Byzantine Agreement with Incomplete Views

Hanjun Li

November 20, 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Vipul Goyal
Bryan Parno

Submitted in partial fulfillment of the requirements
for the degree of Master of Science.

Keywords: Byzantine Agreement

For my dog

Abstract

A short summary

Acknowledgments

My advisor, and my collaborator

Contents

Chapter 1

Introduction

1.1 Related Works

Introduced; deterministic; 1/3 [LSP], [PSL80], [FL82], [GM98],

Authenticated; 1/2 [LSP], [PSL80], [DS83],

Randomized; 1/3 and 1/2 [BO83], [Rab83], [FM97], [KK06], [Mic17],

Imcomplete Network [KS09], [CGO10], [Dol82],

1.2 Contribution

Chapter 2

Preliminaries

2.1 Notations

We use $\{0,1\}^*$ to denote the set of finite binary strings. When describing a probabilistic algorithm $F(\cdot)$, $F(x)$ refers to the probability space that assigns any string y the probability that F on input x outputs y. We write $y \xleftarrow{R} F(x)$ to describe assigning to y an element randomly selected according to $F(x)$. In contrast, for a deterministic algorithm $G(\cdot)$, we simply write $y = G(x)$ to describe the output of G on input x being y.

We will use the usual notation $G = (V, E)$ to represent the communication graph among the participants. V is the set of nodes in the graph. Each node P_i represents the participant P_i. E is the set of edges. Each edge (P_j, P_k) represents the fact that P_k is in the *view* of P_j (see Section 2.2). Our model assumes only bi-directional edges. That is, if P_k is in the view of P_j then P_j is also in the view of P_k. We write Γ_i as the inclusive neighbor of P_i in G (i.e. all participants in the view of P_i including itself).

2.2 Unique Digital Signature and PKI

Similar to other authenticaed Byzantine Agreement protocols, we will use a Digital Signature scheme to ensure that a corrupted participant can either choose to forward or ignore a signed message, but can never forge a signed message. In addition, we also require that for any public key and any message m, the there is a unique valid signature. The additional uniqueness constraint is introduced in [GO92], and a construction based on the RSA assumption is provided in [MRVil]. Briefly, we summarize the notion of Unique Digital Signature below.

Notation: A Unique Digital Signature scheme is a triple of polynomial time computatble algorithms $(Gen, Sign, Verify)$ as described below.

- $Gen(\cdot)$ is a probabilistic algorithm that takes a unary string of length k, the security parameter, as input, and outputs two binary strings, a public key P_k and a secret key S_k. We write this as $(S_k, P_k) \xleftarrow{R} Gen(1^k)$.

- $Sign(\cdot,\cdot)$ is a deterministic algorithm that takes in the secret key S_k and a message x, and produces a signature $Sig_{P_k}(x)$.
  We write this as $Sig = Sign(S_k, x)$.
- $Verify(\cdot,\cdot,\cdot)$ is a probabilisitc algorithm that takes a public key P_k, a message x and a signature Sig as input, and outputs a bit $b \in \{0,1\}$.
  We write this as $b \xleftarrow{R} Verify(P_k, x, Sig)$.

We now briefly describe the correctness and the security of a Unique Digital Signature scheme.

Correctness:

- $Verify$ accepts (i.e. outputs 1) a valid signature produced by $Sign$ with overwhelming probabiblity over $(P_k, S_k) \xleftarrow{R} Gen(1^k)$
- There do not exist values (P_k, x, Sig_1, Sig_2) such that $Sig_1 \neq Sig_2$, and $Verify(P_k, x, Sig_1) = Verify(P_k, x, Sig_2) = 1$.

Security: Let T be any polynomial time algorithm. The probability that T wins the following game must be negligiable:

- Run $(P_k, S_k) \xleftarrow{R} Gen(1^k)$:
- Run $(x, Sig) \xleftarrow{R} T^{Sign(S_k,\cdot)(1^k, P_k)}$
- T wins if $1 \xleftarrow{R} Verify(P_k, x, Sig)$

This security definition is also called exsistentially unforgable.

Public-key Infrastructure: For a Digital Signature scheme to be useful for authenticating messages, a trusted setup phase is required to first assign each participant P_i its key pair (P_{k_i}, S_{k_i}), and next distribute public keys of the participants. This setup phase is generally refered to as a Public-key infrastruture (PKI) setup. In our relaxed model, the setup phase only assigns to each participants a subset of the public keys. If the public key of P_j is assigned to P_i, we say P_i *trusts* P_j. The set of *trusted* participants by P_i is called the *view* of P_i. In the later sections, we will use the player id to identity its public key. For example a signature by P_i on some message m_i will be writen as $Sig_i(m_i)$.

2.3 Verifiable Random Function

When describing and analyzing our protocols, we will assume that every participant has access to a public random function H, mapping $\{0,1\}^*$ to $\{0,1\}^k$ for any k. In this idealized model, when H receives a query string x, it selects a k bit string uniformly at random as its output $H(x)$. All further query of x all result in the same output $H(x)$.

Note that the random oracle H is only introduced to simplify our analysis. The usage of H can be replaced by a verifiable random function (VRF) scheme which can be constructed under the RSA assumption [MRVil]. We now briefly summarize the notion of VRF below.

Notation: A Verifiable Random Function scheme is a triple of polynomial time computable algorithms (G, F, V) as described below.

- $G(\cdot)$ is a probabilistic algorithm that takes a unary string of length k, the security parameter, as input, and outputs two binary strings, a public key PK and a secret key SK.
  We write this as $(PK, SK) \xleftarrow{R} G(1^k)$
- $F(\cdot, \cdot)$ is a deterministic algorithm that takes two binary strings, the secrete key SK and a seed x, and outputs two binary strings, the value v and its corresponding proof $proof$.
  We write this as $(v, proof) = F(SK, x)$. For convenience, we sometimes write $F = (F_1, F_2)$ where $v = F_1(SK, x)$ and $proof = F_2(SK, x)$.
- $V(\cdot, \cdot, \cdot, \cdot)$ is a probabilistic algorithm that takes four binary strings, the public key PK, the seed x, the value v, and the proof $proof$, as input, and output a bit $b \in \{1, 0\}$.
  We write this as $b \xleftarrow{R} V(PK, x, v, proof)$

At a high level, we can think of G as the function generator, F as the function evaluator, and V as the function verifier. We now describe the correctness and the security of a VRF scheme.

Correctness: The following must hold with overwhelming probability over $(PK, SK) \xleftarrow{R} G(1^k)$:

- For all x in its domain, $F_1(SK, x)$ produces a string in its correct range.
- For all x in its domain, if $(v, proof) = F(SK, v)$, then $1 \xleftarrow{R} V(PK, x, v, proof)$.
- For every $x, v_1, v_2, proof_1, proof_2$ such that $v_1 \neq v_2$, either $0 \xleftarrow{R} V(PK, x, v_1, proof_1)$ or $0 \xleftarrow{R} V(PK, x, v_2, proof_2)$.

At a high level, the correctness requires that the proof produced by F_2 can be uniquely verified by V.

Security: Let $T = (T_E, T_j)$ be any pair of polynomial time algorithm (in the security parameter k). The advantage that T has in succeeding the following game over $1/2$ (i.e. random guessing) must be negligiable.

- Run $(PK, SK) \xleftarrow{R} G(1^k)$
- Run $(x, state) \xleftarrow{R} T_E^{F(SK,\cdot)}(1^k, PK)$
- Randomly choose a bit $r \xleftarrow{R} \{0, 1\}$:
  - if $r = 0$, run $v = F_1(SK, x)$
  - if $r = 1$, sampe v at random from the range of F_1.
- Run $b \xleftarrow{R} T_J^{F(SK,\cdot)}(1^k, v, state)$ where b is the guess of T. T succeeds if x is in the domain of F_1, x is not asked as a query to $F(SK, \cdot)$ by T_E or T_J, and $b = r$.

At a high level, the security requires that the output of F_1 is indistinguishable from a random string.

Using VRF in place of $H(\cdot)$: We now describe how to use a VRF scheme to simulate the random oracle H used in our protocol. In the trusted PKI setup phase, each participant P_i is assigned its own key pair (PK_i, SK_i), and also the set of public keys in its view. In our protocol, whenever a participant P_i receives a value v_j originated from P_j, and needs to evaluate $H(v_j)$, we instead ask P_j to attach the evaluation $(r_j, proof_j) = F(SK_j, v_j)$ and its public key PK_j to v_j. That is, P_i will actually receive $v'_j = (v_j, r_j, proof_j, PK_j)$, and can simply run $V(PK_j, v_j, r_j, proof_j)$ to verify that r_j is the correct result. Each message is additionally signed by P_j so that no one can forge an evaluation of $H(v_j)$.

In the case where P_j is not trusted by P_i, (hence all messages of v_j is forwared by some other participant) our protocol always guarantees that at least one forwarding is from an honest participant, who has verified its signature and the included public key PK_i. Whenever P_j receives two contradicting PK_i and PK'_i, P_j discards all messages from P_i since it must have been corrupted.

2.4 Our Model and Main Results

Our model is a relaxiation from the standard one assumed in most authenticated Byzantine Agreement protocols (e.g. [Mic17], [KK06]). Details are provided below.

Communication: All participants communicate in synchronized rounds, over authenticated and private point-to-point channels. Note that such authenticated channels only exists between two parties who trusts each other, so our communication network is not a *complete* network. Messages are sent as the start of a round, and received by the end of the round.

The Adversary: The adversary is a polynomial time algorithm, that can adaptively corrupt honest participants during the protocol. Corrupted participants can deviate from the protocol in arbitrary ways. At the start of any round, the adversay may corrupt additional players before receiving messages from all honest participants, and then decide what to send from all corrupted participants. The adversary knows the public keys from all participants at the beginning.

Results: We first state the standard definition of a Byzantine Agreement protocol. For simplicity, we only consider the *binary* version.

Definition 1. *(Byzantine Agreement) For a set of participants $P_1, ..., P_N$, where each P_i holds an initial input $v_i \in \{0, 1\}$, when the protocol terminates, the following conditions must hold for any adversary:*

- *(Validity) If all honest participants begin with the same input v, they also output v.*
- *(Agreement) All honest participants output the same value*

Let α_i be the fraction of corrupted participants in the view of an honest participant P_i. We define $\alpha \equiv \max_i\{\alpha_i\}$. Let δ_{ij} be the fraction of overlapping between the views of two honest participants P_i, P_j (i.e. $\delta_{ij} = |\Gamma_i \cap \Gamma_j|/|\Gamma_i|$). We similarly define $\delta \equiv \min_{i,j}\{\delta_{ij}\}$. Now we are ready to state our results as the following theorems:

Theorem 1. *If $\delta > 2\alpha$, there exists an expected O(n) round Byzantine Agreement protocol. Further if $\alpha = 1/2 - \epsilon$ for any constant ϵ, there exists an expected constant round Byzantine Agreement protocol in our model.*

Theorem 2. *If $\alpha \geq 1/2$ or $\delta \leq 2\alpha$, there does not exist a Byzantine Agreement protocol in our model*

Chapter 3

Positive Result

This section first introduces several subprotocols as building blocks, and in the end use them in our main protocol. For simplicity, in the following discussion we will assume that all honest participants have a view of fixed size n. However, we note that all of our results hold without this restriction. Intuitively, since every pair of honest participants has at least a δ overlapping in their views, the size of their views can not differ by too much.

3.1 Graded Broadcast

A Graded Broadcast protocol is a similar but weaker notion to broadcast. At a high level, it simulates a broadcast with some participants fail to receive the message. It, however, guarantees that all participants that sccessfully receive a message indeed receive the same message. In addition, if the sender is an honest participant, the broadcast always succeeds. The name "Graded Broadcast" comes from the way a participant decide whether to accept a message: the successful message is assigned a positive grade, while a failed message is assigned grade 0. This idea is first introduced in [FM97] in the standard model. We now give a formal definition in our model.

Definition 2. *(Graded Broadcast) For a set of participants $S = \{P_1, ..., P_N\}$, and a distinguished dealer $P_d \in S$ holding an initial message m, when the protocol terminates, the following conditions must hold for any adversary:*

- *Each honest participant P_i in the view of P_d (i.e. $P_i \in \Gamma_d$) outputs (m_i, g_i), where $g_i \in \{0, 1\}$.*
- *(Validity) If P_d is honest, then $m_i = m$, and $g_i = 1$ for all $P_i \in \Gamma_d$.*
- *(Agreement) If two honest participants $P_i, P_j \in \Gamma_d$ outputs (m_i, 1), and $(m_j, 1)$, then $m_i = m_j$.*

Next we give our protocol (Algorithm 1) that achieves Graded Broadcast assuming $\delta \geq 2\alpha$.

Algorithm 1 Graded Broadcast

1: The dealer P_d signs message m, and sends $(m, Sig_d(m))$ to all participants in its view.
2: Every honest participant P_i in the view of P_d receives $(m, Sig_d(m))$, and verifies the signature.
   - if the signature is valid, P_i forwards it to all participants in the view of P_i.
   - else P_i does nothing.
3: Every honest participant P_i in the view of P_d receives forwarded messages $m_j, Sig_d(m_j)$ from some P_j in its own view.
   - If there are contradicting valid signatures of $m_j \neq m_k$ for some j, k, then P_i outputs $(\phi, 0)$.
   - If there are less than $(\delta - \alpha)n$ number of valid signatures on some message m, then outputs $(\phi, 0)$.
   - Else, outputs $(m, 1)$.

We now show the following claims about Algorithm 1.

Claim 3. *(Validity) If the dealer P_d is honest with message m, then all honest participants P_i in the view of P_d output $(m, 1)$ at the end of the Graded Broadcast protocol in Algorithm 1.*

Proof. Since P_d is honest, only signatures of the message m is ever sent out by P_d. By the security of a Digital Signature scheme, no contradicting signature (on a different message $m' \neq m$) can be forged. Therefore, the first "If" branch in **Step 3** will not happen.
By our model assumption, there are at least $(1 - \alpha)n$ honest participants in the view of P_d. For any honest P_i, at least $((1 - \alpha) - (1 - \delta))n = (\delta - \alpha)n$ of them are also in the view of P_i. Therefore, any honest participant P_i in the view of P_d will receive at least $(\delta - \alpha)n$ forwarded messages with valid signatures on m. Therefore, the second "If" branch in **Step 3** will also not happen, and P_i outputs $(m, 1)$. □

Claim 4. *(Agreement) If two honest participants P_i, P_j in the view of P_d outpus $(m_i, 1)$ and $(m_j, 1)$, and if $\delta > 2\alpha$, then $m_i = m_j$.*

Proof. Suppose $m_i \neq m_j$. In **Step 3**, P_i has received at least $(\delta - \alpha)n$ valid signatures on m_i. By assumption, $(\delta - \alpha)n > \alpha n$, so at least one of them is forwarded by some honest participant P_k in the view of P_d.
Since P_k is honest, in **Step 2**, P_k must have also sent $(m_i, Sig_d(m_i))$ to P_j, and P_j must have seen contradicting valid signatures of $m_i \neq m_j$ in **Step 3**. This leads to a contradiction. □

The above claims lead to the following lemma that we will use as a building block to prove Theorem 1.

Lemma 5. *If $\delta > 2\alpha$, there exists a three round Graded Broadcast protocol.*

Proof. By Algorithm 1, Claim 3, and Claim 4. □

3.2 Oblivious Leader Selection

An Oblivious Leader Selection protocol is used for electing an honest leader that is agreed on by all honest participants. However, we only need this to happen with *some* probability, to which we refer as the *fairness* of the protocol. If the fairness is a constant, then running the protocol repeatedly will give us an honest leader in expected constant round. If the fairness is $\Omega(1/n)$, then running the protocol repeatedly will give us an honest leader in expected $O(n)$ round. Our definition is similar to the one present in [KK06]. Our protocol construction is inspired by the *ConcreteCoin* protocol present in [Mic17]. We now give a formal definition in our model:

Definition 3. *(Oblivious Leader Selection) For a set of participants $P_1, ..., P_N$ and fairness γ, when the protocol terminates, the following conditions must hold with probability at least γ:*

- *Every honest participant P_i outputs P_l and P_l is honest by the end of the protocol.*

When such an event happens, we say that an honest leader P_l is elected.

Next, we give our protocol (Algorithm 2) that achieves Oblivious Leader Selection assuming $\delta \geq 2\alpha$. Its fainess is analyzed below.

Algorithm 2 Oblivious Leader Selection

Input: r

1: Let r be given (representing the current iteration number in the outter protocol). Every honest participant P_i sends $m_i = (i, Sig_i(r))$ to all other participants in its view.
2: Every honest participant P_i forwards messages with valid signatures to all other participants in the view of P_i.
3: Every honest participant P_i receives at most n forwarded messages from each P_j in its view (and ignore the messages after the first n). P_i computes a set S_i of messages that are forwarded by at least $(\delta - \alpha)n$ participants in its view, and send S_i to all participants in its view.
4: Every honest participant P_i receives a set S_j from every participant P_j in its view.
   - P_i computes a set S_i^* of messages that appear in at least $(1 - \alpha)n$ received sets.
   - For every $m_k \in S_i^*$, P_i computes $H(m_k)$ and outputs P_l where l is the smallest id such that for all $m_k \in S_i^*$ $H(m_l) \leq H(m_k)$.

Output: P_l

We now show the following claims about Algorithm 2.

Claim 6. *In any iteration r, if P_i, P_j are any two honest participants, then $m_j = (j, Sig_j(r)) \in S_i^*$ in **Step 4**.*

Proof. Consider any honest participant P_k. By our model assumption, there are at least $(1-\alpha)n$ honest participants in the view of P_j, and at least $((1 - \alpha) - (1 - \delta))n = (\delta - \alpha)n$ of them are also in the view of P_k. Therefore, in **Step 3** P_k receives m_j forwarded by least $(\delta - \alpha)n$ honest participants. Hence $m_j \in S_k$.
By the argument above, every honest P_k in the view of P_i sends S_k with $m_j \in S_k$ in **Step 3**. Since there are at least $(1 - \alpha)n$ of them, we have $m_j \in S_i^*$ in **Step 4**. □

Claim 7. *In any iteration r, if P_i is honest, and if $\delta > 2\alpha$, then S_i contains at most $2n$ messages from corrupted participants.*

Proof. We first calculate the total number of times any message from a corrupted participants gets forwared to P_i. By our model assumption, there are at most αn corrupted participants in the view of P_i that can each forward n corrupted messages. The rest $(1-\alpha)n$ honest participants will each forward at most αn corrupted messages. In total, we get

$$\alpha n^2 + (1-\alpha)\alpha n^2 = \alpha(2-\alpha)n^2$$

For a corrupted message to be accepted into S_i, it must be forwarded at least $(\delta-\alpha)n$ times. Therefore, the number of accepted corrupted messages is at most

$$\frac{\alpha(2-\alpha)n^2}{(\delta-\alpha)n} \leq \frac{\alpha}{\delta-\alpha}2n < 2n$$

□

The above claims lead to the following lemma that we will use as a building block to prove Theorem 1.

Lemma 8. *If $\delta > 2\alpha$, there exists a three round Oblivious Leader Selection protocol with fairness $1/(5n)$. Further if $\alpha < 1/2 - \epsilon$ for some positive constant ϵ, then there exists a three round Oblivious Leader Selection protocol with constant fairness.*

Proof. We show that Algorithm 2 is such a protocol. Let C be the set of all honest participants. Consider the union of all honest S_i^*: $S = \cup_{P_i \in C} S_i^*$. If P_l is an honest participant and l is the smallest id such that for all $m_k \in S^*$ $H(m_l) \leq H(m_k)$, then by Claim 6, $P_l \in S_i^*$ for all honest P_i. Hence all honest P_i will output P_l in **Step 4** (i.e. an honest leader P_l is elected). Furthermore, by the definition of Random Oracle, $H(m_k)$ are uniform random and independent for all $m_k \in S^*$. Therefore, the probability that an honest leader P_l is elected is exactly $|C|/|S^*|$. We now calculate the total number corrupted messages in S^*. In **Step 4**, any corrupted message ever accepted into an honest S_i^* must appears in at least $(1-\alpha)n$ sets, of which at least $(1-2\alpha)n$ are from an honest participant. By Claim 7, the total number of corrupted messages in all honest S_i in **Step 3** is $2nC$. Therefore, the number of corrupted messages in S^* is at most

$$\frac{2n|C|}{(1-2\alpha)n} = \frac{2}{1-2\alpha}|C|$$

And we have $|S^*| \leq |C| + \frac{2}{1-2\alpha}|C|$. Since the number of corrupted participants in the view of any honest party is an integral number, we have $1/2 - \alpha \geq 1/(2n)$. The probability that an honest leader is elected is given by

$$\frac{|C|}{|S^*|} \geq \frac{|C|}{|C| + \frac{2}{1/(2n)}|C|} = \frac{1}{1+4n} \geq \frac{1}{5n}$$

In the case of $\alpha < 1/2 - \epsilon$ for some constant ϵ, we get

$$\frac{|C|}{|S^*|} \geq \frac{|C|}{|C| + \frac{2}{2\epsilon}|C|} = \frac{1}{1+1/\epsilon}$$

Which is a constant. □

3.3 Main protocol

The main protocol is inspired by the one present in [Mic17]. At a high level the protocol ensures that if one honest participant decides to terminate with some output $v \in \{0,1\}$, it is sure that no other honest participant terminates with a different $v' \neq v$ in the same round, and that all honest participants will be able to terminate in the next round. When all honest participants hold the same value, they terminates immediately. When an honest leader is elected, they terminates with probability $1/2$. Overall, if an honest leader is elected with constant probability, then our main protocol terminates with expectd constnat round. Similarly, if an honest leader is elected with probability $\Omega(1/n)$, then our main protocol terminates with expected $O(n)$ rounds. The main protocol is shown in Algorithm 3.

Algorithm 3 Byzantine Agreement

Input: $v_i \in \{0,1\}$: the initial value; $r \leftarrow 0$: the current iteration; $h_i \leftarrow 0$: whether to halt.

1: Every honest P_i runs a Graded Broadcast protocol as the dealer with message v_i. In the end, P_i outputs (v_j, g_j) for every participant P_j in its view, and accepts only values with grade 1.
   - If $h_i > 0$, do nothing.
   - If more than $(1 - 1/2\delta)n$ 0s are accepted, then set $v_i \leftarrow 0$ and $h_i \leftarrow 1$.
   - If more than $(1 - 1/2\delta)n$ 1s are accepted, then set $v_i \leftarrow 1$.
   - Otherwise, set $v_i \leftarrow 0$.

2: Every honest P_i runs a Graded Broadcast protocol as the dealer with message v_i. In the end, P_i outputs (v_j, g_j) for every participant P_j in its view, and accepts only values with grade 1.
   - If $h_i > 0$, do nothing.
   - If more than $(1 - 1/2\delta)n$ 1s are accepted, then set $v_i \leftarrow 1$ and $h_i \leftarrow 1$.
   - If more than $(1 - 1/2\delta)n$ 0s are accepted, then set $v_i \leftarrow 0$.
   - Otherwise, set $v_i \leftarrow 1$.

3: Every honest P_i sends a random bit $b \xleftarrow{R} \{0,1\}$ to every participants in its view. In the end, P_i receives b_j from every participant P_j in its view.

4: Every honest P_i runs an Oblivious Leader Selection protocol with input r and outputs P_{l_i}.

5: Every honest P_i runs a Graded Broadcast protocol as the dealer with message v_i. In the end, P_i outputs (v_j, g_j) for every participant P_j in its view, and accepts only values with grade 1.
   - If $h_i > 0$, do nothing.
   - If more than $(1 - 1/2\delta)n$ 1s are accepted, then set $v_i \leftarrow 1$.
   - If more than $(1 - 1/2\delta)n$ 0s are accepted, then set $v_i \leftarrow 0$.
   - If P_{l_i} is in the view of P_i, then set $v_i \leftarrow b_{l_i}$.

6: Set $r \leftarrow r + 1$.
   - If $h_i = 2$ halts with $v^* = v_i$.
   - If $h_i = 1$, sets $h_i \leftarrow 2$.

   Go back to **Step 1**.

Output: $v_i^* \in \{0,1\}$

We now show the following claims about Algorithm 3.

Claim 9. *(Validity) If every honest participant P_i has the same initial value v, and if $\delta > 2\alpha$, then they all terminate in the second iteration.*

Proof. By our model assumption and the correctness of Graded Broadcast, every honest participant P_i in **Step 1** will output $(v, 1)$ from at least $(1-\alpha)n$ honest Graded Broadcast. Since $\delta > 2\alpha$, $\vec{v}_i$ will contain more than $(1 - 1/2\delta)n$ values of v.
If $v = 0$, P_i sets $h_i \leftarrow 1$ in **Step 1** If $v = 1$, P_i keeps v_i unchanged and sets $h_i \leftarrow 1$ in **Step 2** by the same argument. Once $h_i = 1$, v_i never changes and P_i halts in the next iteration with $v^* = v_i$. □

Claim 10. *If some honest participant P_i sets $h_i \leftarrow 1$ in iteration r with $v_i = v$, then every other honest participant P_j will have set $h_j \leftarrow 1$ with $v_j = v$ by the end of iteration $r + 1$, and all of them halts by the end of iteration $r + 2$.*

Proof. It suffice to consider the first honest P_i that sets $h_i \leftarrow 1$.

- Suppose this happened in **Step 1**, consider a different honest participant P_j, who didn't set $h_j \leftarrow 1$ in **Step 1** (otherwise, we are done). By our model assumption, of any $(1 - 1/2\delta)n$ participants who Graded Broadcasted messages 0 to P_i, at least $((1 - 1/2\delta) - (1 - \delta))n = 1/2\delta n$ of them are also in the view of P_j. By correctness of Graded Broadcast, P_j cannot accept more than $(1 - 1/2\delta)n$ values of 1 in **Step 1**. Hence P_j sets $v_j \leftarrow 0$. That is, by the end of **Step 1**, every honest party P_j holds $v_j = 0$. Repeating the argument from Claim 9, P_j keeps v_j unchanged in this iteration. If P_j didn't set $h_j \leftarrow 1$ in **Step 1**, it will in the next iteration.
- Suppose this happened in **Step 2**. By assumption, P_i is the first honest party who sets $h_i \leftarrow 1$, and no other honest P_j have set $h_j \leftarrow 1$ in **Step 1**. The rest follows from exactly the argument in the previous case.

□

Note that by Claim 10, honest participants always halt with the same value. It now suffice to only consider the case when no honest participant P_i has set $h_i \leftarrow 1$:

Claim 11. *Suppose no honest participant has set $h_i \leftarrow 1$ in some iteration r. If an honest leader P_l is elected in iteration r, and if $\delta > 2\alpha$ then with probability $1/2$ P_l will set P_i in iteration $r + 1$.*

Proof. Suppose some honest P_i in the view of P_l in **Step 5** accepted more than $(1 - 1/2\delta)n$ values of some value v. By the argument from Claim 10, no other honest P_j have accepted more than $(1 - 1/2\delta)n$ values of the diffferent value v^c. That is, in **Step 5** the honest participants in the view of P_l either set their values to v, or to b_l, the random bit from P_l.
With probability $1/2$, $b_l = v$, and all honest participants in the view of P_l hold the same value in the next iteration. Since there are at least $(1 - \alpha)n$ of them, and $\delta > 2\alpha$, P_l will accept more than $(1 - 1/2\delta)n$ values of v and sets in $h_l \leftarrow 1$ in the next iteration. □

Theorem 1 now follows directly from Lemma 5, Lemma 8, Claim 9, Claim 10, and Claim 11.

Chapter 4

Negative Result

4.1 Broadcast from Byzantine Agreement

4.2 Negative Result for Broadcast

The case of $\alpha \geq 1/2$ We start with the case where $\alpha \geq 1/2$. Let p, r be positive integers. Without loss of generality, assume $p > r$. Figure shows a configuration where a total of $N = r + 2p$ participants are divided into 4 groups, A, B, C and F.

The case of $\delta \leq 2\alpha$ asdf

Chapter 5

Extension: Unidirectional Edges

Bibliography

[BO83] Michael Ben-Or. Another Advantage of Free Choice (Extended Abstract): Completely Asynchronous Agreement Protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM. event-place: Montreal, Quebec, Canada. 1.1

[CGO10] Nishanth Chandran, Juan Garay, and Rafail Ostrovsky. Improved Fault Tolerance and Secure Computation on Sparse Networks. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, volume 6199, pages 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 1.1

[Dol82] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, March 1982. 1.1

[DS83] D. Dolev and H. R. Strong. Authenticated Algorithms for Byzantine Agreement. *SIAM Journal on Computing*, 12(4):656–666, November 1983. 1.1

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, June 1982. 1.1

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997. 1.1, 3.1

[GM98] Juan A. Garay and Yoram Moses. Fully Polynomial Byzantine Agreement for Processors in Rounds. *SIAM J. Comput.*, 27(1):247–290, February 1998. 1.1

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In *Annual International Cryptology Conference*, pages 228–245. Springer, 1992. 2.2

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In *Advances in Cryptology - CRYPTO*, Lecture Notes in Computer Science, pages 445–462. Springer Berlin Heidelberg, 2006. 1.1, 2.4, 3.2

[KS09] Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{O}(n^{3/2})$ bits. In *Distributed Computing*, Lecture Notes in Computer Science, pages 464–478. Springer Berlin Heidelberg, 2009. 1.1

[LSP] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):20. 1.1, 1.1

[Mic17] Silvio Micali. Byzantine agreement, made trivial. 2017. 1.1, 2.4, 3.2, 3.3

[MRVil] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, page nil, - nil. 2.2, 2.3

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980. 1.1, 1.1

[Rab83] Michael O. Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 403–409, November 1983. ISSN: 0272-5428. 1.1