CHAPTER 5

Asynchronous Byzantine agreement

We describe the first $(\lceil \frac{n}{3} \rceil - 1)$-resilient asynchronous Byzantine agreement protocol with polynomial complexity. We use ideas and techniques that emerge from secure multiparty computation. See Section 1.5 for an introductory presentation and discussion.

In Section 5.1 we recall the asynchronous model and definitions of Byzantine Agreement and Asynchronous Verifiable Secret Sharing (AVSS). In Section 5.2 we state our main theorems, and present an overview of our protocols. In Section 5.3 we describe tools used in our construction. In Sections 5.4 through 5.6 we describe our $(\lceil \frac{n}{3} \rceil - 1)$-resilient AVSS scheme. In Sections 5.7 and 5.8 we describe our BA protocol given an AVSS scheme.

5.1 Definitions

We assume the same model as in Chapter 4 (see Section 4.1 on page 49). We also use the same conventions for writing asynchronous protocols (see Section 4.1.3), and the same measure of running times of asynchronous protocols (see Section 4-A). We re-define Byzantine agreeement and AVSS. The definitions here are identical to the definitions in Chapter 4 (Definitions 4.5 and 4.18, respectively), with the exception that here we allow the parties to not terminate (and in AVSS also to have wrong output) with small probability.

Definition 5.1 *Let π be an asynchronous protocol for which each party has binary input. We say that π is a $(1-\epsilon)$*-terminating, t-resilient Byzantine Agreement *protocol if the following requirements hold, for every t-adversary and every input.*

- **Termination.** *With probability $1-\epsilon$ all the uncorrupted parties complete the protocol (i.e., terminate locally).*
- **Correctness.** *All the uncorrupted parties who have terminated have identical outputs. Furthermore, if all the uncorrupted parties have the same input, σ, then all the uncorrupted parties output σ.*

Definition 5.2 *Let S be a finite set. Let (S, R) be a pair of protocols in which a dealer, D, shares a secret $s \in S$. All parties invoke protocol* S*, and later invoke protocol* R *with the* *local output of protocol* S *as local input. We say that* (S, R) *is a* $(1-\epsilon)$*-correct, t-resilient AVSS scheme for n parties if the following hold, for every t-adversary.*

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

1. If the dealer is uncorrupted, then each uncorrupted party will eventually complete protocol S*.*

2. If some uncorrupted party has completed protocol S*, then each uncorrupted party will eventually complete protocol* S*.*

3. If all the uncorrupted parties have completed protocol S*, then all the uncorrupted parties will complete protocol* R*.*

• **Correctness.** *Once the first uncorrupted party has completed protocol* S*, then a value, r, is fixed, such that the following requirements hold with probability* $1-\epsilon$*:*

1. Each uncorrupted party outputs r. (Namely, r is the reconstructed secret.)

2. If the dealer is uncorrupted, then r is the shared secret, i.e. $r = s$*.*

• **Secrecy.** *If the dealer is uncorrupted and no uncorrupted party has begun executing protocol* R*, then the information gathered by the adversary during the computation is independent of the shared secret.*

Remark: We stress that an uncorrupted party is not required to complete protocol S in case that the dealer is corrupted. We do not distinguish between the case where an uncorrupted party did not complete protocol S, and the case where an uncorrupted party has completed S unsuccessfully.

5.2 Overview of the protocols

First, let us state our main results.

Theorem 5.3 (AVSS). *Let* $n \geq 3t+1$*. For every* $\epsilon > 0$ *there exists a* $(1-\epsilon)$*-correct, t-resilient AVSS scheme for n parties. Conditioned on the event that the honest parties terminate, they do so in constant time. Furthermore, the computational resources required of each party are polynomial in n and* $\log \frac{1}{\epsilon}$*.*

Theorem 5.4 (BA). *Let* $n \geq 3t+1$*. For every* $\epsilon > 0$ *there exists a* $(1-\epsilon)$*-terminating, t-resilient, asynchronous Byzantine Agreement protocol for n parties. Conditioned on the event that the honest parties terminate, they do so in constant expected time. Furthermore, the computational resources required of each party are polynomial in n and* $\log \frac{1}{\epsilon}$*.*

Our Byzantine Agreement protocol is complex and involves many layers. To facilitate the reading we first present an overview of our protocol. Let $\mathcal{F}$ be a field of size greater than n. All the computations in the sequel are done in $\mathcal{F}$. The BA protocol employs the idea of using 'common coins' to reach agreement, as follows.

BA using Common Coin. This part of our protocol follows the constructions of Rabin, Bracha and Feldman [MRa2, Br, Fe]. The protocol proceeds in rounds. In each round, each party has a 'modified input' value. In the first round, the modified input of each party is his local input. In each round the parties invoke two protocols, called Vote and Common Coin. Protocol Common Coin has the following property. Each party has a random input, and binary output. For every value $\sigma \in \{0,1\}$, with probability at least $\frac{1}{4}$ *all* the honest parties output σ. In protocol Vote each party tries to establish whether there exists a 'distinct majority' for some value amongst the parties' modified inputs of this round (we define distinct majority in the sequel). If a party recognizes a 'distinct majority' for some value he takes this value to be his modified input for the next round. Otherwise, he sets his modified input for the next round to be the output of protocol Common Coin. We show that no two honest parties ever recognize a distinct majority for two different values. This is used to show that in each round, with probability at least $\frac{1}{4}$ all parties have the same modified input.

If all the honest parties have the same modified input in some round, then all the honest parties will recognize this and terminate outputting this value. It follows that the BA protocol terminates within a constant expected number of rounds. Given that all the honest parties complete all Common Coin protocols they invoked, then each round of the Byzantine Agreement protocol terminates in constant time.

Common Coin using AVSS. Our construction follows Feldman, and Feldman and Micali [Fe, FM]. The protocol proceeds roughly as follows. First, each party shares n random secrets using our AVSS scheme. Once a party is assured that enough secrets have been properly shared, he starts reconstructing the relevant secrets. Once all these secrets are reconstructed, each party locally computes his output based on the reconstructed secrets.

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 next scheme is called Asynchronous Weak Secret Sharing (AWSS). The last ('top') layer is an AVSS scheme (as in Definition 5.2). Each scheme is used as a building block for the next. All three sharing schemes satisfy the *termination* and *secrecy* requirements of Definition 5.2. In all three schemes, if the dealer is honest then the honest parties always reconstruct the secret shared by the dealer. The correctness property for the case that the dealer is faulty is upgraded from A-RS to AWSS and finally to AVSS:

A-RS- Once the first honest party completes the reconstruction phase, a subset $S \subseteq \mathcal{F}$, of size $2t+1$, is fixed. With probability $1-\epsilon$, each honest party will reconstruct a value $r \in S \cup \{null\}$. (We stress that it is not required that all honest parties end up with the same reconstructed value.)

AWSS- Once the first honest party completes the sharing phase, a value s is fixed. With probability $1-\epsilon$, each honest party will reconstruct either s or *null*.

AVSS- Once the first honest party completes the sharing phase, a value s is fixed. With probability $1-\epsilon$, each honest party will reconstruct s.

5.3 Tools

5.3.1 Information Checking Protocol- ICP

The Information Checking Protocol (ICP) is a tool for authenticating messages in the presence of (computationally unbounded) faulty parties. The ICP was introduced by T.

ICP

Generation phase (Gen):

1. The dealer D, having a secret $s \in \mathcal{F}$, chooses random values $y_1, ..., y_{2k}$ and $b_1, ..., b_{2k}$ uniformly distributed in F. He computes $c_i \triangleq b_i s + y_i$ for $1 \leq i \leq 2k$.
2. D sends s and $\vec{y}_s \triangleq y_1, ..., y_{2k}$ to I.
3. D sends the *check vectors* $(b_1, c_1), ..., (b_{2k}, c_{2k})$ to R.

Verification (Ver):

1. I chooses k distinct random indices $d_1, ..., d_k$, $1 \leq d_i \leq 2k$, and asks R to reveal $(b_{d_1}, c_{d_1}), ..., (b_{d_k}, c_{d_k})$.
2. R reveals these values. The remaining indices will be the *unrevealed indices.*
3. For each one of the revealed indices d_i, I tests whether $s \cdot b_{d_i} + y_{d_i} = c_{d_i}$. If *all* k indices satisfy this requirement then I sets **Ver**$=1$. Otherwise, he sets **Ver**$=0$.

Authentication (Auth):

1. I sends s and the rest of the y's to R.
2. If $k/2$ out of the unrevealed indices d_i satisfy $s \cdot b_{d_i} + y_{d_i} = c_{d_i}$ then R sets **Auth**$= s$, otherwise **Auth**$=$*null*.

Figure 5-1: The Information Checking Protocol of [TRa, RB]

Rabin and Ben-Or [TRa, RB] We first state the properties of this protocol. Next we sketch the [TRa, RB] construction.

The protocol is executed by three parties: a dealer D, an intermediary I, and a receiver R. The dealer hands a secret value s over to I. At a later stage, I is required to hand this value over to R, and to convince R that s is indeed the value which I received from D. More precisely, the protocol is carried out in three phases:

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

VERIFICATION is carried out by I and R. In this phase I decides whether to continue or abort the protocol. I bases his decision on the prediction whether, in the Authentication phase, R will output s, the secret held by I. We denote continuation (resp., abortion) by **Ver**$=1$ (resp., 0).

AUTHENTICATION is carried out by I and R. In this phase R receives a value s' from I, along with some auxiliary data, and either accepts or rejects it. We denote acceptance of a secret s', (resp., rejection) by **Auth**$= s'$ (resp., null).

The ICP has the following properties, given an 'allowed-error parameter' ϵ:

Correctness:

1. If D, holding a secret s, I and R are all honest, then **Ver**$=1$ and **Auth**$= s$.
2. If I and R are honest, and I has decided, in the Verification phase, to continue the protocol, with local input s', then with probability $(1-\epsilon)$, R will output s' in the Authentication phase.
3. If D and R are honest, and R accepted a secret s' in the Authentication phase, then

$s' = s$, with probability $(1-\epsilon)$.

Secrecy:

4. If D and I are honest, then as long as I has not joined in the Authentication phase, R has no information about the secret s.

For self containment we sketch the [TRa, RB] construction in Figure 5-1.[1]

5.3.2 Broadcast

We use the same definition of Broeadcast as in Chapter 4 (see Section 4.2.2). We also use the elegant implementation of Bracha [Br], described there.

5.4 Asynchronous Recoverable Sharing — A-RS

Unlike synchronous systems, in an asynchronous system it is not possible to decide whether a party from which messages do not arrive is faulty or just slow. As argued in the Introduction, this difficulty causes known techniques for AVSS to fail when $n \leq 4t$. We overcome this difficulty using the Information Checking Protocol (ICP) described in the previous section. We first show how the ICP is used to construct A-RS.

An A-RS scheme satisfies the *termination* and *secrecy* requirements of AVSS (Definition 5.2). Also, if the dealer is honest then the honest parties always reconstruct the secret shared by the dealer. The difference from AVSS lies in the case where the dealer is faulty. For A-RS we only require that:

Correctness of A-RS for the case that the dealer is faulty: *once the first honest party completes the reconstruction phase, a subset $S \subseteq \mathcal{F}$, of size $n-t$, is defined. With probability $1-\epsilon$, each honest party will reconstruct a value $r \in S \cup \{null\}$. (We stress that it is not required that all honest parties end up with the same reconstructed value.)*

A-RS has a "synchronizing effect" on the network, in the sense that it guarantees a bounded wait for receiving values, *even from faulty parties.* That is, if we have completed the sharing of a value by some party, then we are guaranteed to eventually reconstruct *some* (not necessarily valid) value. We use this property in the construction of AWSS in the next section.

We describe our construction. Basically, we use Shamir's classic secret sharing scheme [Sh] while using ICP to verify the shares. This way, we can make sure that the following two properties hold simultaneously (with overwhelming probability): (a) the shares of at least $t+1$ honest parties will always be available at reconstruction, and (b) if the dealer is honest then all the shares available at reconstruction are the originally dealt shares. A more detailed description of our construction follows.

Remark: Here, as well as in all subsequent secret sharing schemes, we use the convention that the dealer, in addition to executing his specific code, also participates as one of the parties (and will eventually have a share of the secret).

[1] In fact, the version presented here differs from the one in [TRa, RB] in the decision rule for R in the Authentication stage (in the original construction R accepted the secret if there *existed* an index d_i that satisfied the requirement. This modification allows us to tolerate fields of size only slightly larger than the number of parties.

Sharing protocol. The dealer, having a secret s, chooses values $a_1, ..., a_t \in_R \mathcal{F}^t$, and defines the polynomial $f(x) = a_t x^t + ... + a_1 x + s$.[2] (We call this process choosing a random polynomial $f(x)$ for s.) Next, for each i, the dealer sends $\phi_i \triangleq f(i)$ to P_i. We say that ϕ_i is P_i's share of s. In addition, for each share ϕ_i and for each party P_j, the dealer executes the Generation phase of ICP (described in Section 5.3.1), with party P_i acting as the intermediary, and party 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 Generation phase by $\mathbf{Gen}_{D,i,j}[\epsilon'](\phi_i)$. (In the sequel, we omit the parameter ϵ' from the notation.) Next, every two parties P_i and P_j execute the Verification phase of ICP, denoted $\mathbf{Ver}_{i,j}[\epsilon']$. Success of $\mathbf{Ver}_{i,j}[\epsilon']$ assures P_i that, with probability $1-\epsilon'$, P_j will later authenticate his share ϕ_i. Once P_i successfully (i.e. with output 1) terminates $2t+1$ invocations of $\mathbf{Ver}_*[\epsilon']$, he broadcasts `(OK,` P_i`)` . A party completes the sharing protocol upon completing $2t+1$ broadcasts of the form `(OK,` P_k`)` . Our protocol, denoted A-RS Share, is presented in Figure 5-2.

Definition 5.5 *Let $\mathcal{F}$ be a field, and let $r \geq t+1$. A set $\{(i_1, \phi_{i_1}), \ldots, (i_r, \phi_{i_r})\}$ where $\phi_{i_j} \in \mathcal{F}$ is said to* define a secret *s if there exists a (unique) polynomial $f(x)$ of degree at most t, such that $f(0) = s$, and $f(i_j) = \phi_{i_j}$, for $1 \leq j \leq r$. (We shall interchangeably say that the* shares *$\phi_{i_1}, \ldots, \phi_{i_r}$ define the polynomial $f(x)$.)*

Using interpolation, one can efficiently check whether a given set of shares define a secret.

Reconstruction protocol. First, each party P_i initiates an broadcast of his share, ϕ_i. He then executes the Authentication phase of ICP with every other party, P_j. (We denote this execution by $\mathbf{Auth}_{i,j}$.) For each party P_j, whose share is accepted (namely, $\mathbf{Auth}_{j,i}[\epsilon'] = \phi'_j$), party P_i broadcasts `(`P_i `authenticates` P_j`)` .

Party P_i considers a share, ϕ_j, legal, if P_j has been authenticated by at least $t+1$ parties. Once P_i has $t+1$ legal shares, he computes the secret which they define, and broadcasts it. If there exists a value which is broadcasted by at least $n-2t$ parties, then P_i output this value. Otherwise, P_i outputs *null*. (This extra broadcast is required to limit the size of the set of possible values reconstructed by the different parties.)

The reconstruction protocol, denoted A-RS-Rec, is presented in Figure 5-3.

Theorem 5.6 *Let $n \geq 3t+1$. Then for every $\epsilon > 0$, the pair* (A-RS-Share[ϵ],A-RS-Rec[ϵ]) *is a $(1-\epsilon)$-correct, t-resilient A-RS scheme for n parties.*

Proof: Fix a t-adversary. We show that the Termination and Secrecy requirements of Definition 5.2 (AVSS) are met, as well as the Correctness property for the case that the dealer is honest. The Correctness for the case that the dealer is honest was stated above.

Termination (1): When the dealer is honest, the Verification phase of ICP between every two honest parties will terminate with **Ver**=1 (see the definition of ICP on page 104). Therefore, each honest party will have $2t+1$ successful verifications for his share ϕ_i. Thus, each honest party P_i will initiate an broadcast of `(OK,` P_i`)` . Consequently, each honest party will complete participation in $2t+1$ broadcasts of the form `(OK,` P_j`)` , and terminate protocol A-RS-Share.

[2]We let $e \in_R D$ denote an element e chosen uniformly at random from domain D.

Protocol A-RS-Share[ϵ]

Code for the dealer, with secret s: [a]

1. Choose a random polynomial, $f(x)$, for s.
2. Set $\phi_i \triangleq f(i)$ for $1 \leq i \leq n$.
   Let $\epsilon' \triangleq \frac{\epsilon}{n^2}$. Execute $\mathbf{Gen}_{D,i,j}[\epsilon'](\phi_i)$ for $1 \leq i, j \leq n$.

Code for party P_i:

3. For all $j \in [n]$: Wait until $\mathbf{Gen}_{D,i,j}$ is completed; then, initiate $\mathbf{Ver}_{i,j}$.
4. If $\mathbf{Ver}_{i,j} = 1$ for $2t+1$ parties P_j, then initiate broadcast `(OK,` P_i`)` .
5. Upon completing participation in $2t+1$ different broadcasts of the form `(OK,` P_k`)` , terminate.

Figure 5-2: The A-RS sharing protocol

[a] In this and all subsequent sharing protocols, the dealer also executes the code of a regular party.

Protocol A-RS-Rec[ϵ]

Code for party P_i:

1. Broadcast the share ϕ_i.
2. For each party P_j for whom the broadcast of his share has been completed (with ϕ'_j), initiate $\mathbf{Auth}_{j,i}$.
   If $\mathbf{Auth}_{j,i} = \phi'_j$, then initiate the broadcast `(`P_i `authenticates` P_j`)` .
3. Consider a share, ϕ'_j, legal if at least $t+1$ broadcasts of the form `(`P_k `authenticates` P_j`)` have been completed. Create a set called IS_i (standing for Interpolation Set. Add every legal share to IS_i.
4. Once $|IS_i| = t+1$, compute, using interpolation, the secret s defined by the shares in IS_i. Broadcast `(`P_i `suggests secret` s`)` .
5. Wait until completing $n-t$ `(`P_* `suggests secret` s'`)` broadcasts.
   If there is a value, s', which appears in at least $n-2t$ such broadcasts, then output s'. Otherwise, output *null*.

Figure 5-3: The A-RS reconstruction protocol

Termination (2): If one honest party has completed A-RS-Share, then he has completed participation in $2t+1$ `(OK,` P_j`)` broadcasts. By the properties of broadcast (Definition 4.6), every honest party will complete these $2t+1$ broadcasts, and will thus complete A-RS-Share.

Termination (3): Let E be the event that no errors occur in all the invocations of ICP. (That is, all the requirements listed in Section 5.3 hold with no probability of error.) We know that event E occurs with probability at least $1 - n^2\epsilon' = 1 - \epsilon$.

If some honest party, P_i, has completed the A-RS-Share protocol, then $2t+1$ parties have broadcasted `(OK, ...)` in Step 3 of A-RS-Share. Out of these broadcasts, at least $t+1$ have originated with honest parties. Let G_i be this set of at least $t+1$ honest parties, relative to party P_i.

For each party $P_j \in G_i$ there exist $t+1$ honest parties who have broadcasted (`OK,` P_j) . Now, assume event E occurs. By the properties of ICP, for each $P_j \in G_i$ there will be $t+1$ parties P_k who broadcast (P_k `authenticates` P_j) . Thus, the share of each party $P_j \in G_i$ will be considered legal by each honest party. Therefore every honest party will have $t+1$ legal shares in Step 3 of A-RS-Rec and will suggest some secret. Consequently, each honest party will have $n-t$ suggested secrets, and will complete protocol A-RS-Rec.

Correctness (1): If the dealer is honest and event E occurs, then each share ϕ_j in the set IS_i of each honest P_i, has originated with D (namely, $\phi_j = f(j)$). Therefore, the shares in IS_i define the secret, s, shared by the dealer, and P_i will broadcast (P_i `suggests secret` s) . Consequently, each honest party will complete $n-t$ such broadcasts, out of which at least $n-2t$ suggest the secret s. Hence, each honest party will output s.

Correctness (2): Let P_i be the first honest party to complete $n-t$ broadcasts of the form (P_k `suggests secret ...`) . Let S denote the set of secrets suggested in these $n-t$ broadcasts. Now, consider another honest party, P_j. At least $n-2t$ out of the $n-t$ broadcasts completed by P_j in Step 5 of A-RS-Rec are of values in S. P_j outputs a non-null value r only if this value appears $n-2t$ times in the broadcasts which he completes. However, in this case it must be that $r \in S$.

Secrecy: If the dealer is honest and no honest party has initiated protocol A-RS-Rec, then any set of $t' \leq t$ parties have only t' shares of the secret, along with the data received during the ICP Generation and Verification Protocols with respect to the other parties' shares of the secret. The secrecy properties of ICP and Shamir's secret sharing scheme imply that the shared secret is independent of this data. □

5.5 Asynchronous Weak Secret Sharing — AWSS

We construct an asynchronous secret sharing scheme called Asynchronous Weak Secret Sharing (AWSS). An AWSS scheme satisfies the *termination* and *secrecy* requirements of AVSS (Definition 5.2 on page 102). The correctness requirement is as follows. If the dealer is honest then the honest parties always reconstruct the secret shared by the dealer, as in AVSS. The difference from AVSS lies in the case where the dealer is faulty. For AWSS we only require that:

Correctness of AWSS for the case that the dealer is faulty: *Once an honest party completes the sharing protocol, a value $r \in \mathcal{F} \cup \{null\}$ is fixed. With probability $1-\epsilon$, each honest party outputs either r or nullupon completing the reconstruction protocol.*

Remarks:

- We stress that if $r \neq null$ then some of the honest parties may output r and some may output *null*. The adversary can decide, during the execution of the reconstruction protocol, which parties will output *null*.

- Our construction of A-RS, described in the previous section, does not satisfy the requirements from an AWSS scheme. There, in case that the dealer is faulty, the adversary has the power to decide, *at the reconstruction stage*, which value each honest party reconstructs (out of the predefined set of size $2t+1$).

Our construction. Our construction follows the synchronous WSS version of [TRa, RB]. The idea is to make sure that, at the end of the sharing protocol, each party will have an interpolation set of parties whose shares will be available in the reconstruction stage. (The availability of these shares is guaranteed using A-RS, as described below.) The interpolation sets of every two honest parties P_i and P_j will have an intersection of at least $t+1$ parties. This way, if the shares of the parties in the interpolation sets of P_i and P_j define some secret, then they define the same secret.

In the reconstruction stage the shares of all the parties are jointly reconstructed. Next, each party computes and outputs the secret defined by the shares of the parties in his interpolation set. If these shares do not define a secret then the party outputs *null*.

Sharing protocol. The dealer, sharing a secret $s \in \mathcal{F}$, chooses a random polynomial $f(x)$ of degree t for s. For each i, the dealer sets $\phi_i \stackrel{\triangle}{=} f(i)$. In addition, for each share ϕ_i and for each party P_j, the dealer executes the generation phase of ICP with party P_i acting as the Intermediary, and party P_j acting as the Receiver. (This part is the same as in A-RS-Share.)

Next, each party shares each value received from the dealer (including the values received in the ICP-Generation Protocol), using A-RS-Share (Figure 5-2). Party P_i creates a dynamic set, $\mathcal{C}_i$. (See Section 4.1.3 on page 53 for a definition of dynamic sets). Once P_i completes all the A-RS-Shares of P_j's values, P_j is added to $\mathcal{C}_i$. Next party P_i initiates the ICP-Verification Protocol for ϕ_i with each party $P_j \in \mathcal{C}_i$, and with the appropriate allowed-error parameter, ϵ'. (We denote this execution by $\mathbf{Ver}_{i,j}[\epsilon'](\phi_i)$).

Let A_i be P_i's set of parties P_j such that $\mathbf{Ver}_{i,j}[\epsilon'](\phi_i) = 1$. Once $|A_i| = 2t+1$, party P_i broadcasts A_i. (This *acceptance set*, A_i, is the set of parties who will later accept ϕ_i with high probability.)

Let $\mathcal{E}_i$ be P_i's dynamic set of parties $P_j \in \mathcal{C}_i$ whose broadcast of A_j has been completed, and $A_j \subseteq \mathcal{C}_i$. Once $|\mathcal{E}_i| \geq 2t+1$, party P_i broadcasts $IS_i \stackrel{\triangle}{=} \mathcal{E}_i^{(2t+1)}$. (This *eligibility set*, $\mathcal{E}_i$, is the set of parties whose share will be later considered either legal or *null*. The *interpolation set*, IS_i, is the set of parties whose shares will later define the secret associated with P_i.)

Let F_i be P_i's set of parties P_j whose IS_j broadcast has been completed, and $IS_j \subseteq \mathcal{E}_i$. Once $|F_i| = n-t$, party P_i completes the sharing protocol. (This *final set*, F_i, is the set of parties, with whom P_i will later associate a secret.)

Reconstruction protocol. (Remark: it will be seen that when party P executes the reconstruction protocol, he essentially doubles up as each other party, i.e. he locally executes the protocols of the other parties). Initially, each party P_i reconstructs the values shared, using A-RS-Share, by each party $P_j \in \mathcal{E}_i$, as follows. Recall that each party in $\mathcal{E}_i$ shared, in the sharing protocol, values he received as an intermediary of ICP, and values he received as a receiver of ICP. Party P_i first invokes A-RS-Rec for all the values that the parties in $\mathcal{E}_i$ received as intermediaries in ICP. Call these invocations of A-RS-Rec *preliminary*. For each party $P_j \in \mathcal{E}_i$, once all the preliminary A-RS-Rec have been completed, P_i invokes the A-RS-Rec of (the rest of) the values shared by each $P_l \in A_j$. *We note that the order in which the A-RS-Rec are invoked is crucial for the correctness of the scheme.*

For each party $P_j \in \mathcal{E}_i$ and for each party $P_l \in A_j$, once all the necessary values are reconstructed, P_i computes P_l's expected result of $\mathbf{Auth}_{j,l}[\epsilon']$ (by 'doubling up' as P_l). Next, P_i associates a value with party P_j as follows. If P_i finds out that $t+1$ parties in A_j have outputted $\mathbf{Auth}_{j,l} = \phi_j$ then P_i associates ϕ_j with P_j. (We then say that ϕ_j is *legal*). Otherwise, P_i associates the value "null" with P_j.

Now, for each party $P_k \in F_i$, once the values associated with all the parties 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.*

Once the values shared by all the parties in $\mathcal{E}_i$ are reconstructed, P_i computes the secrets suggested by all the parties in F_i. (Again, this is done by 'doubling up' as each party in F_i.) If all the parties in F_i suggest the same secret, s, then P_i outputs s. Otherwise, P_i outputs *null.* (Equivalently, P_i can output the secret defined by the reconstructed shares of the parties in $\cup_{P_j \in F_i} IS_j$. If these shares do not define a secret then P_i outputs *null.*)

The AWSS-Share and AWSS-Rec are described in Figures 5-4 and 5-5 respectively.

Protocol AWSS-Share$[\epsilon]$

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

1. Choose a random polynomial $f(x)$ for s. Set ϵ' and ϵ'' to the appropriate values, depending on ϵ.
2. Set $\phi_i \triangleq f(i)$. For each two parties P_i and P_j execute $\mathbf{Gen}_{D,i,j}[\epsilon''](\phi_i)$.

Code for party P_i, on parameter ϵ:

3. Invoke A-RS-Share$[\epsilon']$ (as a dealer) for each value received from D in Step 2. Participate in all other invocations of A-RS-Share$[\epsilon']$.

   Create a dynamic Communication Set, $\mathcal{C}_i$. Add party P_j to $\mathcal{C}_i$ if all the A-RS Shares initiated by P_j have been completed.
4. For each party P_j invoke $\mathbf{Ver}_{i,j}[\epsilon''](\phi_i)$.

   Create an Acceptance Set, A_i. Add party $P_j \in \mathcal{C}_i$ to A_i if $\mathbf{Ver}_{i,j}[\epsilon''](\phi_i) = 1$.
5. Once $|A_i| = 2t+1$ broadcast `(Acceptance Set,`P_i`,`A_i`)`.

   Create a dynamic Eligibility Set, $\mathcal{E}_i$, which will contain the parties P_j for whom the broadcast of the form `(Acceptance Set,`P_j`,`A_j`)` has been completed, and $P_j \in \mathcal{C}_i$ and $A_j \subseteq \mathcal{C}_i$.
6. Define the Interpolation Set $IS_i \triangleq \mathcal{E}_i^{(n-t)}$. Broadcast `(Interpolation set,`$P_i$`,`$IS_i$`)`.

   Create a Final Set, F_i, which will contain the parties P_j for whom the broadcast of the form `(Interpolation Set,`P_j`,`IS_j`)` has been completed, and $IS_j \subseteq \mathcal{E}_i$.
7. Once $|F_i| = n-t$ terminate.

Figure 5-4: The AWSS Sharing Protocol

Theorem 5.7 *Let $n \geq 3t+1$. Then for every $\epsilon > 0$, the pair* (AWSS-Share$[\epsilon]$,AWSS-Rec$[\epsilon]$) *is a $(1-\epsilon)$-correct, t-resilient AWSS scheme for n parties.*

Protocol AWSS-Rec[ϵ]

Code for party P_i, on parameter ϵ:

1. For each party $P_j \in \mathcal{E}_i$ and for each party $P_l \in A_j$:
   (a) Execute A-RS-Rec[ϵ'] (Figure 5-3) of ϕ_j, and of P_j's values needed for $\mathbf{Auth}_{j,l}[\epsilon'']$. (These values were shared by P_j in Step 3 of AWSS-Share.)
   (b) Upon ending the previous step, execute the A-RS-Rec of P_l's values needed for $\mathbf{Auth}_{j,l}[\epsilon'']$ (These values were shared by P_l in Step 3 of AWSS-Share.)
   (c) Using the reconstructed values from the two previous steps simulate $\mathbf{Auth}_{j,l}[\epsilon'']$ acting both as P_j and P_l. If at least $t+1$ $\mathbf{Auth}_{j,*}[\epsilon''] = (\phi_j)$ then associate the share ϕ_j with party P_j, else associate "null" with P_j.
2. Let $B = \{\phi \mid \phi \neq null,\ \phi \text{ is associated with party} P_l, \text{and } P_l \in IS_j,\ P_j \in F_i\}$, If all the shares of B define a secret s, then output s, else output *null*.

Figure 5-5: The AWSS - Reconstruction Protocol

As in the proof of Theorem 5.6, let E be the event that no errors occur in all the invocations of ICP. (That is, all the requirements listed in Section 5.3.1 hold with no probability of error.) In proving Theorem 5.7 we assume that event E occurs. For convenience and clarity, we partition the proof of Theorem 5.7 to several short lemmas. (The Termination and Secrecy properties are taken from Definition 5.2.)

Lemma 5.8 *If P_i, P_j are honest, then eventually $\mathcal{C}_i = \mathcal{C}_j$.*

Proof: Let us look at some $P_l \in \mathcal{C}_i$. If party P_i has placed P_l in $\mathcal{C}_i$ (Step 3) then P_i has completed P_l's A-RS-Share of Step 3. Due to the Termination property of the A-RS protocol we know that eventually P_j will also conclude that A-RS-Share has ended and then P_j will add P_l to $\mathcal{C}_j$. □

Lemma 5.9 *If P_i, P_j are honest, then eventually $\mathcal{E}_i = \mathcal{E}_j$.*

Proof: Let us look at some $P_l \in \mathcal{E}_i$. If party P_i placed P_l in $\mathcal{E}_i$ then P_i has completed the broadcast `(Acceptance set,`P_l`,`A_l`)` and $P_l \in \mathcal{C}_i$ and $A_l \subseteq \mathcal{C}_i$. Due to the broadcast property P_j will also complete the above broadcast, and eventually we will have (due to Lemma 5.8) $P_l \in \mathcal{C}_i = \mathcal{C}_j$, and $A_l \subseteq \mathcal{C}_i = \mathcal{C}_j$. Then, P_j will add P_l to $\mathcal{E}_j$. □

Lemma 5.10 (Termination (1):) *If the dealer is honest then each honest party will complete protocol AWSS-Share.*

Proof: All honest parties will eventually be in the sets $\mathcal{C}_i$ and A_i of every honest party P_i. Thus A_i will eventually be of size $2t+1$, and every honest party P_i will broadcast `(Acceptance Set,`P_i`,...)` in Step 5 of AWSS-Share. Thus all honest parties will eventually be in the set $\mathcal{E}_i$ of every honest party. Consequently, every honest party P_i will broadcast `(Interpolation Set,`P_i`,`IS_i`)` in Step 6. Thus every honest party will eventually have $|F_i| \geq 2t+1$ and will complete protocol AWSS-Share. □

Lemma 5.11 (Termination (2)): *If some honest party, P_i, completes protocol AWSS-Share, then each honest party P_j will eventually complete protocol AWSS-Share.*

Proof: Assume that party P_i has completed AWSS-Share. Thus, $|F_i| = n-t$, which means that P_i has completed $n-t$ broadcasts of the form `(Interpolation set,`P_l`,` IS_l`)`, and $IS_l \subseteq \mathcal{E}_i$. Due to the properties of broadcast P_j will eventually complete these invocations of broadcast. Due to Lemma 5.9, we have that eventually $\mathcal{E}_j \subseteq \mathcal{E}_i$, hence, the requirement that $IS_l \subseteq \mathcal{E}_j$ will be satisfied and P_j will complete AWSS-Share. □

Lemma 5.12 (Termination (3)): *If one honest party has completed protocol AWSS-Share, then each honest party will complete protocol AWSS-Rec.*

Proof: Consider an honest party P_i. For each $P_j \in F_i$ we have $P_j \in \mathcal{C}_i$, thus P_i has completed all the A-RS-Share where P_j is the dealer. Thus P_i will complete, in Steps 1(a) and 1(b) of AWSS-Rec, the A-RS-Rec of all the values shared by P_j. Consequently, P_i will associate a value, in Step 2 of AWSS-Rec, with each party $P_j \in F_i$, and will complete AWSS-Rec. □

The Correctness property is proven in Lemma 5.13 through Lemma 5.16.

Lemma 5.13 *Let ϕ_j be the share that party P_j received from the dealer in Step 2 of AWSS-Share. With overwhelming probability, the value which an honest party P_i associates with a party $P_j \in \mathcal{E}_i$ in Step 1(c) of AWSS-Rec is as follows:*

	honest dealer	faulty dealer
P_j honest	ϕ_j	ϕ_j
P_j faulty	ϕ_j *or null*	—

Proof: Party P_i will associate a value ϕ'_j with party P_j if the following two requirements hold:

1. ϕ'_j is the value which was reconstructed by the A-RS-Rec of P_j's share.
2. There are at least $t+1$ parties in A_j for whom $\mathbf{Auth}_{j,*} = \phi'_j$.

Consider first the case where P_j is honest. In this case P_j shared the value ϕ_j in Step 3 of AWSS-Share. From the correctness of A-RS we have that P_i will reconstruct, in Step 1(a) of AWSS-Rec, the correct values of ϕ_j and of P_j's data needed for executing $\mathbf{Auth}_{j,l}$ for each party $P_l \in A_j$. Furthermore, at least $t+1$ out of the parties in A_j are honest. The data of these $t+1$ parties will also be correctly reconstructed, in Step 1(b) of AWSS-Rec. Thus it follows from the properties of ICP that P_i will successfully verify, in Step 1(c), that $\mathbf{Auth}_{j,l} = \phi_l$ for at least $t+1$ parties $P_l \in A_j$, and will associate ϕ_j with P_j.

Consider the case where P_j is faulty and the dealer is honest, and assume that the value ϕ'_j reconstructed in the A-RS-Rec of P_j's share is different than ϕ_j. Party P_i will, in Step 1(c) of AWSS-Rec, asociate a non-null value with P_j only if P_i has $\mathbf{Auth}_{j,l} = \phi'_j$ for at least $t+1$ parties P_l; one of these $t+1$ parties must be honest. We show that if P_l is honest then P_i computes $\mathbf{Auth}_{j,l} = \phi'_j$ only with negligible probability.

Let $\mathrm{I}_{i,j,l}$ (resp., $\mathrm{R}_{i,j,l}$) denote the value that P_i reconstructs, in Step 1(a) (resp., 1(b)) of AWSS-Rec, for P_l's (resp., P_j's) data relevant to $\mathbf{Auth}_{j,l}$. (This data was respectively shared by P_j and P_l, in Step 3 of AWSS-Share. Recall that here P_j acts as the Intermediary in ICP, and P_l acts as the Receiver.) Since P_l is honest, A-RS ensures that $\mathrm{R}_{i,j,l}$ is indeed P_l's data in $\mathbf{Auth}_{j,l}$. Therefore, if it were true that $\mathrm{I}_{i,j,l}$ is determined by the adversary independently of $\mathrm{R}_{i,j,l}$, then Property 3 of ICP (see Section 5.3) would imply that P_i sets $\mathbf{Auth}_{j,l} = \phi'_j$ with negligible probability. However, since P_j is faulty, A-RS ensures only a weaker requirement on $\mathrm{I}_{i,j,l}$. We show that this property suffices.

A-RS allows $\mathrm{I}_{i,j,l}$ to be set by the adversary during the reconstruction stage. However, once the first honest party completes A-RS-Rec, a small set S of $2t+1$ possible values for $\mathrm{I}_{i,j,l}$ is fixed. By the time that S is fixed no honest party has yet started the reconstruction of $\mathrm{R}_{i,j,l}$ (i.e., of the data shared by P_l in Step 3 of AWSS-Share). Thus, $\mathrm{R}_{i,j,l}$ is unknown to the adversary when S is fixed. It follows that the probability that for a given P_i and P_j there exist an honest P_l and a value $\phi'_j \neq \phi_j$ such that P_i computes $\mathbf{Auth}_{j,l} = \phi'_j$ is exponentially small. □

We define the following value, associated with each party P_i who has completed AWSS-Share. Let

$$V_i = \bigcup_{P_j \in F_i} \{(k, \phi_k) \mid \phi_k \text{ is the share of an } \textit{honest} \text{ party } P_k,\ P_k \in IS_j\}.$$

Let

$$r_i = \begin{cases} v_i & \text{if } V_i \text{ defines a secret } v_i \\ null & \text{otherwise} \end{cases}$$

We stress that P_i does not know V_i. Still, V_i is a useful tool in our analysis.

We set the value r from the Correctness requirement to be the value r_i associated with the first honest party P_i who completes AWSS-Share with $r_i \neq$*null*. Note that $|V_i| \geq t+1$, and that V_i (and consequently r_i) are defined once P_i has completed AWSS-Share.

Lemma 5.14 *If the values r_i and r_j of two honest parties P_i and P_j do not equal null, then $r_i = r_j$.*

Proof: Consider a party $P_l \in F_i \cap F_j$. ($F_i \cap F_j \neq \emptyset$ since both F_i and F_j are of size $n-t$.) In IS_l there are at least $t+1$ honest parties, the shares of which define a single secret. Since V_i and V_j define a secret, it must be the same secret defined by the shares in IS_l. Hence, $r_i = r_j$. □

Lemma 5.15 *Every honest party P_i outputs, upon completing AWSS-Rec, either r_i or null.*

Proof: It follows from Lemma 5.13 that the value that P_i associates with each honest party in F_i is ϕ_i, even if the dealer is faulty. Note that F_i contains at least $t+1$ honest parties. Thus if the values associated with the parties in F_i define a secret then this secret is r_i. Consequently P_i outputs either r_i or *null.* □

Lemma 5.16 *If the dealer is honest, sharing a secret s, then every honest party P_i outputs s.*

Proof: It follows from the definition of r_i that if the dealer is honest then $r_i = s$. Furthermore, Lemma 5.13 implies that if the dealer is honest then the values associated with the parties in F_i always define a secret. Since the dealer is honest, P_i does *not* output *null.* It now follows from Lemma 5.15 that P_i outputs s. □

Lemma 5.17 (Secrecy): *If the dealer is honest then the information gathered by the adversary during AWSS-Share is independent of the shared secret.*

Proof: The proof is the same as the proof of Secrecy for the A-RS protocol (Theorem 5.6). We add that the distribution of the information gathered by the adversary in the invocations of A-RS-Share during AWSS-Share is independent of the shared secret. □

5.5.1 Two&Sum-AWSS

In our AVSS scheme we do not use AWSS as such. Instead, we use a slight variation of AWSS, called Two&Sum-AWSS. A dealer shares a secret s, together with a set $\{a_1, \ldots, a_k\}$ of *auxiliary secrets,* in a way that allows the parties to separately reconstruct the secret, any one of the auxiliary secrets, and the sum $s + a_i$ of the secret with any one of the auxiliary secrets. (That is, *several* invocations of the reconstruction protocol will use the shares obtained in a *single* invocation of the sharing protocol.) For each i, reconstructing any single value out of $\{s, a_i, s + a_i\}$ reveals no information on the other two. The correctness property of Two&Sum-AWSS is stated more formally as follows.

Correctness of Two&Sum-AWSS:

1. *If the dealer is honest, sharing $s, \{a_1, \ldots, a_k\}$, then, with probability $1 - \epsilon$, when reconstructing the secret (resp., the ith auxiliary secret, or the sum of the secret and the ith auxiliary secret), each honest party outputs s (resp., a_i, or $s + a_i$).*
2. *Once an honest party completes the sharing protocol, the values*

$$r_s, \{r_{a_1}, \ldots, r_{a_k}\}, \{r_{s+a_1}, \ldots, r_{s+a_k}\} \in \mathcal{F} \cup \{null\}$$

   are fixed.

   (a) *With probability $1 - \epsilon$, when reconstructing the secret (resp., the ith auxiliary secret, or the sum of the secret and the ith auxiliary secret), each honest party outputs either r_s (resp., r_{a_i}, or r_{s+a_i}) or null.*

   (b) *If r_s is null then, for each i, at least one of $\{r_{a_i}, r_{s+a_i}\}$ is null. If r_s, r_{a_i} and r_{s+a_i} are not null then $r_{s+a_i} = r_s + r_{a_i}$.*
3. *Even if for each i one value out of $\{a_i, s + a_i\}$ are reconstructed, the adversary gathers no information about the secret s.*

Our construction of Two&Sum-AWSS, presented in Figure 5-6, is a straightforward generalization of our AWSS scheme. (The scheme is based on the *synchronous* Two&Sum-WSS presented in [TRa].)

The reconstruction protocol is identical to AWSS-Rec, with the addition of parameters specifying which value should be reconstructed. We denote by AWSS-Rec$_{\mathrm{S}}$ (resp., AWSS-Rec$_{\mathrm{A}_i}$, AWSS-Rec$_{\mathrm{S+A}_i}$) the invocation for reconstructing the secret (resp., the ith auxiliary secret, the sum of the secret and the ith auxiliary secret).

The correctness of Two&Sum AWSS is proven in a way similar to the proof of correctness of the AWSS scheme. (Similar proofs appear in [TRa].)

Protocol Two&Sum AWSS-Share$[\epsilon, k]$

Code for the dealer: (on input $s, a_1, \ldots, a_k$ and ϵ)

1. Choose random polynomials $f(\cdot)$ for s and $g_l(\cdot)$ for each a_l. Let $\phi_i \triangleq f(i)$ and $\gamma_{l,i} \triangleq g_l(i)$ for $0 \leq l \leq k$ and $1 \leq i \leq n$. Let $\sigma_{l,i} \triangleq \phi_i + \gamma_{l,i}$.
   Send $\phi_i, \{\gamma_{1,i} \ldots, \gamma_{k,i}\}$ to each party P_i.
2. For every two parties P_i, P_j, invoke: $\mathbf{Gen}_{D,i,j}(\phi_i)$, $\mathbf{Gen}_{D,i,j}(\gamma_{1,i}), \ldots \mathbf{Gen}_{D,i,j}(\gamma_{k,i})$, and $\mathbf{Gen}_{D,i,j}(\sigma_{1,i}), \ldots \mathbf{Gen}_{D,i,j}(\sigma_{k,i})$.

Code for party P_i

3. Invoke A-RS-Share (as a dealer) for each value received from the dealer in Steps 1 and 2. Participate in all other invocations of A-RS-Share.
   Create a dynamic Communication Set, $\mathcal{C}_i$, which is the set of all parties for whom all the invocations of A-RS Share have been completed.
4. For each party P_j invoke $\mathbf{Ver}_{D,i,j}(\phi_i)$, $\mathbf{Ver}_{D,i,j}(\gamma_{1,i}), \ldots \mathbf{Ver}_{D,i,j}(\gamma_{k,i})$, and $\mathbf{Ver}_{D,i,j}(\sigma_{1,i}), \ldots \mathbf{Ver}_{D,i,j}(\sigma_{k,i})$.
   Create an Acceptance Set, A_i. Add party $P_j \in \mathcal{C}_i$ to A_i if all the above invocations of $\mathbf{Ver}_{D,i,j}(*)$ were completed successfully.
5. Execute the rest of the code of AWSS-Share (i.e., Steps 5 through 7 of Figure 5-4).

Figure 5-6: The Two&Sum AWSS Sharing Protocol

5.6 Asynchronous Verifiable Secret Sharing — AVSS

The idea of the AVSS scheme is as follows. First, the dealer sends each party a share of the secret, as in the previous schemes. The parties will then 'commit' to their shares by re-sharing them using AWSS. Next, the parties will make sure, using a cut-and-choose method (as in [CCD, Fe]), that enough AWSS-Sharings have been successful and that the 'committed upon' shares indeed define a secret. We describe the scheme in some more detail. Full details appear in Figure 5-7.

Sharing protocol. The dealer, sharing a secret s, chooses a random polynomial $f(x)$ for s. For each i, the dealer sends $f(i)$ to P_i. In addition, the dealer chooses $k \cdot n \cdot t$ random polynomials $g_{1,1,1}(x), \ldots, g_{k,n,t}(x)$ of degree t, where k is an appropriate security parameter (polynomial in $\log 1/\epsilon$). For each such polynomial $g(x)$, and for each i, the dealer sends $g(i)$ to party P_i. Upon receiving all the expected values from the dealer, each P_i re-shares the values $f(i), \{g_{1,1,1}(i), \ldots, g_{k,n,t}(i)\}$ using Two&Sum-AWSS-Share with the appropriate allowed-error parameter (see Section 5.5.1).

Next, the dealer proves to the parties that their shares indeed define a secret. Each party P_i is looking for a set IS_i of at least $n-t$ parties, such that the Two&Sum-AWSS-Share's of these parties have been completed, and the corresponding values define a polynomial. For this purpose P_i participates in up to t iterations, as follows.[3] Initially, all parties are valid. At the beginning of each iteration P_i waits until he has completed the Two&Sum-AWSS-Share of $n-t$ valid parties. Let $IS_{i,r}$ denote this set of parties, in iteration r. Next P_i verifies, using cut-and-choose as described in the code, that the share of each party in $IS_{i,r}$ is valid (in a sense defined in Figure 5-7). P_i removes, from the set of valid parties, all the parties in $IS_{i,r}$ whose validation failed. If the validity of the shares of at least $n-t$ parties in $IS_{i,r}$ is confirmed, then we say that the iteration was successful. In this case, P_i broadcasts (P_i `confirms` P_j) for each party $P_j \in IS_{i,r}$. (Now P_i is assured that the shares of the parties in $IS_{i,r}$ define a polynomial.) When the verification procedure is completed, and P_i has completed the Two&Sum-AWSS-Share of at least one other valid party, P_i proceeds to the next iteration.

Let F_i be P_i's set of parties who were confirmed by at least $t+1$ parties. Party P_i completes the sharing protocol once $|F_i| \geq 2t+1$.

Reconstruction protocol. The reconstruction protocol is simple: The share of each party P_i is reconstructed using Two&Sum-AWSS-Rec$_{i,s}$.[4] Once $t+1$ shares of parties in F_i are reconstructed with a non-null value, party P_i computes the secret defined by these shares, and terminates.

Protocols AVSS-Share and AVSS-Rec are presented in Figures 5-7 and 5-8, respectively.

Theorem 5.18 *Let $n \geq 3t+1$. Then for every $\epsilon > 0$, the pair* (AVSS-Share[ϵ],AVSS-Rec[ϵ]) *is a $(1-\epsilon)$-correct, t-resilient AVSS scheme for n parties.*

---

[3] Partitioning the protocol into t iterations is done for clarity of exposition. All these iterations are completed in a constant number of asynchronous time steps.

[4] We let Two&Sum-AWSS-Rec$_{i,*}$ denote an invocation of Two&Sum-AWSS-Rec$_*$ that corresponds to the Two&Sum-AWSS-Share of P_i.

Protocol AVSS-Share[ϵ]

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

1. Choose a random polynomial $f(x)$ for s. Send $\phi_i \triangleq f(i)$ to each party P_i.
2. Set $k \triangleq O(n, \log \frac{1}{\epsilon})$. For $1 \leq l \leq k$, for $1 \leq j \leq n$, and for $1 \leq r \leq t$ do:
   Pick a random polynomial $g_{l,j,r}(x)$ of degree t, and send $g_{l,j,r}(i)$ to each party P_i.
3. Upon completing an broadcast
   `(Random vector,`$b_{1,j,r}, ..., b_{k,j,r}$`,iteration` r`,party,`P_j`)` (see Step 6a), broadcast
   `(Polynomials of` P_j`, iteration` r`,` $g_{1,i,r}(\cdot) + f(\cdot) \cdot b_{1,i,r}, ..., g_{k,i,r}(\cdot) + f(\cdot) \cdot b_{k,i,r}$`)`.

Code for party P_i, on parameter ϵ:

4. Set $\epsilon' = \epsilon/2n$. Share $\phi_i, \{g_{1,1,1}(i), \ldots, g_{k,n,t}(i)\}$ using Two&Sum-AWSS-Share[$\epsilon', n \cdot k \cdot t$] (see Figure 5-6). Participate in the Two&Sum-AWSS-Share of other parties.
5. Create a dynamic set $\mathcal{C}_i$. Add each party whose Two&Sum-AWSS-Share of the previous step has been completed to $\mathcal{C}_i$.
   Let V_i be the set of valid parties. Initially $V_i = \{1, \ldots, n\}$.
6. A local variable r is set to 0. Initialize sets $IS_{i,r} = F_i = \emptyset$.
   As long as $|F_i| < 2t + 1$, do:
   (a) Wait until $|\mathcal{C}_i \cap V_i| \geq n - t$ and $\mathcal{C}_i \cap V_i \neq IS_{i,r}$. Set $r := r+1$. Let $IS_{i,r} := \mathcal{C}_i \cap V_i$. Choose $b_{1,i,r}, \ldots, b_{k,i,r} \in_{\mathrm{R}} \{0,1\}$, and broadcast `(Random vector,`$b_{1,i,r}, ..., b_{k,i,r}$`, iteration` r`,party,`P_i`)`.
   (b) Upon completing an broadcast `(Polynomials of` P_j`,...)` (see Step 3), for $l = 1..k$, $m = 1..n$, if $b_{l,j,r} = 0$ execute Two&Sum-AWSS-Rec$_{m,\mathrm{A}_{l,j,r}}$. If $b_{l,j,r} = 1$ then execute Two&Sum-AWSS-Rec$_{m,\mathrm{S}+\mathrm{A}_{l,j,r}}$.
   (c) If any Two&Sum-AWSS-Rec of a share of party P_m terminates with *null*, or if any reconstructed share of P_m does not agree with the corresponding polynomial broadcasted by the dealer, then P_m is removed from V_i and from $IS_{i,r}$.
   Once all the relevant invocations of Two&Sum-AWSS-Rec of shares of parties in $IS_{i,r}$ are completed, if $|IS_{i,r}| \geq n - t$ then for each $P_m \in IS_{i,r}$ broadcast `(`P_i`,confirms party,`P_m`)`. (In this case we say that the iteration was successful.)

   * Once there are $t + 1$ broadcasts of the form `(...,confirms party,`P_m`)` for some P_m (see Step 6c) then add P_m to F_i.

Figure 5-7: The AVSS Sharing Protocol

Protocol AVSS-Rec[ϵ]

Code for party P_i, on parameter ϵ:

1. For each party P_j execute Two&Sum-AWSS-Rec$_{j,s}$. (P_j's share, ϕ_j, was shared by P_j in the Two&Sum-AWSS-Share of Step 4 of AVSS-Share.)
2. Take any $t+1$ reconstructed non-null shares of parties in F_i, and output the secret defined by these shares.

Figure 5-8: The AVSS Reconstruction Protocol

For convenience and clarity we partition the proof of Theorem 5.18 to several lemmas. Throughout the proof we assume that the following event E occurs. All invocations of Two&Sum-AWSS have been properly completed. That is, if an honest party has completed Two&Sum-AWSS-Share then all honest parties will complete all corresponding invocations of AWSS-Rec, outputting either the required value or *null*. (See the Correctness requirement of Two&Sum-AWSS, on page 115.) Event E occurs with probability at least $1-n\epsilon' = 1-\epsilon/2$.

Lemma 5.19 (Termination (1)): *If the dealer is honest then each honest party will complete protocol AVSS-Share.*

Proof: Each honest party P_i will complete the Two&Sum-AWSS-Share of the shares of each honest party P_j in Step 4. Thus P_j will eventually be in the set $IS_{i,r}$ of P_i for some iteration r. If the dealer is honest then P_j's reconstructed shares will agree, in Step 6c, with the polynomials broadcasted by the dealer. Thus, P_j will not be removed from $IS_{i,r}$ or V_i. Hence there will exist an iteration r where all $n-t$ honest parties are in $IS_{i,r}$, causing each honest party P_i to broadcast (P_i,`confirms`,P_j). Consequently, each honest party will be in the final set F_k of each honest P_k. Thus P_k will have $|F_k| \geq 2t+1$ and will complete AVSS-Share. We note that all honest parties terminate in a constant number of (asynchronous) rounds. □

Lemma 5.20 (Termination (2)): *If some honest party P_i completes protocol AVSS-Share, then each honest party will eventually complete protocol AVSS-Share.*

Proof: Assume an honest party P_i completed protocol AVSS-Share. Then $|F_i| \geq 2t+1$. It follows from the correctness properties of broadcast that, for any honest party P_j, any party in F_i will eventually be in F_j. Thus P_j will complete AVSS-Share. □

Lemma 5.21 (Termination (3)): *If AVSS-Share has been completed by the honest parties, then each honest party will complete protocol AVSS-Rec.*

Proof: Each honest party P_i has at least $t+1$ honest parties in F_i. It follows from the correctness of Two&Sum-AWSS that the share of each honest party $P_j \in F_i$ will be successfully reconstructed by P_i in Step 1 of AVSS-Rec. Thus, P_i will have at least $t+1$ non-null shares in Step 2 of AVSS-Rec, and will complete the protocol. □

Lemma 5.22 (Secrecy): *If the dealer is honest then the information gathered by the adversary during AVSS-Share is independent of the shared secret.*

Proof: Assuming that Two&Sum-AWSS-Share is secure, the relevant information gathered by the adversary in AVSS-Share consists of up to t shares of the polynomial $f(\cdot)$ shared by the dealer in Step 1, and up to t shares of each of the random polynomials shared in Step 2. Furthermore, for each random polynomial $g_*(\cdot)$, only one of $g_*(\cdot)$ or $f(\cdot) + g_*(\cdot)$ is known. It can be verified that this information is distributed independently of the shared secret. □

The Correctness property is proven via Lemmas 5.23 through 5.25.

Lemma 5.23 *Once an honest party P_i executes Step 6a of AVSS-Share in iteration r, a value $\phi'_m \in \mathcal{F} \cup null$ is fixed for each $P_m \in IS_{i,r}$. Furthermore, if iteration r was successful (as defined in Step 6c of AVSS-Share) then the following properties hold with overwhelming probability:*

1. *For each $P_m \in IS_{i,r}$, $\phi'_m \neq null$.*
2. *The set $C_{i,r} \triangleq \{(m, \phi'_m) | P_m \in IS_{i,r}\}$ defines a secret s'.*
3. *If the dealer is honest then s' is the secret s shared by the dealer.*
4. *When reconstructing P_m's share using Two&Sum-AWSS-Rec$_{m,\mathrm{s}}$ in Step 1 of AVSS-Rec, each honest party outputs $\phi'_m \cup null$.*

Proof: Define the value ϕ'_m to be the value fixed for the secret shared by P_m in the Two&Sum-AWSS-Share of Step 4. Once P_i executes Step 6a, in iteration r, he has completed the Two&Sum-AWSS-Share of each $P_m \in IS_{i,r}$. It follows from the correctness of Two&Sum-AWSS that the value ϕ'_m is fixed. Part 4 of the lemma also follows.

Part 1. It follows from the correctness of Two&Sum-AWSS that if $\phi'_m = null$ then for each $l = 1..k$ at least one out of {Two&Sum-AWSS-Rec$_{m,\mathrm{A}_{l,i,r}}$, Two&Sum-AWSS-Rec$_{m,\mathrm{S}+\mathrm{A}_{l,i,r}}$} will have output *null*. Thus for each $l = 1..k$ P_i will have *null* output with probability at least $\frac{1}{2}$. The probability that P_i has, in Step 6c, non-null output of all the k invocations of Two&Sum-AWSS-Rec is at most 2^{-k}. Thus, executions where $\phi'_m = null$ and iteration r is successful occur only with negligible probability.

Part 2. We use the same cut-and-choose argument as in part 1. Assume that $C_{i,r}$ does not define a secret. It follows that for each $l = 1..k$ there exists a P_m such that either the output of Two&Sum-AWSS-Rec$_{m,\mathrm{A}_{l,i,r}}$ is not equal to $h_{l,i,r}(m)$ or the output of Two&Sum-AWSS-Rec$_{m,\mathrm{S}+\mathrm{A}_{l,i,r}}$ is not equal to $h_{l,i,r}(m)$ (where $h_{l,i,r}$ is the corresponding polynomial broadcasted by the dealer). Thus for each $l = 1..k$ P_i will detect an error with probability at least $\frac{1}{2}$. The probability that P_i has not detected an error in all the k invocations of Two&Sum-AWSS-Rec in Step 6c is at most 2^{-k}. Thus, iteration r is successful only with negligible probability.

Part 3. The correctness of Two&Sum-AWSS assures that the value reconstructed in Two&Sum-AWSS-Rec$_{m,\mathrm{S}+\mathrm{A}_{l,i,r}}$ equals the sum of the values reconstructed in Two&Sum-AWSS-Rec$_{m,\mathrm{A}_{l,i,r}}$ and in Two&Sum-AWSS-Rec$_{m,\mathrm{S}}$. Since the dealer is honest, he broadcasts, in Step 3 of AVSS-Share, the same polynomials that he shared in Step 1. Part 3 follows using the same cut-and-choose argument as in parts 1 and 2. □

Lemma 5.24 *Let $C \triangleq \{(m, \phi'_m) | P_m \in F_i$ and P_i is honest$\}$, where ϕ'_m is the value fixed for P_m (See Lemma 5.23). Then the following properties hold with overwhelming probability:*

1. *The set C defines a secret s'.*

2. *If the dealer is honest then s' is the shared secret, s.*

Proof: **1.** Consider two parties P_{m_1} and P_{m_2} such that $(m_1, \phi'_{m_1}), (m_2', \phi'_{m_2}) \in C$. Then both P_{m_1} and P_{m_2} were confirmed by honest parties. Assume P_{m_1} was confirmed by honest party P_{j_1} in iteration r_1, and P_{m_2} was confirmed by honest party P_{j_2} in iteration r_2. Let $C_{j_1,r_1} \triangleq \{(m, \phi'_m) | P_m \in IS_{j_1,r_1}\}$, and let $C_{j_2,r_2} \triangleq \{(m, \phi'_m) | P_m \in IS_{j_2,r_2}\}$. Then the intersection $I \triangleq C_{j_1,r_1} \cap C_{j_2,r_2}$ is of size at least $t+1$. Lemma 5.23 implies that the values in I are non-null with overwhelming probability, even if the corresponding parties are faulty. It follows that C_{j_1,r_1} and C_{j_2,r_2} define the same secret. Part 1 follows.
Part **2** follows from part 3 of Lemma 5.23. □

Lemma 5.25 *Let r denote the secret defined by the set C (see Lemma 5.24). An honest party has output different than r of AVSS-Rec only with negligible probability.*

Proof: Consider an honest party P_i. The set $\{(m, \phi'_m) | P_m \in F_i\}$ is a subset of C. Thus it defines the same secret r. P_i will have at least $t+1$ honest parties P_j in F_i; the corresponding reconstructed values ϕ'_j will be non-null. Thus P_i will be able to interpolate a value, and this value will be r. □

5.7 Common Coin

We define an asynchronous common coin primitive, and describe an construction. We employ the AVSS scheme described in Section 5.6. We first present a definition of a common coin primitive.

Definition 5.26 *Let π be a protocol, where each party has local random input and binary output. We say that π is a $(1-\epsilon)$-*terminating*, t-*resilient* Common Coin *protocol if the following requirements hold for every t-adversary:*

- **Termination.** *With probability $1-\epsilon$, all the honest parties terminate.*

- **Correctness.** *For every value $\sigma \in \{0,1\}$, with probability at least $\frac{1}{4}$ all the honest parties output σ.*

Our construction (with 'termination parameter' ϵ). Roughly speaking, the protocol consists of two stages. First, each party shares n random secrets, using the AVSS-Share protocol of our AVSS scheme, with allowed error parameter $\epsilon' \triangleq \frac{\epsilon}{n^2}$. Say that the ith secret shared by each party is assigned to party P_i. Once a party, P_i, completes $t+1$ AVSS-Share protocols of secrets assigned to him, he broadcasts the identity of the dealers of these secrets. We say that these $t+1$ secrets are attached to P_i. (Later, the value associated with P_i will be computed based on the secrets attached to him.)

Upon completing the AVSS-Share of all the secrets attached to some P_j, party P_i is certain that a fixed (and yet unknown) value is attached to P_j. (The way in which this value will be computed is described in the protocol.) Once P_i is assured that the values attached to enough parties has been fixed, he starts reconstructing the relevant secrets. (This process of ensuring that enough values have been fixed is at the heart of the protocol.) Once all the relevant secrets are reconstructed, each party locally computes his output based on the reconstructed secrets, in a special way described in the sequel. The protocol is presented in Figure 5-9 below.

Protocol Common-Coin$[\epsilon]$

Code for party P_i, given 'termination' parameter, ϵ:

1. Let $\epsilon' \triangleq \frac{\epsilon}{n^2}$. For $1 \leq j \leq n$, choose a random secret $x_{i,j} \in_R \mathcal{F}$ and execute AVSS-Share for this value. Denote the execution by AVSS-Share$_{i,j}(x_{i,j})$.
   Participate in all the other AVSS-Share protocols.
2. Create a dynamic communication set $\mathcal{C}_i$. Add party P_j to $\mathcal{C}_i$ if AVSS-Share$_{j,l}$ has been completed for all $1 \leq l \leq n$.
   Wait until $|\mathcal{C}_i| \geq t+1$. Then, set $C_i = \mathcal{C}_i^{(t+1)}$ and broadcast `(Attach` A_i `to` P_i`)` .
   (We say that the secrets $\{x_{j,i} | P_j \in C_i\}$ are the secrets attached to party P_i.)
3. Accept a party P_j if the broadcast `(Attach` A_j `to` P_j`)` has been completed, and $C_j \subseteq \mathcal{C}_i$. Let $\mathcal{G}_i$ be the dynamic set of accepted parties.
   Wait until $|\mathcal{G}_i| \geq n-t$. Then, let $G_i = \mathcal{G}_i^{(n-t)}$ and broadcast `(`P_i `accepts` H_i`)` .
4. Say that party P_j is supportive, if the `(`P_j `accepts` H_j`)` broadcast has been received, and each party in G_j is accepted (namely, if $G_j \subseteq \mathcal{G}_i$).
   Wait until $n-t$ parties are supportive. Then, raise flag `'reconstruct enabled'` .
   Let H_i denote the current contents of $\mathcal{G}_i$.
   (Note that a party P_j who was not considered supportive since some $P_k \in G_j$ was not in $\mathcal{G}_i$ can become supportive later if P_k is added to $\mathcal{G}_i$.)
5. Wait until the flag `'reconstruct enabled'` is raised. Then, reconstruct the secrets attached to all the accepted parties. That is, for each $P_k \in C_j$ such that $P_j \in \mathcal{G}_i$ invoke AVSS-Rec$_{k,j}$, and let $r_{k,j}$ be the corresponding output.
   (Note that some parties may become accepted *after* the flag `'reconstruct enabled'` has been raised. The corresponding AVSS-Rec protocols are invoked immediately.)
6. Let $u \triangleq \lceil 0.87n \rceil$. For every party $P_j \in \mathcal{G}_i$, let V_j, the value associated with P_j, be the sum modulo u of all the secrets attached to P_j. That is, $V_j = (\sum_{k \in C_j} r_{k,j}) \bmod u$.
7. Wait until the values associated with all the parties in G_i are computed. If there exists a party $P_j \in H_i$ where $V_j = 0$, output 0.
   Otherwise, output 1.

Figure 5-9: The Common Coin protocol

Theorem 5.27 *Let $n \geq 3t+1$. Then, for every $0 < \epsilon \leq 0.2$ protocol* Common-Coin *is a $(1-\epsilon)$-terminating, t-resilient common coin protocol for n parties.*

The Termination property is asserted in Lemma 5.28. The Correctness property is asserted in Lemmas 5.29 through 5.31. Throughout the proof we assume that the following event E occurs. All invocations of AVSS have been properly completed. That is, if an honest party has completed AVSS-Share then a value s' is fixed. All honest parties will complete the corresponding invocation of AVSS-Rec, outputting s'. If the dealer is honest then s' is the shared secret. (See Definition 5.2.) Event E occurs with probability at least $1 - n^2\epsilon' = 1 - \epsilon$.

Lemma 5.28 *All the honest parties complete protocol* Common-Coin$[\epsilon]$ *in constant time.*

Proof: Assume event E occurs, and let P_i be an honest party. Then, P_i will complete all the AVSS-Share protocols initiated by honest parties in Step 1. Thus, $\mathcal{C}_i$ will eventually be of size $t+1$, and Step 2 will be completed.

For every honest party P_j, the broadcast `(Attach` A_j `to` P_j`)` will be received by P_i. Furthermore, since P_j completed the AVSS-Share$_{k,j}$ protocol for every $P_k \in C_j$, then P_i will complete these AVSS-Share$_{k,j}$ protocols as well. Therefore, every honest party will eventually be considered accepted by P_i (namely, added to the set $\mathcal{G}_i$). Thus, $\mathcal{G}_i$ will eventually be of size $n-t$, and Step 3 will be completed. Similar reasoning implies that every honest party will eventually be considered supportive by every honest party in Step 4. Consequently, every honest party will raise his `'reconstruct enabled'` flag, and will invoke his AVSS-Recs of Step 5.

It remains to be shown that all the AVSS-Rec protocols invoked by each honest party will be completed. If an honest party received an `(Attach` A_j `to` P_j`)` broadcast, then all the honest parties will receive this broadcast. Thus, if an honest party invokes AVSS-Rec$_{j,k}$, then all the honest parties will invoke AVSS-Rec$_{j,k}$. Event E now assures us that all the honest parties will complete all their AVSS-Rec protocols. Therefore, all the honest parties will execute Step 6 and complete the protocol. (The invocations of AVSS-Share with faulty dealers need not be terminated. Once an honest party completes Step 6, he may abort all non-terminated invocations of AVSS-Share.)

Given event E, all invocations of AVSS-Share and AVSS-Rec terminate in constant time. Protocol broadcast terminates in constant time. Thus, protocol Common-Coin terminates in constant time as well. □

Lemma 5.29 *Let* $u \triangleq \lceil 0.87n \rceil$. *Let* P_i *be a party whose broadcast* `(Attach` A_i `to` P_i`)` *in Step 2 has been completed by some honest party. Then, there exists a value,* v_i, *such that all the honest parties associate* v_i *with* P_i *in Step 6. Furthermore,*

- v_i *is fixed once the first honest party has completed the* `(Attach` A_j `to` P_j`)` *broadcast.*
- v_i *is distributed uniformly over* $[1 \ldots u]$, *and is independent of the values associated with the other parties.*

Proof: Let P_i be a party whose `(Attach` A_i `to` P_i`)` broadcast of Step 2 has been completed by some honest party. Then, all the honest parties will complete this broadcast with output C_i. Furthermore, The definition of AVSS assures us that for each party $P_j \in C_i$, there exists a fixed value, $r_{j,i}$, such that all the honest parties reconstruct have $r_{j,i}$ as their output of AVSS-Rec$_{j,i}[\epsilon']$ in Step 5 (that is, $r_{j,i}$ is the value shared by P_j, and attached to P_i.) Consequently, the value that each honest party associates with P_i in Step 6 is $\sum_{P_j \in C_i} r_{j,i} \bmod u$; let v_i be this value. This value is fixed once C_i is fixed, namely by the time that the first honest party has completed the broadcast `(Attach` A_j `to` P_j`)` .

In remains to show that v_i is uniformly distributed over $[1 \ldots u]$, and is independent of the values associated with the other parties. Recall that an honest party starts reconstructing the secrets attached to P_i (namely, invokes the AVSS-Rec$_{j,i}[\epsilon']$ protocols for $P_j \in C_i$) only *after* it completes the `(Attach` A_i `to` P_i`)` broadcast. Namely, the set C_i is fixed *before* any honest party invokes an AVSS-Rec$_{k,i}$ for some k. The Privacy property of AVSS now assures us that the bad parties have no information on the secrets shared by the honest parties when the set C_i is fixed. Thus, the set C_i, as well as the values that were shared by bad parties, are independent of the values shared by honest parties. Furthermore, each set C_i contains at least one honest party, and honest parties share uniformly distributed, mutually independent values. Consequently, the sum v_i is uniformly and independently distributed over $[1 \ldots u]$. □

Lemma 5.30 *Assume that some honest party has raised the* `'reconstruct enabled'` *flag. Then there exists a set, M, such that:*

1. *For each party $P_j \in M$, the* `(Attach` A_j `to` P_j`)` *broadcast of Step 2 has been completed by some honest party. (Note that Lemma 5.29 applies to each $P_j \in M$.)*
2. *When any honest party, P_j, raises his* `'reconstruct enabled'` *flag, it will hold that $M \subseteq H_j$.*
3. $|M| \geq \frac{n}{3}$.

Proof: Let P_i be the first honest party to raise his `'reconstruct enabled'` flag. Let M be the set of parties, P_k, for whom $P_k \in G_l$ for at least $t+1$ parties P_l who are considered supportive by P_i, in Step 4. We show that the set M has the properties required in the Lemma.

Clearly, $M \subseteq H_i$. Thus, party P_i has received the broadcast `(Attach` A_k `to` P_k`)` of every party $P_k \in M$. This asserts the first property of M. We now assert the second property. Let $P_k \in M$. An honest party P_j raises his `'reconstruct enabled'` flag, when he has found at least $n-t$ parties who are supportive in Step 4. However, $P_k \in G_l$ for at least $t+1$ of the `(`P_l `accepts` H_l`)` broadcasts; thus, there must exist a party P_l such that $P_k \in G_l$ and $G_l \subseteq H_j$. Consequently, $P_k \in H_j$.

It remains to show that $|M| \geq \frac{n}{3}$. We use a counting argument. Let $m \triangleq |H_i|$. We have $m \geq n-t$. Consider the $m \times n$ table T (relative to party P_i), where $T_{l,k} =$ `one` iff P_i has received the `(`P_l `accepts` H_l`)` broadcast and $P_k \in G_l$. The set M is the set of parties P_k such that the kth column in T has at least $t+1$ `one` entries. There are $(n-t)$ `one` entries in each row of T; thus, there are $m \cdot (n-t)$ `one` entries in T.

Let q denote the minimum number of columns in T that contain at least $t+1$ `one` entries. We show that $q \geq \frac{n}{3}$. Clearly, the worst distribution of the `one` entries in this table is letting q columns be all `one`'s (namely, each of the q columns has m `one` entries), and letting each of the remaining $(n-q)$ columns have t `one` entries. This distribution requires that the number of `one` entries be no more than $q \cdot m + (n-q) \cdot t$. However, there are $m \cdot (n-t)$ `one` entries in T. Thus, we must have:

$$q \cdot m + (n-q) \cdot t \geq m \cdot (n-t)$$

or, equivalently, $q \geq \frac{m(n-t)-nt}{m-t}$. Since $m \geq n-t$ and $n \geq 3t+1$, we have

$$q \geq \frac{m(n-t)-nt}{m-t} \geq \frac{(n-t)^2-nt}{n-2t} \geq \frac{(n-2t)^2+nt-3t^2}{n-2t} \geq n-2t+\frac{nt-3t^2}{n-2t} \geq n-2t+\frac{t}{n-2t} > \frac{n}{3}.$$

□

Lemma 5.31 *Let $\epsilon \leq 0.2$, and assume that all the honest parties have completed protocol* Common-Coin[ϵ]. *Then, for every value $\sigma \in \{0,1\}$, with probability at least* 0.25*, all the honest parties output σ.*

Proof: By Lemma 5.29 we have that for every party, P_j, who is accepted by some honest party, there exists a value, v_j, distributed uniformly and independently over $[1 \ldots u]$, such that with probability $1-\frac{\epsilon}{n}$ all the honest parties associate v_j with P_j in Step 6. Consequently, with probability $1-\epsilon$, all the honest parties agree on the value associated with each one of the parties.)

Consider the case $\sigma = 0$. Let M be the set of parties guaranteed by Lemma 5.30. Clearly, if $v_j = 0$ for some party $P_j \in M$ and all the honest parties associate v_j with P_j, then all the honest parties output 0. The probability that at least one party $P_j \in M$ has $v_j = 0$ is $1-(1-\frac{1}{u})^{|M|}$. Recall that $u = \lceil 0.87n \rceil$, and that $|M| \geq \frac{n}{3}$. Therefore, for all $n > 4$ we have $1-(1-\frac{1}{u})^{|M|} \geq 1-\mathrm{e}^{-0.38} \geq 0.316$. Thus, Prob(all the honest parties output 0) $\geq 0.316 \cdot (1-\epsilon) \geq 0.25$.

Consider the case $\sigma = 1$. Clearly, if no party P_j has $v_j = 0$ (and all the honest parties associate v_j with every P_j), then all the honest parties output 1. The probability of this event is at least $(1-\frac{1}{u})^n(1-\epsilon) \geq \mathrm{e}^{-1.15} \cdot 0.8 \geq 0.25$. Thus, Prob(all the honest parties output 1) ≥ 0.25. □

5.8 Byzantine Agreement

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

5.8.1 The Voting Protocol

In protocol Vote each party tries to find out whether there is a detectable majority for some value among the (binary) inputs of the parties. More precisely, each party's output of the protocol can take five different values. For $\sigma \in \{0,1\}$, the output $(\sigma, 2)$ stands for 'overwhelming majority for σ'. Output $(\sigma, 1)$ stands for 'distinct majority for σ'. Output $(\Lambda, 0)$ stands for 'non-distinct majority'. It will be shown that if all the honest parties have the same input, σ, then all honest parties output $(\sigma, 2)$. Furthermore, if some honest party outputs $(\sigma, 2)$ then every honest party will output either $(\sigma, 2)$ or $(\sigma, 1)$. If some honest party outputs $(\sigma, 1)$ then either all outputs are in $\{(\sigma, 2), (\sigma, 1)\}$, or all outputs are in $\{(\Lambda, 0), (\sigma, 1)\}$.

The protocol consists of three 'rounds', having similar structure. In the first round, each party broadcasts his input value, waits to complete $n-t$ broadcasts of other parties, and sets his vote to the majority value among these inputs. In the second round, each party broadcasts his vote (along with the identities of the $n-t$ parties whose broadcasted inputs were used to compute the vote), waits to complete $n-t$ broadcasts of other votes that are consistent with the broadcasted inputs of the first round, and sets his re-vote to the majority value among these votes. In the third round each party broadcasts his re-vote, along with the identities of the $n-t$ parties whose broadcasted votes were used to compute the re-vote, and waits to complete $n-t$ broadcasts of other re-votes that are consistent with the consistent votes of the second round.

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

Protocol Vote is presented in Figure 5-10 below.

Protocol Vote(x_i)

Code for party P_i, on input x_i:

1. Broadcast $(\texttt{Input}, P_i, x_i)$.
2. Define a dynamic set $\mathcal{A}_i$. Add (P_j, x_j) to $\mathcal{A}_i$ if the $(\texttt{Input}, P_j, x_j)$ broadcast is completed.

   Wait until $|\mathcal{A}_i| \geq n-t$. Set A_i to $\mathcal{A}_i^{(n-t)}$. Then, set v_i to the majority bit among $\{x_j | (P_j, x_j) \in A_i\}$, and broadcast $(\texttt{Vote}, P_i, A_i, v_i)$.
3. Define a dynamic set $\mathcal{B}_i$. Add (P_j, A_j, v_j) to $\mathcal{B}_i$ if the $(\texttt{Vote}, P_j, A_j, v_j)$ broadcast has been completed, $a_j \subseteq \mathcal{A}_i$, and v_j is the majority bit of A_j.

   Wait until $|\mathcal{B}_i| \geq n-t$. Set B_i to $\mathcal{B}_i^{(n-t)}$. Then, set rv_i to the majority bit among $\{v_j \ | (P_j, \mathsf{vote}_j, A_j) \in B_i\}$, and broadcast $(\texttt{Re-vote}, P_i, B_i, rv_i)$.
4. Define a set C_i. Add (P_j, B_j, rv_j) to C_i if the $(\texttt{Re-vote}, P_j, B_j, rv_j)$ broadcast has been completed, $B_j \subseteq \mathcal{B}_i$, and rv_j is the majority bit of B_j.
5. Wait until $|C_i| \geq n-t$.
   If all the parties $P_j \in B_i$ had the same vote $v_j = \sigma$, then output $(\sigma, 2)$ and terminate.
   Otherwise, if all the parties $P_j \in C_i$ have the same re-vote, $rv_j = \sigma$, then output $(\sigma, 1)$ and terminate.

   Otherwise, output $(\Lambda, 0)$ and terminate.

Figure 5-10: The Vote protocol

In Lemmas 5.32 through 5.35 we assert the properties of protocol Vote, as describe above. The Lemmas hold for every input and every t-adversary.

Lemma 5.32 *All the honest parties terminate protocol Vote in constant time.*

Proof: The $(\texttt{Input}, P_j, x_j)$ broadcast of every honest party P_j in Step 1 will be completed. Thus, every honest party P_i will eventually have $|\mathcal{A}_i| = n-t$, in Step 2, and will broadcast $(\texttt{Vote}, P_i, A_i, v_i)$. For every honest party P_j the broadcast of Step 2 will be completed. An honest party P_i will add (P_j, A_j, v_j) to $\mathcal{B}_i$ for each honest P_j, in Step 3. Thus, every honest party P_i will eventually have $|\mathcal{B}_i| = n-t$, in Step 3, and will broadcast $(\texttt{Re-vote}, P_i, B_i, rv_i)$. Similarly, P_i will add every honest party P_j to C_i. Thus, every honest party P_i will eventually have $|C_i| = n-t$, in Step 5. Consequently, P_i will complete the protocol. We note that the protocol runs in constant time. □

Lemma 5.33 *If all the honest parties have input σ, then all the honest parties output $(\sigma, 2)$.*

Proof: Consider an honest party P_i. If all the honest parties have input σ, then at most t parties will broadcast $\bar{\sigma}$ as their input in Step 1. Therefore, each party P_k who was added to $\mathcal{B}_i$ in Step 3 has a majority for the value σ in his A_k set, and $v_k = \sigma$. Thus, P_i outputs $(\sigma, 2)$ in Step 5. □

Lemma 5.34 *If some honest party outputs $(\sigma, 2)$, then each honest party outputs either $(\sigma, 2)$ or $(\sigma, 1)$.*

Proof: Assume that honest party P_i outputs $(\sigma, 2)$. The size of B_i is $n - t$, hence for each other honest party P_j it holds that $B_i \cap B_j$ is of size at least $t + 1$. Thus, P_j will set his revote rv_j to σ. Therefore, every honest party outputs either $(\sigma, 2)$ or $(\sigma, 1)$ in Step 5. □

Lemma 5.35 *If some honest party outputs $(\sigma, 1)$, and no honest party outputs $(\sigma, 2)$, then each honest party outputs either $(\sigma, 1)$, or $(\Lambda, 0)$.*

Proof: Assume some honest party outputs $(\sigma, 1)$. Then, at most t parties P_j broadcasted a revote $rv_j = \bar{\sigma}$ in Step 3; therefore, no honest party P_j has a unanimous $\textsf{re-vote}_j = \bar{\sigma}$ in Step 4, and no honest party outputs $(\bar{\sigma}, 1)$. Furthermore, at least $t + 1$ parties P_k have broadcasted $\textsf{vote}_k = \sigma$ in Step 2; therefore, no honest party had a unanimous $\textsf{vote}$ in Step 3, and no honest party outputs $(\bar{\sigma}, 2)$. □

5.8.2 The Byzantine Agreement protocol

The Byzantine Agreement protocol proceeds in iterations. In each iteration, each party has a 'modified input' value; in the first iteration, the modified input of each party is his local input. In each iteration the parties invoke two protocols: Vote and Common-Coin. Protocol Common-Coin is invoked only *after* protocol Vote is completed. (The reason for this provision will become clear in the proof of Lemma 5.38 below.) If a party 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 Common-Coin protocol. (Protocol Common-Coin is invoked by all parties in each iteration, regardless of whether their output is used.) Once a party recognizes an 'overwhelming majority' for some value, σ, (namely, output $(\sigma, 2)$ of protocol Vote), he broadcasts σ. Once a party completes $t + 1$ broadcasts for some value, σ, he terminates with output σ.

The code of the Byzantine Agreement protocol is presented in Figure 5-11 below.

In Lemmas 5.36 through 5.40 we assert the validity of protocol BA. These Lemmas hold for every input and every t-adversary.

Lemma 5.36 *If all the honest parties have input σ, then all the honest parties terminate and output σ.*

Proof: Assume that all the honest parties have input σ. By Lemma 5.33, every honest party P_i has $(y_1, m_1) = (\sigma, 2)$ by the end of Step 1 of the first iteration. Therefore, every honest party broadcasts `(Terminate with` σ`)` in Step 2 of the first iteration. Therefore, every honest party will receive at least $n - t$ `(Terminate with` σ`)` broadcasts, and at most t `(Terminate with` $\bar{\sigma}$`)` broadcasts. Consequently, every honest party will output σ. □

Protocol BA$[\epsilon](x_i)$

Code for party P_i, on input x_i, and termination parameter ϵ:

1. Set $r := 0$. Set $v_1 := x_i$.

Repeat until terminating:

2. Set $r := r + 1$. Set (y_r, m_r) :=Vote(v_r).
3. Wait until Vote(v_r) is completed. Then, set c_r :=Common-Coin$[\frac{\epsilon}{4}]$.
4. (a) If $m_r = 2$, set $v_r := y_r$ and broadcast `(Terminate with` v_r`)` . Participate in only one more Vote protocol and only one more Common-Coin protocol. [a]
   (b) If $m_r = 1$, set $v_{r+1} := y_r$.
   (c) Otherwise, set $v_{r+1} := c_r$.

- Upon receiving $t + 1$ `(Terminate with` σ`)` broadcasts for some value σ, output σ and terminate.

Figure 5-11: The Byzantine Agreement protocol

[a] The purpose of this restriction is to prevent the parties from participating in an unbounded number of iterations before enough `(Terminate with` σ`)` broadcasts are completed.

Lemma 5.37 *If an honest party terminates with output σ, then all honest parties terminate with output σ.*

Proof: Let us first establish that if an honest party broadcasts `(Terminate with` σ`)` for some value σ, then all honest parties eventually broadcast `(Terminate with` σ`)` . Let k be the first iteration in which an honest party initiated a `(Terminate with` σ`)` broadcast for some value σ. By Lemma 5.34, every honest party P_i has $y_k = \sigma$ and either $m_k = 2$ or $m_k = 1$. Therefore, no honest party broadcasts `(Terminate with` $\bar{\sigma}$`)` at iteration k. Furthermore, all the honest parties execute the Vote protocol of iteration $k+1$ with input σ. Lemma 5.33 now implies that by the end of Step 1 of iteration $k+1$, every honest party has $(y_{k+1}, m_{k+1}) = (\sigma, 2)$. Thus, all the honest parties broadcast `(Terminate with` σ`)` , either at iteration k or at iteration $k+1$.

Now, assume an honest party terminates with output σ. Thus, at least one honest party broadcasted `(Terminate with` σ`)` . Consequently, all the honest parties broadcast `(Terminate with` σ`)` . Hence, every honest party will receive at least $n - t$ `(Terminate with` σ`)` broadcasts, and at most t `(Terminate with` $\bar{\sigma}$`)` broadcasts. Therefore, every honest party will output σ. □

Lemma 5.38 *Assume all honest parties have initiated and completed some round k. Then, with probability at least $\frac{1}{4}$ all honest parties have the same value for v_{k+1}.*

Proof: We distinguish two cases. If all the honest parties execute Step 4(c) in iteration k, then all the honest parties set their v_{k+1} value to their (local) output of protocol Common-Coin. In this case all the parties have the same v_{k+1} value with probability at least $\frac{1}{2}$.

Otherwise, some honest party has set $v_{k+1} = \sigma$ for some $\sigma \in \{0,1\}$, either in Step 4(a) or Step 4(b) of iteration k. By Lemma 5.35, no honest party will use either Step 4(a) or Step 4(b) to set his v_{k+1} variable to $\bar{\sigma}$. Furthermore, with probability at least $\frac{1}{4}$, all the honest parties have output σ of the Common-Coin protocol of Step 3. Therefore, with probability at least $\frac{1}{4}$, all the honest parties have $v_{k+1} = \sigma$ at the end of iteration k. (Note that the parties' outputs of protocol Vote are fixed *before* Common-Coin is invoked. Were it not the case, the bad parties could force the output of protocol Vote to prevent agreement.) □

Let C_k denote the event that each honest party completes all the iterations he initiated, up to (and including) the kth iteration (that is, for each iteration $1 \le l \le k$ and for each party P, if P initiated iteration l then he computes v_{l+1}). Let C denote the event that C_k occurs for all k.

Lemma 5.39 *Conditioned on event C, all the honest parties complete protocol BA in constant expected time.*

Proof: We first establish that all the honest parties terminate protocol BA within constant time after the first honest party initiates a `(Terminate with` σ`)` broadcast in Step 4(a) of the protocol. Assume the first honest party initiates a `(Terminate with` σ`)` broadcast in iteration k. Then, all the honest parties participate in the Vote and Common-Coin protocols of all the iterations up to iteration $k+1$. We have seen in the proof of Lemma 5.37 that in this case, all the honest parties initiate a `(Terminate with` σ`)` broadcast by the end of iteration $k+1$. All these broadcasts terminate in constant time. Each honest party terminates upon completing $t+1$ of these broadcasts.

Let the random variable τ count the number of iterations until the first honest party broadcasts `(Terminate with` σ`)` . (If no honest party broadcasts `(Terminate with` σ`)` then $\tau = \infty$). Conditioned on event C, all the honest parties terminate each iteration in constant time. It is left to show that $\mathrm{E}(\tau|C)$ is constant. We have

$$\mathrm{Prob}(\tau > k|C_k) \le \mathrm{Prob}(\tau \ne 1|C_k) \cdot \ldots \cdot \mathrm{Prob}(\tau \ne k|C_k \cap \tau \ne 1 \cap \ldots \cap \tau \ne k-1)$$

If follows from Lemma 5.38 that each one of the k multiplicands of the right hand side of the above equation is at most $\frac{3}{4}$. Thus, $\mathrm{Prob}(\tau > k|C_k) \le \left(\frac{3}{4}\right)^k$. It follows, via a simple calculation, that $\mathrm{E}(\tau|C) \le 16$. □

Lemma 5.40 $\mathrm{Prob}(C) \ge 1 - \epsilon$.

Proof: We have

$$\begin{aligned}
\mathrm{Prob}(\overline{C}) &\le \sum_{k\ge 1} \mathrm{Prob}(\tau > k \cap \overline{C_{k+1}}|C_k) && (5.1)\\
&\le \sum_{k\ge 1} \mathrm{Prob}(\tau > k|C_k) \cdot \mathrm{Prob}(\overline{C_{k+1}}|C_k \cap \tau > k) && (5.2)
\end{aligned}$$

We have seen in the proof of Lemma 5.39 that $\mathrm{Prob}(\tau > k|C_k) \le \left(\frac{3}{4}\right)^{k-1}$. We bound the term $\mathrm{Prob}(\overline{C_{k+1}}|C_k \cap \tau \ge k)$. If all the honest parties execute the kth iteration and complete the kth invocation of Common-Coin, then all the honest parties complete the kth iteration. Protocol Common-Coin is invoked with 'termination parameter' $\frac{\epsilon}{4}$. Thus, with probability

$1-\frac{\epsilon}{4}$, all the honest parties complete the kth invocation of Common-Coin. Therefore, for each k, $\text{Prob}(\overline{C_{k+1}}|C_k \cap \tau \geq k) \leq \frac{\epsilon}{4}$. Inequality 5.1 yields $\text{Prob}(\overline{C}) \leq \sum_{k\geq 1} \frac{\epsilon}{4}\left(\frac{3}{4}\right)^{k-1} = \epsilon$. □

We have thus shown:

Theorem 5.2 (Byzantine Agreement.) *Let* $n \geq 3t+1$. *Then, for every* $0 < \epsilon \leq 0.2$, *protocol* BA[ϵ] *is a* $(1-\epsilon)$*-terminating,* t*-resilient, asynchronous Byzantine Agreement protocol for* n *parties. Given that the parties terminate, they do so in constant expected time. Furthermore, the computational resources required of each party are polynomial in* n *and* $\log \frac{1}{\epsilon}$.