Fast Asynchronous Byzantine Agreement with Optimal Resilience

(Extended Abstract)

Ran Canetti
Dept. of Computer Science
Technion

Tal Rabin [*]
Dept. of Computer Science
Hebrew University

Abstract

The Byzantine Agreement problem is one of the most fundamental problems in the field of distributed computing. However, despite extensive research, a few important questions have remained open.

The *resilience* of a protocol is the maximum number of faults in the presence of which the protocol meets its specification. It is known that no BA protocol for n players (either synchronous or asynchronous) can be $\lceil\frac{n}{3}\rceil$-resilient. The only known asynchronous $(\lceil\frac{n}{3}\rceil - 1)$-resilient BA protocol runs in expected exponential time. A long standing open question is whether there exists a *fast* asynchronous $(\lceil\frac{n}{3}\rceil - 1)$-resilient BA protocol.

We answer this question in the affirmative. We consider a completely asynchronous network of n players, in which every two players are connected via a private channel. Furthermore, we let the faulty players have unlimited computational power. In this setting, we describe an $(\lceil\frac{n}{3}\rceil - 1)$-resilient Byzantine Agreement protocol. With overwhelming probability all the non-faulty players complete the execution of the protocol. Conditioned on the event that all the non-faulty players have completed the execution of the protocol, they do so in constant expected time.

Our construction employs an $(\lceil\frac{n}{3}\rceil - 1)$-resilient Asynchronous Verifiable Secret Sharing (AVSS) scheme. No $(\lceil\frac{n}{3}\rceil - 1)$-resilient AVSS scheme was previously known. Our AVSS scheme employs new asynchronous tools which are of independent interest.

[*]Supported by the Eshkol Fellowship of the Israel Ministry of Education.

1 Introduction

The problem of reaching agreement in the presence of faults is one of the most fundamental problems in the field of distributed computing. A particularly interesting variant of this problem, introduced by Pease, Shostak and Lamport [PSL], allows Byzantine faults (namely, the faulty players may collaborate in any way). A standard formulation of this problem, called the Byzantine Agreement (BA) problem, follows: Design a protocol that allows the non-faulty players to agree on a common value. The agreed value should be the input value of one of the non-faulty players.

The BA problem was investigated in various models, characterized by several parameters (e.g. synchrony, privacy of the communication channels, computational power of the players). Of these, a key parameter is the synchrony of the network. Two important criteria for evaluating agreement protocols are *resilience* and time ("round") complexity. Informally, an n player protocol is t-resilient if the outputs of the non-faulty players meet the specifications of the protocol, as long as up to t of the players are faulty.

A great deal of work concerning BA protocols has been carried out. We refer the interested reader to the surveys of Fischer [F] and Chor and Dwork [CD]. However, despite extensive research a few important questions have remained open. One of these questions is the subject of this paper.

State of the art and our result. Bounds on the resilience of BA protocols were proven by [PSL]. They showed that agreement cannot be reached by a *deterministic* protocol, in an n-player synchronous network, with $\lceil\frac{n}{3}\rceil$ Byzantine faults. Karlin and Yao [KY] generalized this result to randomized protocols. Clearly, these results apply to *asynchronous* networks as well. Furthermore, Fischer, Lynch and Paterson's [FLP] seminal impossibility result for deterministic protocols implies that any (randomized) protocol reaching BA must have non-terminating runs. Bracha [Br] describes an $(\lceil\frac{n}{3}\rceil - 1)$-resilient asynchronous BA protocol which runs in $2^{\Theta(n)}$

expected time. Feldman and Micali [FM] describe a *synchronous* ($\lceil\frac{n}{3}\rceil - 1$)-resilient BA protocol, which runs in constant expected time. Feldman [Fe] generalizes the [FM] construction to an asynchronous setting, yielding a constant expected time, ($\lceil\frac{n}{4}\rceil - 1$)-resilient asynchronous BA protocol. These works allow computationally unbounded faulty players ([FM,Fe] assume private channels).

A long standing open question (cf. [FM,CD]) is whether there exists a *fast* asynchronous BA protocol which is ($\lceil\frac{n}{3}\rceil - 1$)-resilient.

We answer this question in the affirmative. We consider a completely asynchronous network of n players, in which each two players are connected via a private channel. Furthermore, we let the faulty players have unlimited computational power. In this setting, we describe a Byzantine Agreement protocol that is ($\lceil\frac{n}{3}\rceil - 1$)-resilient. With overwhelming probability all the non-faulty players complete the execution of the protocol. Conditioned on the event that all the non-faulty players have completed the execution of the protocol, they do so in constant expected time. The constructions we use in our protocol are of independent interest. We describe these constructions below.

Previous work. Let us describe the chain of results leading to our result, and sketch the techniques used. Rabin [R1] describes an ($\lceil\frac{n}{8}\rceil - 1$)-resilient Byzantine Agreement protocol that runs in constant expected time, provided that all the players have access to a 'global coin' (namely, a common source of randomness with certain properties). Rabin's construction can be used in synchronous as well as asynchronous networks. Bracha [Br] improved the resilience of Rabin's protocol in an *asynchronous* network, given a 'global coin', to ($\lceil\frac{n}{3}\rceil - 1$). The essence of the [FM] synchronous Byzantine Agreement protocol is a scheme for generating such a 'global coin'; once this global coin is generated, the players proceed in a conceptually similar manner to Bracha's protocol. The protocol for generating this 'global coin' relies heavily on a *Verifiable Secret Sharing* (VSS) scheme. (The notion of VSS was introduced by Chor *et al.* [CGMA].) Feldman's asynchronous construction for achieving BA, given an r-resilient *Asynchronous* VSS (AVSS) scheme, is $min(r, \lceil\frac{n}{3}\rceil - 1)$-resilient. However, up to now, no known AVSS scheme has been more than ($\lceil\frac{n}{4}\rceil - 1$)-resilient. (An ($\lceil\frac{n}{4}\rceil - 1$)-resilient AVSS scheme is presented in [Fe,BCG].)

The main difficulty. We offer an intuitive exposition of the difficulties encountered in trying to devise an ($\lceil\frac{n}{3}\rceil - 1$)-resilient AVSS scheme. Generally, in an asynchronous network of n players, out of which t may be faulty, a player can never wait to communicate with more than $n - t$ other players, because the faulty players may not cooperate. An AVSS scheme is composed of two phases: (1) a sharing phase, in which a dealer shares a secret among the players, and each player verifies for himself that a unique secret has been defined by the shares, and (2) a reconstruction phase in which the players reconstruct the secret from its shares. A player P must be able to complete the execution of the sharing phase even if he has communicated with only $n - t$ of the players. This means that he has verified the existence of a well defined secret only with this subset of the players (denoted C_1). When P proceeds to carry out the reconstruction phase he again can wait to communicate with at most $n - t$ of the players. Denote this set by C_2. The set C_2 might include players with whom P has not communicated in the sharing phase and hence he does not know whether their shares are in accordance with the secret defined by the shares of the players in C_1. (Possibly, the players who are not in C_1 did not receive shares at all.) Thus, P can depend only on the players in the intersection, C, of the two sets. Still, t out of the players in C may be faulty. As long as $n \geq 4t + 1$, it holds that $|C| \geq 2t + 1$; thus, the majority of the players in C are honest. However, when $n = 3t + 1$, it is possible that $|C| = t + 1$. In this case, there might be only a single honest player in C.

Our solution. We overcome these difficulties by devising a tool, called Asynchronous Recoverable Sharing (A-RS), assuring that, with overwhelming probability, the shares of *all* the players in the set C_1 (defined above) will be available in the reconstruction phase. The A-RS protocol uses a tool presented in [R2,RB], called Information Checking Protocol. Using A-RS as a primitive, we construct a secret sharing scheme, called Asynchronous Weak Secret Sharing (AWSS). Using AWSS, we construct our AVSS scheme. (Both the AWSS and the AVSS schemes generalize synchronous constructs introduced in [R2,RB].)

Since Feldman's work was never published we also present a modified version of Feldman's construction for reaching BA given an AVSS scheme. Put together, these constructions constitute an asynchronous ($\lceil\frac{n}{3}\rceil - 1$)-resilient BA protocol.

We note that our ($\lceil\frac{n}{3}\rceil - 1$)-resilient AVSS scheme has applications in other contexts as well (for instance, strengthening the results of [BE]).

Organization. This paper is organized as follows: In Section 2 we describe our asynchronous model and give a formal definition of Byzantine Agreement and Asynchronous Verifiable Secret Sharing. In Section 3 we state our main theorems, and present an overview of our protocols. Section 4 contains tools used in our construction. In Sections 5 through 1 we describe our A-RS, AWSS, and AVSS schemes. Sections 8 and 9 contain a short description of our BA protocol given an AVSS scheme.

2 Definitions

2.1 The model

Consider an asynchronous network of n players, where every two players are connected via a reliable and private communication channel. Messages sent on a channel may have arbitrary (however, finite) delays. Furthermore, there is no need that the order of the messages on the channel be preserved.

We regard an execution of an asynchronous protocol as a sequence of *atomic steps.* In each atomic step a single player is active. The player is activated by receiving a message; he then performs an internal computation, and possibly sends messages on its outgoing channels. We consider the order of the atomic steps as controlled by a *scheduler.* We 'enforce' the privacy of the channels by considering *oblivious* schedulers only. The only information known to these schedulers is the origin and destination of each message sent.

In our model, up to t of the players may become faulty at any time during the run of the protocol. We consider a dynamic, Byzantine adversary. This adversary may choose, at the beginning of each atomic step, which additional players to corrupt, without exceeding the limit of t faulty players. Once a player is corrupted, he hands all his internal data over to the adversary. From now on he follows the instructions of the adversary. We colloquially call non-faulty players *honest.*

For convenience, we unite the adversary controlling the faulty players (parameterized by t, the maximum number of faulty players) with the scheduler, to one entity. We call this entity a *t-adversary.* We allow computationally unbounded adversaries.

A time measure. Let us briefly and informally present a standard definition of the running time of an asynchronous protocol. (For a fuller definition of asynchronous time, see, for instance, [AM].) Consider a virtual 'external clock' measuring time in the network (of course, the players cannot read this clock). Let the *delay* of a message be the time elapsed from its sending to its receipt. Let the *period* of a finite execution of a protocol be the longest delay of a message in this execution. The *duration* of a finite execution is the total time measured by the global clock divided by the period of this execution. (Infinite executions have infinite duration.)

The *expected running time* of a protocol, *conditioned on an event,* is the maximum over all inputs and applicable adversaries, of the average over the random inputs of the players, of the duration of executions of the protocol in which this event occurs.

2.2 Asynchronous Byzantine Agreement

Definition 1 *Let π be an asynchronous protocol for which each player has binary input. We say that π is a* $(1-\epsilon)$-terminating, t-resilient Byzantine Agreement *protocol if the following hold, for every t-adversary and every input.*

• **Termination.** *With probability $1-\epsilon$ all the honest players complete the protocol (i.e., terminate locally).*

• **Correctness.** *All the honest players who have terminated have identical outputs. Furthermore, if all the honest players have the same input, denoted σ, then all the honest players output σ.*

2.3 Asynchronous Verifiable Secret Sharing

A secret sharing scheme (either synchronous or asynchronous) consists of two protocols: a sharing protocol, in which a dealer shares a secret among the players, and a reconstruction protocol, in which the players reconstruct the secret from its shares. An honest dealer should be able to share a secret in a reconstructible way. Furthermore, if the dealer is honest then the shared secret should remain unknown to the faulty players, as long as no honest player has started the reconstruction protocol. We say that a secret sharing scheme is *verifiable* if the following holds: if *one* honest player accepts the sharing of a secret, then *all* the honest players accept this sharing, and a unique secret will be reconstructed (even if the dealer is faulty). In the following definition, we formalize the requirements of a verifiable secret sharing scheme for our asynchronous setting. (In the sequel we define some weaker secret sharing schemes.)

Definition 2 *Let* (Sh, Rec) *be a pair of protocols in which a dealer, D, shares a secret s. We say that* (Sh, Rec) *is a $(1-\epsilon)$-correct, t-resilient AVSS scheme for n players if the following hold, for every t-adversary.*

• **Termination.** *With probability $1-\epsilon$ the following requirements hold.*

1. If the dealer is honest, then each honest player will eventually complete protocol Sh.

2. If some honest player has completed protocol Sh, *then each honest player will eventually complete protocol* Sh.

3. If an honest player has completed protocol Sh, *and all the honest players invoke protocol* Rec, *then each honest player will complete protocol* Rec.

• **Correctness.** *Once an honest player has completed protocol* Sh, *then there exists a* unique *value, r, such that with probability $1-\epsilon$ the following holds.*

1. If the dealer is honest, then r is the shared secret, i.e. $r = s$.

2. If all the honest players invoke protocol Rec, *then each honest player outputs r. (Namely, r is the reconstructed secret.)*

• **Secrecy.** *If the dealer is honest and no honest player has begun executing protocol* Rec, *then the faulty players have no information about the shared secret.*

Remark. We stress that an honest player is not required to complete protocol Sh in case that the dealer is faulty. (We do not distinguish between the case where an honest player did not complete protocol Sh, and the case where an honest player has completed Sh unsuccessfully.)

3 Overview of the protocols

First, let us state our main results.

Theorem 1 (AVSS). *Let $n \geq 3t+1$. Then, for every $\epsilon > 0$ there exists a $(1-\epsilon)$-correct, t-resilient AVSS scheme for n players. Conditioned on the event that the honest players terminate, they do so in constant time. Furthermore, the computational resources required of each player are polynomial in n and $\log \frac{1}{\epsilon}$.*

Theorem 2 (BA). *Let $n \geq 3t+1$. Then, for every $\epsilon > 0$ there exists a $(1-\epsilon)$-terminating, t-resilient, asynchronous Byzantine Agreement protocol for n players. Conditioned on the event that the honest players terminate, they do so in constant expected time. Furthermore, the computational resources required of each player are polynomial in n and $\log \frac{1}{\epsilon}$.*

We present an overview of our Byzantine Agreement protocol. Let F be a field of size $\Theta(2^n)$. All the computations in the sequel are done in F. The BA protocol employs the idea of using a 'global coin' to reach agreement.

BA using Global-Coin. This part of our protocol follows Bracha's [B] and Feldman's [Fe] constructions. The protocol proceeds in iterations. In each iteration, each player has a 'modified input' value; in the first iteration, the modified input of each player is his local input. In each iteration the players invoke two protocols, called Vote and Global-Coin. In the Vote protocol, described in Section 9, the players 'vote on an output'; namely, each player tries to establish whether there exists a distinct majority for some value amongst the players' modified inputs of this iteration. If a player recognizes a distinct majority for some value, he takes this value to be his modified input for the next iteration. Otherwise, he sets his modified input for the next iteration to be the output of the Global-Coin protocol (outlined below). We show that no two honest players can recognize a distinct majority for two different values. Furthermore, if all the honest players have the same input value in some iteration, then all the honest players will recognize an 'overwhelming majority' for this value. Once a player recognizes an overwhelming majority for some value, σ, he broadcasts σ (using the A-cast protocol described in Section 4.2), and, subsequently, terminates with output σ.

A Global-Coin protocol has the following specification. Each player has a random input, and outputs a value in $\{0,1\}$. For every value $\sigma \in \{0,1\}$, with probability at least $\frac{1}{4}$ *all* the honest players output σ. Consequently, if the coin is used by a subset (or all) of the honest players in some iteration, then with probability of at least $\frac{1}{4}$ all the honest players will have the same modified input for the next iteration. It follows that the BA protocol terminates within an expected constant number of iterations.

Global-Coin using AVSS. We implement a Global-Coin using techniques introduced in [FM,Fe], employing our AVSS scheme (see Section 8).

AVSS from scratch. Our AVSS scheme is constructed in three 'layers'. Each layer consists of a different secret sharing scheme (with an allowed-error parameter, ϵ). The scheme of the lowest layer is called Asynchronous Recoverable Sharing (A-RS). The scheme of the next layer is called Asynchronous Weak Secret Sharing (AWSS). The top layer is an AVSS scheme (as in Definition 2). Each layer is a building block for the level above it. All three sharing schemes satisfy the *termination* and *secrecy* requirements of Definition 2. In all three schemes, if the dealer is honest then the honest players reconstruct the secret shared by the dealer. The correctness property for the case when the dealer is faulty is upgraded from A-RS to AWSS and finally to AVSS:

A-RS- Once the first honest player completes the Reconstruction Phase, there exists a subset, $S \subseteq F$ of size $2t+1$, such that each honest player will reconstruct a value $r \in S \cup \{null\}$.

AWSS- Once the first honest player completes the Sharing Phase, a value, s, is fixed. Each honest player will reconstruct s or *null*.

AVSS- Once the first honest player completes the Sharing Phase, a value, s, is fixed. Each honest player will reconstruct s.

Given that all the honest players complete all AVSS-reconstruction protocols they invoked, then each iteration of the Byzantine Agreement protocol terminates in constant time.

4 Tools

4.1 Information Checking Protocol- ICP

The Information Checking Protocol (ICP) is a tool for authentication of messages in the presence of computationally unbounded faulty players. We state the properties of this protocol without giving implementation details. The ICP was introduced by T. Rabin and T. Rabin and Ben-Or ([R2],[RB]).

The protocol is executed by three players: a dealer, D, an intermediary, INT, and a receiver, R. The three players have an allowed-error parameter, ϵ. The dealer has a secret value s which he hands over to INT. At a later stage, INT is required to hand this value over to R. The protocol is carried out in three phases:

Generation(s) is initiated by D. In this phase D hands the secret s to INT and some auxiliary data to both INT and R.

Verification is initiated by INT. The receiver, R, is requested to participate. In this phase INT decides whether to continue or abort the protocol. (INT bases his decision on the prediction whether, in the Authentication Phase, R will accept the secret held by INT. We denote continuation (resp. abortion) by Verification= 1 (resp. 0).

Authentication is initiated by R. The intermediary, INT, is requested to participate. In this phase R receives s' from INT, and either accepts or rejects it. We denote acceptance of a secret s', (resp. reject) by Authentication= s' (resp. null).

The ICP has the following properties.

1. If D, holding secret s, INT and R are honest, then Verification= 1 and Authentication= s.
2. If INT and R are honest, and INT has decided,in the Verification Phase, to continue the protocol, then with probability $(1-\epsilon)$, R will accept the secret received from INT in the Authentication Phase.
3. If R and D are honest, and R accepted the secret s in the Authentication Phase, then with probability $(1-\epsilon)$, s originated with D.
4. If D and INT are honest, then as long as INT has not started the Authentication Phase, R has no information about the secret s.

We note that in the implementation that we use, each phase involves sending only two messages, and the computational resources required of each player are polynomial in $\log \frac{1}{\epsilon}$.

4.2 A-cast

The asynchronous broadcast primitive (denoted A-cast) was introduced and elegantly implemented by Bracha [Br]. Bracha's A-cast protocol is $(\lceil \frac{n}{3} \rceil - 1)$-resilient.

Definition 3 *Let π be an n-player protocol initiated by a special player (called the sender), having input m (the message to be broadcast). Protocol π is a t-*resilient A-cast *protocol if the following hold, for every input, and every t-adversary.*

• **Termination.**

1. If the sender is honest and all the honest players participate in the protocol, then each honest player will eventually complete the protocol.

2. If any honest player completes the protocol, then each honest player will eventually complete the protocol.

• **Correctness.** *If the honest players complete the protocol, then they do so with a common output m^*. Furthermore, if the sender is honest then $m^* = m$.*

We stress that the Termination property of A-cast is much weaker than the Termination property of BA: for A-cast, we do not require that the honest players complete the protocol when the sender is faulty.

5 Asynchronous Recoverable Sharing — A-RS

Definition 4 *Let* (Sh, Rec) *be a pair of protocols in which a dealer, D, shares a secret s. We say that* (Sh, Rec) *is a $(1-\epsilon)$-correct, t-resilient A-RS scheme for n players if the following hold, for every t-adversary.*

• **Termination.** *Same as in the definition for AVSS (Definition 2).*

• **Correctness.** *The following holds, with probability $1-\epsilon$.*

1. If the dealer is honest, then each honest player, upon completing protocol Rec, *outputs the shared secret.*

2. Once The first honest player completes protocol Rec then a set $S \subseteq F$ of size $2t+1$ is defined. Each honest player, upon completing protocol Rec, outputs a value $r \in S \cup \{null\}$.

• **Secrecy.** *Same as in the definition for AVSS (Definition 2).*

Our implementation. Sharing protocol. The dealer, having a secret s, chooses values $a_1, \ldots, a_t \in F$ with uniform distribution, and defines the polynomial $f(x) = a_t x^t + \ldots + a_1 x + s$. (We call this process *choosing a random polynomial $f(x)$ for s.* The technique of using a t degree polynomial was introduced by Shamir [S].) Next, for each i, the dealer sends $\beta_i \triangleq f(i)$ to P_i. We say that β_i is P_i's share of s. In addition, for each share β_i and for each player P_j, the dealer executes the ICP-Generation Phase (described in Section 4.1), with player P_i acting as the Intermediary, and player P_j acting as the Receiver. The ICP protocol will have the appropriate allowed-error parameter ϵ' (we set $\epsilon' = \frac{\epsilon}{n^2}$). We denote this execution of the ICP-Generation Phase by $\text{Gen}_{D,i,j}[\epsilon'](\beta_i)$. Next, each two players P_i and P_j execute the ICP-Verification Phase, denoted $\text{Ver}_{i,j}[\epsilon']$, with respect to β_i. Once P_i successfully (i.e. with output 1) terminates $2t+1$ Verification Phases, he A-casts("OK",P_i). A player completes the sharing protocol upon completing $2t+1$ A-casts ("OK",...). Our protocol, denoted A-RS Share is presented in Figure 1.

Let $r \geq t+1$. In the sequel we say that a sequence of shares, $\beta_1, \ldots, \beta_r$, *define a secret s*, if there exists a unique polynomial $f(x)$ of degree at most t, whose constant term is s, such that $f(i) = \beta_i$, for $1 \leq i \leq r$. (We shall interchangeably say that "the shares define the polynomial $f(x)$".)

Reconstruction protocol. First, each player P_i A-casts his share, β_i. He then executes the ICP-Authentication Phase with each other player, P_j. (We denote this execution by $\text{Auth}_{i,j}[\epsilon']$.) For each player P_j, whose share is accepted (namely, $\text{Auth}_{j,i}[\epsilon'] = beta'_j$), player P_i A-casts ("Authenticates",P_i,β'_j).

Player P considers a share *legal*, if it is authenticated by at least $t+1$ players. Once P has $t+1$ legal shares, he computes the secret, If there exists a value which is suggested by $t+1$ players, then P output this value. Otherwise, P outputs *null*. Our protocol denoted A-RS-Recon is presented in Figure 2.

Theorem 3 *Let $n \geq 3t+1$. Then for every $\epsilon > 0$, the pair* (A-RS-Share[ϵ],A-RS-Recon[ϵ]) *is a $(1-\epsilon)$-correct,* *t-resilient A-RS scheme for n players.*

Protocol A-RS-Share[ϵ]

Code for the dealer, on parameter ϵ and secret s:

1. Choose a random polynomial, $f(x)$, for s.
2. Send $\beta_i \overset{\Delta}{=} f(i)$ to each player P_i.
   Let $\epsilon' \overset{\Delta}{=} \frac{\epsilon}{n^2}$. For each two players P_i and P_j execute $\text{Gen}_{D,i,j}[\epsilon'](\beta_i)$.

Code for player P_i, with parameter ϵ:

3. For all $j \in [n]$: Wait until $\text{Gen}_{D,i,j}[\epsilon']$ is completed; then, initiate $\text{Ver}_{i,j}[\epsilon']$.
4. If $Ver_{i,j}[\epsilon'] = 1$ for $2t+1$ players P_j, then A-cast `(OK,` P_i`)` .
5. Upon completing $2t+1$ different A-casts of the form `(OK, ...)` , terminate.

Figure 1: The A-RS sharing protocol

Protocol A-RS-Recon[ϵ]

Code for player P_i, with parameter ϵ:

1. A-cast the share β_i.
2. For each player P_j such that the β_j A-cast is completed, initiate $\text{Auth}_{j,i}[\epsilon']$.
   If $\text{Auth}_{j,i}[\epsilon'] = \beta_j$, then A-cast `(`$P_i$ **`authenticates`** P_j, β_j`)` .
3. Consider a share, β_j, *legal*, if at least $t+1$ A-casts of the form `(`P_k **`authenticates`** P_j, β_j`)` have been completed. Create an *Interpolation Set, IS_i*. Add every legal share to IS_i.
4. Once $|IS_i| = t+1$, compute the secret, s, defined by the shares in IS_i. A-cast **`(suggested secret,`** s`)` .
5. Wait until completing $2t+1$ **`(suggested secret,`** `...)` A-casts.
   If there is a value, s', which appears in at least $t+1$ such A-casts, then output s'.
   Otherwise, output *null*.

Figure 2: The A-RS reconstruction protocol

Proof: Fix a t-adversary. **Termination (1).** When the dealer is honest, the verification protocol of ICP between every two honest players will be successful (ICP property 2, see Section 4.1). Therefore, each honest player will have $2t+1$ successful verifications for its share. Thus, each honest player P will A-cast `(OK,` P`)` . Consequently, each honest player will complete $2t+1$ `(OK, ...)` A-casts, and terminate.

Termination (2). If an honest player has completed A-RS-Share, then he has completed $2t+1$ `(OK, ...)` A-casts. By the definition of broadcast (Definition 3), each honest player will complete these $2t+1$ `(OK, ...)` A-casts, and will thus complete A-RS-Share.

Termination (3). Let E be the event that 'no errors occur' in all the invocations of ICP. Namely, for each two players P_i and P_j, if both P_i ad P_j are honest and $\text{Ver}_{i,j}[\epsilon'] = 1$ then P_i accepts β_j in Step 2 of protocol A-RS-Share; furthermore, if either the dealer or P_j are honest, then the values accepted by P_i is the value that P_j received from the dealer. We have that event E occurs with probability at least $1 - n^2\epsilon' = 1 - \epsilon$.

By the properties of ICP (see Section 4.1), if P_i and P_j are honest and $\text{Ver}_{i,j}[\epsilon'] = 1$ in Step 3 of A-RS-Share, then, conditioned on the event E, the player P_j will accept P_i's share in Step 2 of A-RS-Recon. If an honest player P_i A-casted `(OK,` P_i`)` in Step 3 of A-RS-Share, then for at least $t+1$ *honest* players, P_j, we have $\text{Ver}_{i,j}[\epsilon'] = 1$. Thus, conditioned on event E, these $t+1$ players will accept P_i's share, and will broadcast `(`P_j **`authenticates`** P_i, β_i`)` in Step 2 of A-RS-Recon. Consequently, P_i's share will be considered legal by every honest player in Step 3 of A-RS-Recon.

Now, if an honest player, P, has completed the A-RS-Share protocol, then $2t+1$ players have A-casted `(OK, ...)` in Step 3 of A-RS-Share. Out of these A-casts, at least $t+1$ have originated with honest players. Consequently, if event E occurs, then every honest player will have $t+1$ legal shares in Step 3 of A-RS-Recon. In this case, P will have $2t+1$ suggested secrets, and will complete protocol A-RS-Recon.

Correctness (1). If the dealer is honest and event E occurs, then for each honest player, P_i, each share β_j in IS_i originated with D (namely, $\beta_j = f(j)$). Therefore, the shares in IS_i define the secret, s, shared by the dealer, and P_i will A-cast **`(suggested secret,`** s`)` . Consequently, each honest player will complete $2t+1$ A-casts, out of which at least $t+1$ suggest the secret s. Hence, each honest player will output s.

Correctness (2). If a player, P, has completed protocol A-RS-Recon, then he has completed $2t+1$ A-casts of the form **`(suggested secret,`** `...)` . Let S denote the set of $2t+1$ Suggested secrets in these $2t+1$ A-casts. Clearly, at least $t+1$ out of the $2t+1$ A-casts completed by each other honest player, P', in Step 4 of A-RS-Recon, are of values in S. Thus, if some secret, r, appears $t+1$ times in the A-casts completed by P', then it must be that $r \in S$.

Secrecy. If the dealer is honest and no honest player has initiated protocol A-RS-Recon, then any set of $b \leq t$ players have only b shares of the secret, along with the data received during the ICP generation and verification protocols with respect to the other players' shares of the secret. By the Secrecy properties of ICP and Shamir's secret sharing scheme, this data is independent of the shared secret. □

6 Asynchronous Weak Secret Sharing — AWSS

Definition 5 *Let* (Sh, Rec) *be a pair of protocols in which a dealer, D, shares a secret s. We say that* (Sh, Rec) *is a $(1-\epsilon)$-correct, t-resilient AWSS scheme for n players if the following hold, for every t-adversary.*
• **Termination.** *Same as the definition of AVSS (Definition 2).*
• **Correctness.**
1. If the dealer is honest, then with probability $1-\epsilon$ each honest player, upon completing protocol Rec, *outputs the shared secret.*
2. Once the first honest player completes protocol Sh, *then there exists a value s, such that with probability $1-\epsilon$ each honest player, upon completing protocol* Rec, *will output s or null.*
• **Secrecy.** *Same as the definition of AVSS (Definition 2).*

Our implementation. Sharing protocol. The dealer, having a secret s, chooses a random polynomial $f(x)$ for s. Next, for each i, the dealer sends $\beta_i \triangleq f(i)$ to P_i. In addition, for each share β_i and for each player P_j, the dealer executes the Generation Phase of ICP with player P_i acting as the Intermediary, and player P_j acting as the Receiver. (This part is the same as in A-RS-Share.)

Next, each player shares each value received from the dealer (including the values received in the Generation Phase of ICP), using A-RS-Share (Figure 1). Player P_i creates a set, COM_i. Once P_i completes all the A-RS-Shares of P_j's values, P_j is added to COM_i. (Players may be added to COM_i as long as the AWSS sharing protocol is being executed. We call such sets *dynamic* sets.)

For each player $P_j \in COM_i$, player P_i initiates the Verification Phase of ICP for β_i with P_j, and with the appropriate allowed-error parameter, ϵ'. (We denote this execution by $\mathrm{Ver}_{i,j}[\epsilon']$).

Let ACC_i be P_i's set of players P_j such that $\mathrm{Ver}_{i,j}[\epsilon'] = 1$. Once $|ACC_i| = 2t+1$, player P_i A-casts ACC_i. (This *acceptance set, ACC_i*, is the set of players who will later accept β_i.)

Let ELG_i be P_i's dynamic set of players P_j whose ACC_j A-cast has been completed, and $ACC_j \subseteq COM_i$. We define the set IS_i as the first $2t+1$ players in ELG_i. Once $|ELG_i| \geq 2t+1$, player P_i A-casts IS_i. (This *eligibility set, ELG_i*, is the set of players whose share will be later considered either legal or *null*. The *interpolation set, IS_i*, is the set of players whose shares will later define the secret associated with P_i.)

Let $FINAL_i$ be P_i's set of players P_j whose IS_j A-cast has been completed, and $IS_j \subseteq ELG_i$. Once $|FINAL_i| = 2t+1$ complete the sharing protocol. (This *final set, $FINAL_i$*, is the set of players, with whom P_i will later associate a secret.)

Reconstruction protocol. Remark: it will be seen that when player P executes the reconstruction protocol, he essentially doubles up as each other player (namely, he locally executes the protocols of the other players).

Initially, each player P_i reconstructs the values shared by each player $P_j \in ELG_i$, in the following manner. Recall that each player in ELG_i shared, in the sharing protocol, values he received as an Intermediary of ICP, and values he received as a Receiver of ICP. Player P_i first invokes A-RS-Recon for all the values that the players in ELG_i received as Intermediaries in ICP. Call these invocations of A-RS-Recon *preliminary.* For each player $P_j \in ELG_i$ and for each $P_l \in ACC_j$, once all the preliminary A-RS-Recons have been completed, P_i invokes the A-RS-Recons of (the rest of) the values shared by P_l. *We note that the order in which the A-RS-Recons are invoked is crucial for the proof of the Correctness property of the scheme.*

For each player $P_j \in ELG_i$ and for each player $P_l \in ACC_j$, once all the necessary values are reconstructed, simulate $\mathrm{Auth}_{j,l}[\epsilon']$, in order to compute P_l's result of $\mathrm{Auth}_{j,l}[\epsilon']$. Player P_i associates a value with player P_j in the following manner: If P_i finds out that $t+1$ players in ACC_j have accepted β_j then he associates β_j with P_j. (We then say that β_j is *legal*). Otherwise, he associates the value "null" with P_j.

Now, for each player $P_k \in FINAL_i$, once the values associated with all the players in IS_k are computed, if all the legal shares in IS_k define a secret, s, then P_i considers s to be the secret *suggested* by P_k. Otherwise, the secret suggested by P_k is *null.*

Finally, P_i waits until the secrets suggested by all the players in $FINAL_i$ are computed. If all the players in $FINAL_i$ suggest the same secret, s, P_i outputs s. Otherwise, P_i outputs *null.*

Theorem 4 *Let $n \geq 3t+1$. Then for every $\epsilon > 0$, the pair* (AWSS-Share[ϵ],AWSS-Recon[ϵ]) *is a $(1-\epsilon)$-correct, t-resilient AWSS scheme for n players.*

Proof: See our Technical Report [CR].

6.1 Two&Sum Asynchronous WSS

In the AVSS scheme we will need the following variation of an AWSS scheme: A dealer shares three secrets: s_1, s_2, s_3, such that $s_3 = s_1 + s_2$. The sharing of these secrets should have the properties of the AWSS scheme; In addition, the following requirements should hold:
1. If we reconstruct either s_2 or $s_1 + s_2$, then the faulty players have no information about s_1.
2. Let r_i be the value set once the first honest player completes the sharing of s_i. Then, either $r_1 + r_2 = r_3$, or at least two of $\{r_1, r_2, r_3\}$ are null.

An implementation of this scheme, denoted Two&Sum-AWSS, is a straight forward generalization of our AWSS scheme. The scheme is based on the *synchronous* Two&Sum-WSS presented in [R2]. We present our scheme in [CR].

7 Asynchronous Verifiable Secret Sharing — AVSS

Our implementation. In this extended abstract we only sketch our construction. For full details see our Technical Report [CR].
Sharing protocol. The sharing protocol consists of two phases: in the first phase, the dealer sends shares to all the players. In the second phase, the players verify that their shares define a unique secret.

The dealer, having a secret s, chooses a random polynomial $f(x)$ for s. Next, for each i, the dealer sends $\beta_i \overset{\Delta}{=} f(i)$ to P_i. (We say that β_i is P_i's share of the secret.) In addition, the dealer chooses an appropriate number of random polynomials $g(x)$ of degree t. For each such polynomial $g(x)$, and for each i, the dealer sends $g(i)$ to player P_i. Upon receiving all the expected values from the dealer, each player shares the values $\{f(i), g(i), f(i)+g(i)\}$ for each polynomial $g(x)$, using the Two&Sum AWSS sharing protocol with the appropriate allowed-error parameter (see Section 6.1).

Next, the dealer proves to each player, using a cut-and-choose method (as in [BGW,CCD,Fe,FM]), that the shares of the players indeed define a secret. Once a player is convinced that the shares of $2t+1$ players are acceptable (namely, the Two&Sum-AWSS-Share of these shares are completed, and the values set by the Two&Sum-AWSS-Shares of these shares, define a secret), he *confirms* these players by A-casting their identities. Once a a player, P_i, completes $2t + 1$ A-casts in which the player P is confirmed, he adds P to his *final set*, $FINAL_i$. Once $FINAL_i$ contains $2t + 1$ players, player P_i completes the sharing protocol.

For each player $P_j \in FINAL_i$ let r_j be the values set by the AWSS-Share of P_j. Note that the values r_j of the players $P_j \in FINAL_i$ define a secret; furthermore, the values in all the $FINAL_i$'s define the same secret. This holds because of the following reason. For each player $P_j \in FINAL_i$ there exists at least one honest player, who has confirmed him. Thus, there exists a t-degree polynomial, $h(x)$, such that r_j, together with at least $2t$ other values, define the polynomial $h(x)$. We say that $h(x)$ is *associated* with P_j. It can be easily seen that all the polynomials associated with all the players $P_j \in FINAL_i$ are identical.

Reconstruction protocol. The reconstruction protocol is simple. The shares of all the players are reconstructed using AWSS-Recon. Once $t+1$ shares in $FINAL_i$ are reconstructed, player P_i computes the secret defined by these shares, and terminates. (Recall that the shares of all the players in $FINAL_i$ define a secret. Thus, this secret can be computed based on any $t+1$ of these shares.)

Theorem 1 *Let $n \geq 3t + 1$. Then for every $\epsilon > 0$, the pair* (AVSS-Share[ϵ],AVSS-Recon[ϵ]) *is a $(1 - \epsilon)$-correct, t-resilient AVSS scheme for n players.*

Proof: See our Technical Report [CR]. We note that showing that the sharing protocol terminates in constant time (given that all the honest player complete the protocol) requires subtle arguments.

8 Global Coin from AVSS

Definition 6 *Let π be a protocol, where each player has a random input, and outputs a value in $\{0, 1\}$. We say that π is a* $(1 - \epsilon)$-terminating, t-resilient global coin *protocol if the following hold, for every t-adversary.*
• **Termination.** *With probability $1 - \epsilon$, all the honest players terminate locally.*
• **Correctness.** *For every value $\sigma \in \{0, 1\}$, with probability at least $\frac{1}{4}$ all the honest players output σ.*

An implementation. Our construction follows the outline of Feldman's construction. However, several modifications have been introduced. In this extended abstract we only sketch this construction. For full details see our Technical Report [CR]. Roughly speaking, the protocol, denoted Global-Coin[ϵ] (with 'termination parameter' ϵ), consists of four stages. First, each player shares n random secrets, using the AVSS-Share protocol of an AVSS scheme. (These AVSS-Shares are invoked with the appropriate 'allowed error' parameter, which is a function of ϵ.) We say that the ith secret shared by each party is assigned to player P_i. Once a player completes $n - t$ AVSS-Share protocols of secrets assigned to him, the player A-casts the identity of the dealers of these secrets. (Later, the value associated with this player will be computed based on these secrets.)

Once a player is assured that enough players have A-casted their set of assigned secrets, he starts reconstructing the relevant secrets. Once all these secrets are reconstructed, each player locally computes his output based on the reconstructed secrets.

For the proof of correctness, we show that once some honest player completes the A-cast of the secrets assigned with player P, a random value, v_P, is fixed. With overwhelming probability, the value that each honest player will later associate with P will be v_P. Now, each honest player P_i computes his output based on the values associated with the parties in some set, S_i, of size $n-t$. We show that once the first honest player starts reconstructing secrets, a large enough "core" set, C, of players is fixed. Each honest player will have $C \subset S_i$. We use these facts to show that for each value $\sigma \in \{0,1\}$, with large enough probability all the players output σ.

Theorem 5 *Let $n \geq 3t+1$. Then, for every $\epsilon > 0$ protocol* Global-Coin$[\epsilon]$ *is a $(1-\epsilon)$-terminating, t-resilient global coin protocol for n players.*

Proof: See our Technical Report [CR]. □

9 Byzantine Agreement from Global Coin

Before describing the Byzantine Agreement protocol, let us describe another protocol used in our construction. This protocol, denoted Vote, does 'whatever can be done deterministically to reach agreement'.

9.1 The voting protocol

Each player's input of the Vote protocol is a binary value. Each player tries to find out whether there exists a majority for some value among the inputs of the players. More precisely, each player's output of the Vote protocol has two fields: the first field is the 'majority vote' (i.e., a value in $\{0,1\}$). The second field is the 'majority parameter'. A majority parameter 2 stands for 'overwhelming majority'; majority parameter 1 stands for 'distinct majority'; majority parameter 0 stands for 'no distinct majority'. It will be shown that the difference in the majority parameter of every two honest players is at most 1, and no two honest players recognize a distinct majority for different values. Furthermore, if all the honest players have the same input, σ, then all the honest players will recognize an overwhelming majority for σ.

The protocol consists of three 'rounds'. In the first round, each player A-casts his input value, waits to receive $n-t$ other inputs, and sets his vote to the majority value among these inputs. In the second round, each player A-casts his vote (along with the identities of the $n-t$ players whose A-casted inputs were used to compute the vote), waits to receive $n-t$ other votes that are consistent with the A-casted inputs of the first round, and sets his re-vote to the majority value among these votes. In the third round each player repeats the same procedure with respect to the A-casted re-votes. (Namely, each player A-casts his re-vote, along with the identities of the $n-t$ players whose A-casted votes were used to compute the re-vote, and waits to receive $n-t$ other re-votes that are consistent with the consistent votes of the second round.)

Now, if all the consistent votes received by a player agree on a value, σ, then this player outputs $(\sigma, 2)$. Otherwise, if all the consistent re-votes received by the player agree on a value, σ, then the player outputs $(\sigma, 1)$. Otherwise, the player outputs $(0,0)$.

9.2 The Byzantine Agreement protocol

Protocol BA$[\epsilon](x_i)$

Code for player P_i, on input x_i, and parameter ϵ:

```
1. Set $r \leftarrow 1$. Set $v_1 \leftarrow x_i$.

Repeat until terminating:

2. Set $(y_r, m_r)$ =Vote($v_r$).
   (Namely, invoke protocol Vote with input $v_r$;
   once the protocol is completed, let $(y_r, m_r)$
   be the outputs.)

3. Wait until $(y_r, m_r)$ are computed. Then, set
   $c_r$ =Global-Coin[$\frac{\epsilon}{4}$].
   (a) If $m_r = 2$, set $v_{r+1} \leftarrow y_r$ and A-cast
       (Terminate with $y_r$) .
       Participate in only one more Vote protocol
       and only one more Global-Coin
       protocol.
   (b) If $m_r = 1$, set $v_{r+1} \leftarrow y_r$.
   (c) Otherwise, set $v_{r+1} \leftarrow c_r$.

• Upon receiving $t+1$ (Terminate with $\sigma$)
  A-casts for some value $\sigma$, output $\sigma$ and terminate.
```

Figure 3: The Byzantine Agreement protocol

The Byzantine Agreement protocol proceeds in iterations. In each iteration, each player has a 'modified input' value; in the first iteration, the modified input of each player is his local input. In each iteration the players invoke two protocols: Vote and Global-Coin. Protocol Global-Coin is invoked only *after* protocol Vote is completed. If a player recognizes a 'distinct majority' for some value, σ, in the output of protocol Vote (namely output $(\sigma, 1)$ or $(\sigma, 2)$), then he sets his modified input for the next iteration to σ. Otherwise, he sets his modified input for the next iteration to be the output of the Global-Coin protocol.

Once a player recognizes an 'overwhelming majority' for some value, σ, (namely, output $(\sigma, 2)$ of protocol Vote), he A-casts σ. Once a player completes $t+1$ A-casts for some value, σ, he terminates with output σ.

The code of the Byzantine Agreement protocol is presented in Figure 3.

Theorem 2 *Let $n \geq 3t+1$. Then, for every $\epsilon > 0$ protocol* BA[ϵ] *is a $(1-\epsilon)$-terminating, t-resilient, asynchronous Byzantine Agreement protocol for n players. Given that the players terminate, they do so in constant expected time. Furthermore, the computational resources required of each player are polynomial in n and* $\log \frac{1}{\epsilon}$.

The proof of Theorem 2 is presented in [CR]. The following claims are the milestones of this proof.

Claim 1 *Let $k \geq 0$, and assume that all the honest players complete the kth invocation of Global-Coin. Then the execution of the kth iteration of protocol BA terminates in constant time.*

Claim 2 *For each $k \geq 0$, the probability that all the players complete the kth iteration (namely, compute v_{k+1}), given that all the players complete all the previous iterations, is at least $1 - \frac{\epsilon}{4}$.*

Claim 3 *For each $k \geq 0$, the probability that all the players have the same v_{k+1} value, given that all he players complete the kth iteration (namely, compute v_{k+1}), is at least $\frac{1}{4}$.*

Acknowledgements

A special thank is due to Michael Rabin for his patience and wise advice. In addition we thank Oded Goldriech and Michael Ben-Or for commenting on our drafts. Finally, we warmly thank Boaz Kelmer.

References

[AM] H. Attiya and M. Mavronicolas, "Efficiency of Semi-Synchronous versus Asynchronous Networks," *Mathematical Systems Theory,* to appear. Preliminary version in proceedings of *the 28th annual Allerton Conference on Communication, Control and Computing,* October 1990, pp. 578–587.

[BCG] M. Ben-Or, R. Canetti and O. Goldreich, "Asynchronous Secure Computations", submitted to these proceedings.

[BE] M. Ben-Or and R. El-Yaniv, "Interactive Consistency in Constant Time", submitted for publication, 1991

[BGW] M. Ben-Or, S. Goldwasser and A. Wigderson, " Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", *20th STOC,* pp. 1-10, 1988.

[Br] G. Bracha, "An Asynchronous $\lfloor (n-1)/3 \rfloor$-resilient Consensus Protocol", *3rd PODC,* pp. 154-162, 1984.

[CR] R. Canetti, and T. Rabin, "Optimal Asynchronous Byzantine Agreement", *Technical Report #92-15, CS Department, Hebrew University,* 1992.

[CCD] D. Chaum, C. Crepeau and I. Damgard, "Multiparty Unconditionally Secure Protocols", *20th STOC,* pp. 11-19, 1988.

[CGMA] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults", *26th FOCS,* pp. 383-395, 1985.

[CD] B. Chor and C. Dwork, "Randomization in Bysantine Agreement", *Advances in Computing Research, Vol. 5,* pp. 443-497, 1989.

[Fe] P. Feldman, "Asynchronous Byzantine Agreement in Constant Expected Time", unpublished manuscript, 1989.

[FM] P. Feldman, and S. Micali, "An Optimal Algorithm For Synchronous Byzantine Agreement", *20th STOC,* pp. 148-161, 1988.

[F] M. Fischer, "The Concensus Problem in Unreliable Distributed System", *Technical Report, Yale University,* 1983.

[FLP] M. Fischer, N. Lynch and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process", *JACM, Vol. 32, no. 2,* pp. 374-382, 1985.

[KY] A. Karlin and A. Yao, "Probabilistic Lower Bounds for Byzantine Agreement", unpublished manuscript, 1986.

[PSL] M. Pease, R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults", *JACM, Vol. 27 No. 2,* pp. 228-234, 1980.

[R1] M. Rabin, "Randomized Byzantine Generals", *24th FOCS,* pp. 403-409, 1983.

[R2] T. Rabin , "Robust Sharing Of Secrets When The Dealer Is Honest Or Faulty", *Masters Thesis, Hebrew University, submitted for publication to the JACM.*

[RB] T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority", *21st STOC,* pp. 73-85, 1989.

[Sh] A. Shamir, "How to share a secret", *CACM, Vol. 22, No. 11,* pp. 612-613, 1979.