2 achieves unconditional strong consensus in a synchronous network using* $n > \max(3, m)t$ *players,* $O(t)$ *rounds and* $O(n^3 m)$ *message complexity.*

We first prove a series of lemmas regarding the first sub-protocol (steps 1–7) of Protocol 2.

Lemma 3.1 *For each phase* k, $1 \leq k \leq t+1$, *it holds that every good player* p_i *holds a valid value* v_i *at the end of the phase.*

Proof: Consider the first phase ($k = 1$). If a good player p_i ignores the value sent by p_1 during step 7, then v_i is still p_i's input and is hence valid. Otherwise, if p_i adopts p_1's value ($v_i := v_k$), 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 good 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 good player (i.e., $v_i \in \mathcal{I}$).

For all other phases, the lemma follows by induction since no good player p_i ever starts a phase with an invalid value. ■

Lemma 3.2 *Assume* $\mathcal{I} \neq \mathcal{D}$ *(i.e.,* $|\mathcal{I}| < |\mathcal{D}|$*). Then for each phase* k, $1 \leq k \leq t+1$, *and for every good 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 good player starts the phase with a valid value. Hence, some value v is held by at least $t+1$ good players at the beginning of the phase, which means that $v \in L_j$ for every good player p_j, and hence $v \in N_i$. ■

Lemma 3.3 *Let* p_i *and* p_j *be good 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. ■

Lemma 3.4 *Assume* $\mathcal{I} \neq \mathcal{D}$. *Then Protocol 2 achieves unconditional strong consensus under the conditions of Theorem 3.*

Proof (sketch): Persistency: This condition follows from Lemma 3.1, with $|\mathcal{I}| = 1$, and the Validity property of the standard consensus protocol of step 9.
Agreement: Consider the first phase g such that p_g is good (such a phase exists since there are $t+1$ phases). By Lemma 3.2, p_g distributes some value $v_g \in N_g$ during step 6. By Lemma 3.3, for every good player p_i, $v_g \in M_i$. Hence, all correct players adopt p_g's value during step 7 and finish the phase with the same value. It then follows from Persistency that no good player will change his value.
Strong Validity: By Lemma 3.1, the value above is valid. ■

To complete the proof of the theorem, note that if $\mathcal{I} = \mathcal{D}$ then any output value is allowed and hence, consensus is achieved by the standard consensus protocol executed in step 9. The round complexity of the first protocol is $3(t+1)$, and the message complexity follows from inspection and the fact that there exist standard consensus protocols with complexity as low as $O(n^2)$.

We now close the unconditional synchronous setting by presenting a *randomized* protocol for strong consensus that is player-optimal and runs in constant number of rounds.[9] We call this protocol Protocol 3 (refer to the figure). Essentially, we obtain this protocol from Protocol 2 by 1) electing a "king" at random in the first sub-protocol and running this sub-protocol a constant number R of phases; and 2) running a randomized player-optimal standard consensus protocol [FM97] as the second sub-protocol. The correctness relies on the fact that, by using the coin tossing protocol of [FM97], a good player is chosen as the leader during each phase of the first sub-protocol with constant probability (roughly, 2/3). We defer detailed analysis of the protocol and proof to the full version of the paper.

[9] For simplicity, we present here the non-early-stopping version that runs for a fixed number of rounds and achieves exponentially small error probability, and indicate how to turn it into an expected $O(1)$ rounds protocol.

Protocol 3. (Uncond. randomized protocol for synchronous networks)

1. `for` $k = 1$ `to` R `do`
2. $p_i \rightarrow P$: 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. $p_i \rightarrow P$: L_i; receive $L_i^{(1)}, \ldots, L_i^{(n)}$
5. p_i: $M_i := \{v \in \mathcal{D} \mid v \text{ received more than } t \text{ times.}\}$
   $N_i := \{v \in \mathcal{D} \mid v \text{ received at least } n - t \text{ times.}\}$
   $v_i :\in_{\mathcal{R}} N_i$
6. $p_i \rightarrow P$: v_i; receive $v_i^{(1)}, \ldots, v_i^{(n)}$
7. P: $r :=$ `CoinFlip`$(\{1, \ldots, n\})$
8. p_i: `if` $v_i^{(r)} \in M_i$ `then` $v_i := v_i^{(r)}$
9. `StandardRandConsensus`(v)

Theorem 4 *Protocol 3 achieves unconditionally secure strong consensus in a synchronous network with $n > \max(3, m)t$ players, in $O(R)$ rounds and with exponentially small (in R) probability of error.*

Protocol 3 can be modified to terminate in expected $O(1)$ rounds. Having the players re-distribute their N_i sets after Step 5, and "grade-cast" their values in Step 6 (instead of a simple distribute), and slightly modifying the decision rule in Step 8, allows the players to detect that agreement has been reached along the lines of [Rab83, FM97], which, in turn, allows for "early stopping." We leave the details for the full version of the paper.

3.2 Computational security

In this setting, we assume that the players have access to a secure digital signature scheme [GMR88]. Specifically, each player p_i obtains a public key/private key pair (PK_i, SK_i) from the 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 PK_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."

As for the case of unconditional security, $n > mt$ is also necessary in the computational setting in order to achieve strong consensus, as can be proven in a way similar to the proof of Theorem 1. However, for the special case $m = 2$, $n > 3t$ is no longer necessary since computational consensus is already achievable with $n > 2t$.

Theorem 5 *Computational strong consensus is not possible in a synchronous network unless $n > mt$.*

Computational strong consensus can also be achieved by running n standard authenticated broadcast protocols (e.g., [DS83]), as in Protocol 1. Also, a "merged" version along the lines of Protocol 2 is possible. Instead of analyzing those protocol versions, we directly present a new randomized protocol. Since for every $m > 2$, the bounds on t coincide with the unconditional setting's, we can apply the unconditional randomized protocol from the previous section. Thus, it remains to give a protocol for $m = 2$ that is secure for $n > 2t$. We call it Protocol 4 (see figure).

The protocol proceeds along the lines of Protocol 3. In the first round of a phase (step 2), every player distributes his value v_i together with his signature $\sigma_i(v_i, k)$ on v_i and the phase number (the phase number must be included in order to make signatures unique). In the second round (step 3), every player distributes all the signatures he has received. Now a player p_i adopts a value $v_i = b$ if

Protocol 4. (Authenticated randomized protocol for synchronous networks)

1. `for` $k = 1$ `to` R `do`
2. $p_i \;\to\; P$: v_i, $\sigma_i(v_i, k)$
   p_i: $L_i^0 := \{p_j \mid \exists\, \sigma_j(0,k)\}$, $L_i^1 := \{p_j \mid \exists\, \sigma_j(1,k)\}$
3. $p_i \;\to\; P$: all valid $\sigma_j(0,k)$, $\sigma_j(1,k)$ received
   p_i: $M_i^0 := \{p_j \mid \exists\, \sigma_j(0,k)\}$, $M_i^1 := \{p_j \mid \exists\, \sigma_j(1,k)\}$,
   $U_i^0 := M_i^0 \setminus M_i^1$, $U_i^1 := M_i^1 \setminus M_i^0$
4. p_i: `if` ($\exists b : |U_i^{(b)}| \geq n-t$) `then` $v_i := b$ `else` $v_i := 0$
5. $p_i \;\to\; P$: all valid $\sigma_j(0,k)$, $\sigma_j(1,k)$ received
   p_i: $N_i^0 := \{p_j \mid \exists\, \sigma_j(0,k)\}$, $N_i^1 := \{p_j \mid \exists\, \sigma_j(1,k)\}$
6. p_i: `if` ($|L_i^{(v_i)} \setminus N_i^{(1-v_i)}| \geq n-t$) `then` $g_i := 1$ `else` $g_i := 0$
7. $p_i \;\to\; P$: v_i; receive $v_i^{(1)}, \ldots, v_i^{(n)}$
8. P: $r :=$ `CoinFlip`$(\{1, \ldots, n\})$
9. p_i: `if` $g_i = 0$ `then` $v_i := v_i^{(r)}$

he receives $n-t$ valid signatures from different players (including himself) on b, but never a valid signature for $1-b$ by any of these $n-t$ players; otherwise he adopts 0. In the third round (step 5), again, every player distributes all the signatures he has seen so far. Now a player p_i decides on *grade* $g_i = 1$ if and only if there is a subset $S \subseteq P$ of $n-t$ players such that

1. during the first round, all the players in S already sent a valid signature on v_i, and
2. after the third round, p_i has not seen a valid signature on $1-v_i$ from any player in S.

The first condition guarantees that every good player has seen all these signatures on v_i at the end of the second round, since p_i distributed them himself. The second condition guarantees that no good player has seen any such signature on $1-v_i$ after the second round. Hence, every good player must have decided on v_i.

Finally, players distribute their values again (step 7) and elect a "king" p_r at random using a (cryptographic) shared coin protocol; the king's value v_r will serve as the decision value if players are undecided (grade 0). As with Protocol 3, it can be shown that, after a good king p_r is elected, the protocol achieves strong consensus.

Lemma 3.5 *For each phase k of Protocol 4, $1 \leq k \leq R$, it holds that if all the good players p_i start the phase with the same value $v_i = v$, then all the good players end the phase with the same value $v_i = v$.*

Proof: Let $C \subseteq P$ be the set of all the good players at the end of the phase and suppose that all good players p_i start the phase with the same value $v_i = v$. For every such p_i it holds that $C \subseteq L_i^{(v)}$ and $C \subseteq U_i^{(v)}$. Since $|C| \geq n-t$ and $n > 2t$, $|U_i^{(v)}| \geq n-t$ and $|U_i^{(1-v)}| \leq t < n-t$, and all p_i decide on $v_i = v$ during step 4. Furthermore, since $N_i^{(1-v)} \cap C = \emptyset$, it follows that $C \subseteq L_i^{(v)} \setminus N_i^{(1-v)}$, and hence $g_i = 1$, and $v_i^{(r)}$ is ignored by every $p_i \in C$ during step 9. ∎

Lemma 3.6 *For each phase k of Protocol 4, $1 \leq k \leq R$, if a good player p_i sets $g_i = 1$ during step 6 of the phase, then, for all good players p_j, it holds that $v_j = v_i$.*

Proof: Assume that p_i is good and that $g_i = 1$. Then there is a set $C \subseteq P$ such that
(1) $|C| \geq n-t$, and
(2) $C \subseteq L_i^{(v_i)} \setminus N_i^{(1-v_i)}$.

From (2) it follows that $C \subseteq L_i^{(v_i)}$, and hence $C \subseteq M_j^{(v_i)}$ for every good player p_j, since p_i distributed $L_i^{(v_i)}$ in step 3 of the phase. Furthermore, it follows that $C \cap N_i^{(1-v_i)} = \emptyset$, and hence $C \cap M_j^{(1-v_i)} = \emptyset$ for every good player p_j, since every signature corresponding to $M_j^{(1-v_i)}$ reaches p_i during step 5 and would therefore be considered for $N_i^{(1-v_i)}$.

Hence, from (1) together with the facts that $C \subseteq M_j^{(v_i)}$ and $C \cap M_j^{(1-v_i)} = \emptyset$, it follows that $v_j = v_i$. ∎

Lemma 3.7 *Let k_g be the first phase of Protocol 4 in which p_r is good. Then there is a value $v \in \{0,1\}$ such that for all good players p_i, $v_i = v$ at the end of the phase.*

Proof: If, at the end of phase k_g, all good players p_i adopt $v_i := v_i^{(r)}$ then the claim holds. On the other hand, if there is a good player p_i who does not adopt $v^{(r)}$, then $g_i = 1$, and it follows from Lemma 3.6 that $v_i = v_j$ for all good players, hence p_r also distributed $v_r = v_i$. ∎

Theorem 6 *Let $m = 2$ and assume that the players have access to a secure signature scheme and that the assumptions of the cryptographic shared coin protocol hold. Then Protocol 4 achieves computational consensus in a synchronous network with $n > 2t$ players, in $O(R)$ rounds and with an exponentially small (in R) probability of error.*

Proof (sketch): Validity: If all good players start the protocol with the same input value $v_i = v$, then all good players decide on $v_i = v$ at the end of the protocol, as it follows inductively from Lemma 3.5.
Agreement: Given a p-fair computational shared coin protocol tossing on $\{1, \dots, n\}$ with $p \sim \frac{1}{2}$ (see § 2), then for an adequate (but fixed) value of R with overwhelming probability ($> 1 - 2^{-R}$) there is a phase where a good king is elected. By Lemma 3.7, all good players p_i agree on the same value $v_i = v$ at the end of that phase, and it follows from Lemma 3.5 that the agreement persists through the end of the protocol. ∎

3.3 Simplified randomized standard consensus

Before we start the treatment of strong consensus in asynchronous networks, we "interleave" a protocol for standard consensus based on an observation on random selections of independent value; the protocol will also serve as the basis for our asynchronous protocols.

Note that the "random phase king" approach that we used in Protocols 3 and 4 will not work in asynchronous environments, since a corrupted king can always refuse to send a value, which makes him indistinguishable from a correct player with arbitrarily long message delay [FLP85]. Hence we will base our asynchronous protocols on the standard "random value" paradigm, introduced in [Rab83] and used in subsequent randomized protocols. Recall that in that paradigm players who are undecided adopt the random value produced by the coin. However, we will follow here a different approach in which the coin is adopted by the good players only if there is already some bias in the configuration in the coin's direction. Applying this approach yields Protocol 5, a simplified version of the protocol of [FM97] where, apart from the shared coin, *only one round* of communication is needed per phase.

Lemma 3.8 *Protocol 5 achieves unconditional standard ($m = 2$) consensus in a synchronous network with $n > 3t$ players in $O(R)$ rounds and with exponentially small (in R) probability of error.*

Protocol 5. (Simplified randomized standard consensus protocol)

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

Proof (sketch): Persistency: If every correct player starts a phase with the same value b, then, for every good player p_i, $M_i = \{b\}$ after step 3, and all good players finish the phase in agreement on b. Validity follows from persistency.

Agreement: Since there are $n - t > 2t$ good players, during step 2 of every phase there is a value $b \in \{0,1\}$ that is distributed by more than t good players, and hence $b \in M_i$ for every good p_i. If the outcome of the coin is $v = b$ then every good p_i decides on $v_i := b$ and agreement is reached at the end of the phase. For an adequate value of R, this event occurs with overwhelming probability, and will persist. ■

By making the players terminate if M_i contains a unique value (i.e., a value with plurality $> 2t$), the protocol achieves standard consensus in expected $O(1)$ rounds. While the protocol of [FM97] requires two additional rounds (in each phase) to achieve this, no additional communication is required here.

4 Strong Consensus in Asynchronous Networks

4.1 Lower bounds

In contrast to the synchronous model, the bounds on n for both unconditional and computational security in asynchronous networks are the same.

Theorem 7 *Unconditional (resp., computational) 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 [KY]. 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 correxctly and are hence not distinguishable from the correct players. ■

We now give randomized asynchronous protocols for strong consensus that are player-optimal.

4.2 Unconditional security

Our protocol builds on Protocol 5 above. We extend it in order to achieve unconditional strong consensus with optimal resilience in the asynchronous model. Note that, again, the case $m = 2$ can be solved by a standard randomized consensus protocol for $n > 3t$ [CR93]. We thus focus on the case $m > 2$.[10]

Protocol 6. (Uncond. randomized protocol for asynchronous networks)

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

Protocol 7. ($\texttt{BuildCoreSet}(v_i)$)

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

The protocol is denoted Protocol 6 (see figure). The only difference with the (synchronous) Protocol 5 is that the set M_i needs to be built in such a way that it contains only valid values, and that all the M_i's of correct players contain a common value. Achieving this in an asynchronous environment, however, is not straightforward, but possible, as demonstrated by Protocol 7 (BuildCoreSet).

First, Protocol 7 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 good 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 6) all values v received in more than t different sets L_j. This results in the required property for M_i.

As a building block to achieve this we make use of Bracha's asynchronous broadcast protocol [Bra84], called `A-cast` in [Fel89]. `A-cast` is a fixed constant number of rounds protocol that achieves broadcast with termination if the sender is correct. Furthermore, all the players that terminate the protocol agree on their output. We run n `A-cast`'s in parallel (step 2), one for each player as a sender. Note that, although termination is guaranteed by `A-cast` only if the sender is correct, if we apply these n protocols in parallel we cannot guarantee that every good player terminates *all* the protocols with a correct sender, but at most $(n - 2t)$ of them. The reason is that on terminating $(n - t)$ `A-cast`s, a player does not know whether the remaining t protocols involve a corrupted sender that refuses to send any message, or whether they involve a correct sender that is being delayed.

[10] The protocol of [CR93], however, differs from the protocols presented here in that it is a Las Vegas protocol (non-terminating runs), and only tolerates static adversaries. Obtaining a Monte Carlo, adaptive-adversary protocol for this case remains an open problem.

We now prove a series of facts about Protocol 7.

Lemma 4.1 *If a good player p_i terminates (exits) Protocol 7, then*

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

Proof:

1. First note that since $m \geq 3$, then $n > 4t$. 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 good players and every good player will eventually receive more than t "accept" messages, and thus send an "accept" message. Since every good player eventually sends an "accept" message, every good 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 good players p_j. This implies that at most t "accept" messages can be received by any good player, and hence, no good player terminates.
3. By claim 2 there is some value v such that $|T_j^{(v)}| \geq n - t$ holds for some good 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 good player during step 2, and for at least $n - 2t > 2t$ good players p_k it holds that $|S_k^{(v)}| > t$. Hence, eventually, $v \in L_k$ for more than $2t$ good players p_k. Finally, during step 5, every good player receives at least $n - 3t > t$ sets containing v, and therefore $v \in \bigcap_{p_\ell \text{ good}} M_\ell$.

■

Lemma 4.2 *In Protocol 7, for every good player p_i and every value $v \in M_i$, there is a good player p_j that sent v during step 2 of the protocol.*

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

Lemma 4.3 *There is a good player that terminates Protocol 7.*

Proof: By way of contradiction, assume that no good player terminates. But then, since all values sent by good players are eventually delivered, every player p_i eventually gets the values v_j from all $n - t$ good 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 good 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 6.

Lemma 4.4 *In Protocol 6, no good player p_i ever holds an invalid value v_i.*

Proof: Consider the first phase of the protocol, and let v be the outcome of the coin toss protocol. 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 4.2, this v is the input from a good player.

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

Lemma 4.5 *In Protocol 6, if all good players start a phase with the same value v, then they all end the phase with that value.*

Proof: If all good players start the phase with the same value v, then, during Protocol 7, for all good 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 4.1(3) and 4.3, $M_i = \{v\}$ for all good players p_i, and $v_i = v$ at the end of the phase. ■

Theorem 8 *Let $m > 2$. Then Protocol 6 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): Agreement: By Lemmas 4.3 and 4.1(1), in each phase, all good players proceed to step 3 of the protocol. By Lemma 4.1(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 $v \in \bigcap_{p_j \text{ good}} M_j$ (since there are mR phases, this event occurs with overwhelming probability — see below), all good players will adopt v at the end of the phase. By Lemma 4.5, this value remains unchanged through the end of the protocol.
Strong validity: By Lemma 4.4, the common value above is a valid value.
Error probability: Since the coin protocol produces each value with probability $\frac{1}{m}$, the error probability can be estimated as

$$\left(1 - \frac{1}{m}\right)^{mR} = e^{\ln\left(1-\frac{1}{m}\right)^{mR}} \overset{\substack{\text{Clinton} \\ \text{bound}}}{\leq} e^{mR(-\frac{1}{m})} = e^{-R}.$$

■

By having the players detect agreement, Protocol 6 can be transformed into a protocol that terminates in expected $O(m)$ rounds. (Details in the full version of the paper.)

4.3 Computational security

Since in the asynchronous model the lower bounds for the achievability of strong consensus are the same in the computational setting as in the unconditional setting, we can basically run the protocols of § 4.2. By applying a crytographic coin instead of an unconditional one, these protocols can be made more efficient (less interactive). Additionally, there is one major advantage in the computational setting, for the case $m = 2$ and $n > 3t$. Whereas no Monte Carlo protocol is known for this case in the unconditional setting (the protocol of [CR93] can have non-terminating runs), the protocol of [CKS00] can easily be made to terminate.

5 Summary and Open Questions

In this paper, for several standard models for distributed computation, we gave tight bounds on t, the number of actively corrupted players, such that strong consensus on a domain with cardinality m is possible. Achieavability in all these models was constructively demonstrated by efficient protocols.

For synchronous networks, the tight bounds are $n > \max(3, m)t$ for unconditional and $n > mt$ for computational security. For both cases we presented randomized protocols requiring a constant number of communication rounds. For the unconditional case we also presented a round-optimal deterministic protocol, and more efficient protocols at the expense of more rounds.

For asynchronous networks, the bound is $n > (m+1)t$ for both, unconditional and computational security. For the case $m > 2$ we presented an unconditionally secure randomized protocol requiring $O(m)$ rounds. For the case $m = 2$ and unconditional security, the protocol of [CR93] can be applied. However, this protocol's termination is not guaranteed. For the case $m = 2$ and computational security, the protocol of [CKS00] can be slightly modified in order to get a constant-round protocol.

Our unconditional randomized protocols can be shown to be *adaptively secure* except for the case of asynchronous networks and $m = 2$, where we apply the protocol of [CR93]. In contrast, the security of our computational protocols is stated with respect to a static adversary.

The way we flip coins in our asynchronous protocols implies that the number of rounds becomes $\Omega(m)$ instead of $O(1)$. The reason is that the probability of "hitting" a good value is $O(\frac{1}{m})$. It is still an open question whether strong consensus can be achieved in expected constant number of rounds in this model.

References

[BDDS87] 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*, pp. 42–51, Vancouver, British Columbia, Canada, 10–12 Aug. 1987.

[BDGK91] 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*, pp. 95–107, Delphi, Greece, Oct. 1991.

[BE98] M. Ben-Or and R. El-Yaniv. Optimally-resilient interactive consistency in constant time. Manuscript, 1998.

[BG89] 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*, pp. 80–94. Springer-Verlag, July 1989.

[BGP89] P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pp. 410–415, Research Triangle Park, North Carolina, 30 Oct.–1 Nov. 1989. IEEE.

[BGR96] 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*, pp. 191–200, Philadelphia, PA, May 1996.

[BGW88] 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)*, pp. 1–10, 1988.

[Bra84] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Symposium on Principles of Distributed Systems (PODC '84)*, pp. 154–162, New York, USA, Aug. 1984. ACM, Inc.

[BS93] 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*, pp. 424–434. Springer-Verlag, 1994, 23–27 May 1993.

[CKPS01] 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*, pp. 524–541. Springer-Verlag, Berlin Germany, 2001.

[CKS00] 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.

[CR93] 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*, pp. 42–51, San Diego, California, 16–18 May 1993.

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

[Fel89] P. Feldman. Asynchronous byzantine agreement in constant expected time. Manuscript, 1989.

[FL82] M. J. Fischer and N. A. Lynch. A lower bound on the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

[FLP85] 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.

[FM97] 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*.

[GM98] 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*.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[GMW87] 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)*, pp. 218–229, 1987.

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

[LSP82] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Lyn96] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann series in data management systems. Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1996.

[MR90] 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*, pp. 253–266. Springer-Verlag, 1991, 11–15 Aug. 1990.

[Nei94] G. Neiger. Distributed consensus revisited. *Information Processing Letters*, 49(4):195–201, February 1994.

[Nie02] J. Nielsen. A threshold pseudorandom function construction and its applications. In *Advances in Cryptology — CRYPTO '02*, Lecture Notes in Computer Science. Springer-Verlag, 2002.

[Rab83] M. O. Rabin. Randomized Byzantine Generals. In *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 403–409, 1983.

[TC84] R. Turpin and B. A. Coan. Extending binary Byzantine Agreement to multivalued Byzantine Agreement. *Information Processing Letters*, 18(2):73–76, Feb. 1984.