Byzantine Agreement Using Partial Authentication

Piyush Bansal* Prasant Gopal† Anuj Gupta* K. Srinathan*
Pranav K. Vasishta*

Abstract

Three decades ago, Pease *et al.* introduced the problem of *Byzantine Agreement* [PSL80] where nodes need to maintain a consistent view of the world in spite of the challenge posed by Byzantine faults. Subsequently, it is well known that Byzantine agreement over a completely connected synchronous graph of n nodes tolerating up to t faults is (efficiently) possible if and only if $t < n/3$. Pease *et al.* further empowered the nodes with the ability to authenticate themselves and their messages and proved that agreement in this new model (popularly known as authenticated Byzantine agreement (ABA)) is possible if and only if $t < n$. (which is a huge improvement over the bound of $t < \frac{n}{3}$ in the absence of authentication for the same functionality).

To understand the utility, potential and limitations of using authentication in distributed protocols for agreement, Gupta *et al.* [GGBS10] studied ABA in new light. They generalize the existing models and thus, attempt to give a unified theory of agreements over the authenticated and non-authenticated domains. In this paper we extend their results to synchronous (undirected) arbitrary graphs and give a complete characterization of agreement protocols in the aforementioned family of graphs.

As a corollary, we show that agreement can be *strictly* easier than all-pair point-to-point communication. It is well known that in a synchronous graph over n nodes of which up to any t are corrupted by a Byzantine adversary, BA is possible only if all pair point-to-point reliable communication is possible [Dol82, DDWY93]. Specifically, in the standard unauthenticated model, $(2t+1)$-connectivity is necessary whereas in the authenticated setting $(t+1)$-connectivity is sufficient. Thus, a folklore in the area is that maintaining *global* consistency(Agreement) is at least as hard as the problem of all pair *point-to-point* communication. Equivalently, it is widely believed that protocols for BA over incomplete graphs exist only if it is possible to simulate an overlay-ed complete graph. Surprisingly, we show that the folklore is not always true. Thus, it seems that agreement protocols may be more fundamental to distributed computing than reliable communication.

Keywords: Byzantine Agreement, reliable communication, arbitrary graphs, authentication.

1 Introduction

Informally, the goal of Byzantine Agreement (BA) is to maintain a consistent view of the world in spite of the challenge posed by (Byzantine) faults. The problem was first introduced by Pease *et al.* [PSL80]. They went on to show that BA(in a synchronous and non-authenticated setting) is

*Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India.

†Contact Author: prasant@csail.mit.edu, 32 Vassar Street, CSAIL, MIT, Cambridge, MA - 02139. Phone: 617-939-5548

possible if and only if 2/3 of the nodes are non-faulty. In a seminal paper, Fisher *et al.* [FLP85] proved the impossibility of BA tolerating even a single fail-stop in the asynchronous model. Being a fundamental problem in the area of distributed algorithms, BA has been studied in wide variety of models such as partially synchronous graphs [DDS87], incomplete graphs [Dol82, Lyn96], hypergraphs [FM00], non-threshold adversaries [HM00], mixed-adversaries [AFM99], mobile adversaries [Gar94], and probabilistic correctness [Rab83] to name a few.

Owing to its high fault tolerance, an important variant on BA is the authenticated model proposed by Pease *et al.* [PSL80]. Since generation of authenticated signature for every message is costly, some works choose to consider alternatives for authentication and avoid the excess use of signatures. Specifically, Borcherding [Bor95, Bor96b] investigated the case when signatures are used in only some rounds but not all. A different approach was taken by Srikanth and Toueg [ST87] where authenticated messages are simulated by non-authenticated sub-protocols. In another line of work, Borcherding [Bor96a] considered different levels and styles of authentication and its effects on the agreement protocols. His work focuses on the properties of authentication scheme that allows us to build faster protocols for BA. Katz *et al.* [KK07] investigated the use of other tools such as Verifiable Secret Sharing (VSS) to achieve Byzantine agreement in constant number of rounds. Gong *et al.* [GLR95] studied the assumptions required for the authentication mechanism in protocols for BA that use signed messages. They present new protocols for BA that add authentication to oral message protocols so that additional resilience is obtained with authentication. In all, ABA has been fairly well studied by researchers.

Gupta *et al.* [GGBS10] consider the problem of Authenticated Byzantine Agreement(ABA) under a mixed adversary model. They give completeness theorems for ABA protocols over a complete graph. In this work, we extend their results to arbitrary (undirected) graphs.

1.1 Motivating Example

For starters, consider the graph Figure 1. Imagine the case when either 1 or 3 is Byzantine faulty but both of them have the capability to authenticate themselves; while 2, 4 are non-faulty but do not have the power to authenticate their messages. We claim that in such a scenario 2 cannot reliably send a message across to 4.

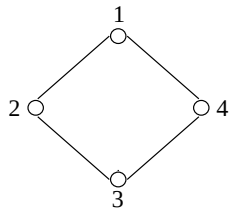

Figure 1: Graph G

In particular, consider the following executions $\mathbf{E_1}$ and $\mathbf{E_3}$: In $\mathbf{E_1}$, 2 intends to send α to 4, and during $\mathbf{E_3}$, 2 wants to send β (different from α). Now, the adversary employs the following strategy: In $\mathbf{E}_\gamma$[1], adversary corrupts γ and sends what an honest γ would have sent in $\mathbf{E}_{\overline{\gamma}}$. It can be shown that the messages received by 4 in both the executions are same and thus, 4 cannot distinguish between $\mathbf{E_1}$ and $\mathbf{E_3}$. The actual views can be proved to be same using inductive arguments. However, we do not take this up in any more detail. We, also, note that we do not use this fact elsewhere in the article.

Thus, it seems that in the aforementioned scenario, nodes in G (ref. Figure 1) cannot have Byzantine agreement(BA) given the parties can't establish a reliable communication channel – which is fundamental to every distributed protocol. But, interestingly, our theorems show that nodes can agree in spite of nodes 2 and 4 not being able to establish a reliable communication link.

Graph G turns out to be a classic example, which seems to suggest - "Perhaps, BA is more fundamental to Distributed Computing than all pair point-to-point communication". We give a simple protocol in Table 1 that solves BA over G. The proof of correctness is simple yet beautiful

[1]Let $\gamma \in \{1, 3\}$ and define $\overline{\gamma}$ to be 3 if $\gamma = 1$ or 1 otherwise.

and we establish it via Theorem 8. This is one of the many interesting examples where it seems that a BA protocol cannot exist - but our characterization gives an "efficient" way to identify if a protocol can indeed exist or not.

2 Model and Definitions

We consider a set of n nodes, communicating over a synchronous graph. That is, the protocol is executed in a sequence of *rounds* wherein in each round, a node can perform some local computation, send new messages to its neighbours, receive the messages sent to him in that round by the nodes (and if necessary perform some more local computation), in that order. The communication network is abstracted as an *undirected* graph. We further assume that there is a communication channel for each edge of the graph. Also, the communication channel between any two nodes is perfectly reliable and authenticated. We remark that all the nodes are *aware* of the topology of the graph. During the execution, the adversary may corrupt up to t nodes. The adversary can make the corrupted node behave in an arbitrary way. Further, the adversary can read the internal states of up to another k nodes. We refer to such an adversary as (t,k)-adversary. One may view such an adversary as a mixed adversary. We restrict ourselves to a static, threshold adversary.

We also assume existence of authentication tools such as Public Key Infrastructure (PKI) and Digital Signature Schemes (DSS). Nodes can authenticate themselves and their messages with these authentication tools. We assume that every node has a secret key SK and a signature scheme $Sign(\mathsf{SK}, \mathsf{mesg})$ that allows it to sign message mesg with its signature. Also, we assume that any node can verify if a message carries a valid signature of a node. It is assumed that the nodes sign whenever they send any message and also discard any received message that does not have a valid signature on it. This ensures that the receiver can uniquely identify the sender of the message. Since the adversary can look into the internal states of k nodes outside its control, it can forge the signature of all the k. So, in all the adversary can forge/generate the signatures of $(t + k)$ nodes. From now on, we use the term κ-connected graph to denote a κ-vertex connected graph. Also, throughout the paper we use n to denote the number of nodes in the graph. Every node starts with an input value from the set $V = \{0, 1\}$.

Definition 1 (Byzantine Agreement).

- Agreement: *All non-faulty nodes decide on the same value $u \in V$.*
- Validity: *If all non-faulty nodes start with the same initial value $v \in V$, then $u = v$.*
- Termination: *All non-faulty nodes eventually decide.*

Here a node is considered as *faulty* if and only if he deviates from the delegated protocol. Therefore, the nodes that do not deviate from the designated protocol are *non-faulty* nodes. A node who follows the designated protocol diligently, even if adversary has complete access to his internal state is referred as *passively corrupt* node. A node is *honest* if he follows the designated protocol, and over whom adversary has absolutely no control. In particular, the adversary cannot replicate/forge the signature of honest nodes. For the purpose of this paper, we refer to both *honest* and *passively corrupt* nodes together as *non-faulty*.

Definition 2 ((t,k)-BA Protocol)**.** *A protocol is a (t, k)-BA protocol if it accomplishes Byzantine agreement (ref. Definition 1) in the presence of a (t,k)-adversary.*

Definition 3 ((2t,t)-Connectivity)**.** *A graph $\mathcal{N} = (\mathbb{P}, \mathcal{E})$ is $(2t, t)$-Connected if its minimum degree is at least $2t$ and it is $(t + 1)$-connected.*

3 Results and Contributions

The contributions of this paper are many fold :

1. *Complete Characterization:* We give the necessary and sufficient condition(s) for designing protocols for agreement over (undirected) arbitrary graphs. We prove that BA Protocols over a n node graph tolerating a (t,k)-adversary exist if and only if $n > 2t+ \min(t,k)$ and the graph is

$$\begin{array}{rlrcl} (t+1)\text{-connected} & if & & n & > (2t+k) \\ (2t,t)\text{-connected} & if & (t+k) & < n & \leq (2t+k) \\ (2t+1)\text{-connected} & if & & n & \leq (t+k) \end{array}$$

   The above holds for $k > 0$. For $k = 0$ it reduces to $n > t$ and graph should be $(t+1)$-connected.

2. *Unification:* In the standard authenticated model (ABA) [PSL80] the adversary can forge messages only on behalf of corrupt nodes. On the other hand in the unauthenticated model (BA) every node can be treated as passively corrupt node[2]. Thus, characterizing possibility of BA protocols for the entire spectrum leads to unification of the extant literature on BA. As an elaboration consider the result presented in previous paragraph. It says if $n \leq (t+k)$ then $2t+1$ connectivity is necessary and sufficient – which is characterization of BA over non-authenticated setting. To elaborate, $n \leq (t+k)$ implies there are no honest nodes with a signature scheme that the adversary cannot forge and thus, collapses to the setting of non-authenticated BA.

3. *Agreement can be easier than all pair point-to-point communication:* From the results presented in this paper, it is evident that for all graphs with $n > (2t+k)$, $(t+1)$-connectivity is sufficient for agreement. We, now, show that if $k > 0$ then for simulating point-to-point communication $(2t+1)$-connectivity is necessary. Consider a scenario where the Sender, say S, is non-faulty but the adversary can sign for S; in such a scenario, it is well-known that if the graph is not $(2t+1)$-connected, there must exist a node j such that reliable message transmission from S to j is impossible[DDWY93]. From the above argument one can see that BA is easier than all pair point-to-point communication.

4. *BA is easier than Byzantine Generals (BG) [PSL80, LSP82]:* Informally, the problem of reliable broadcast in presence of Byzantine faults is also studied under the name of BG. Note that if a protocol for BG exists, then it vacuously is also a protocol for reliable point-to-point message transmission. Till this juncture, BA and BG have been isotopic forms. That is, in the authenticated and un-authenticated models, BA iff BG. However, we show that BA and BG are two different problems and in fact BA is more primitive and fundamental to distributed computing than BG. In other words, there are several graphs over which BA is possible whereas BG is impossible, in spite of having an overwhelming non-faulty majority.

3.1 Organization of the paper

From now on, we assume that $k > 0$. We, also, assume that, *w.l.o.g*, either $n > (2t+k)$ or $(t+k) < n \leq (2t+k)$ or $n \leq (t+k)$. We consider these cases in Sections 4, 5 and 6 respectively. We establish the main theorem of the paper in Theorem 14.

[2] A node not having no authentication facility at all can also be visualized as passively corrupt and therefore, the adversary can forge messages on his behalf

4 The Good: When the honest are in abundance

Lemma 1. *(t, k)-BA protocol over any general graph $\mathcal{N} = (\mathbb{P}, \mathcal{E})$, $|\mathbb{P}| > (2t + k)$, exists if and only if $\mathcal{N}$ is $(t + 1)$-connected.*

Proof. *Necessity:* The necessity of $(t + 1)$-connectivity is straight forward due to the presence of t Byzantine faults. Elaborating further, the adversary may crash t nodes and disconnect the graph. *Sufficiency:* We assume that every node i has a secret key $\mathsf{SK_i}$ and a signature scheme $Sign(\mathsf{SK_i}, \mathsf{mesg})$ that allows i to sign message mesg with i's signature. Also, we assume that any node can verify if a message carries a valid signature. It is assumed that the nodes sign whenever they send any message and also discard any received message that does not have a valid signature on it. Nodes run the Flood-Set protocol given in Algorithm 1.[3]

Algorithm 1 Flood-Set$(\mathcal{N}, i, \sigma)$ — Node i starts with its input $\sigma \in \{0, 1\}$

```
Ω[n] = ∅                                   ▷ Maintain a set for each node in the graph, Initially empty
for each j : (i, j) ∈ E do
    Send(σ, j)                             ▷ i sends its input to its neighbours
end for
Round ← 1
while Round ≤ 2n do                        ▷ Flood for 2n rounds
    for each j : (i, j) ∈ E do
        Receive(j)
        ∀x ∈ P, Ω[x] = Ω[x] ∪ Messages originating from node x and received from j
    end for
    for each j : (i, j) ∈ E do
        Send(Ω, j)                         ▷ Send messages to neighbours
    end for
    Round = Round + 1                      ▷ Increment Round
end while
```

Lemma 2. *All non-faulty nodes will have a consistent view of whether node i's execution was clean or dirty.*

Proof. We define a contradiction as if at some point in the execution, a node i receive at least two valid messages with different content. If in any round $R \in \{1, ..., (n - 1)\}$ a non-faulty node encounters a contradiction, then from the specification of the protocol, all non-faulty nodes see a contradiction in round $R + n$, all agree that the run was dirty, and all output a default value.

However, assume that no non-faulty node observes a contradiction before the beginning of round n. In order to observe a contradiction in round n, i must receive two messages, each of the messages carrying with n signatures. Recall, however, that the adversary can forge at most $n - 1$ signatures, since $n > t + k$[4], and therefore some honest node, a node with a secure signature scheme, must have signed both messages in one of the previous round. This contradicts our assumption that no non-faulty node saw a contradiction prior to round n. Hence, all non-faulty nodes can agree up on the status of any execution. □

[3] This protocol is essentially the Dolev-Strong protocol [DS83] followed by n rounds of flooding.

[4] This is a weaker condition than $n > 2t + k$, but we shall use Flood-Set protocol as a sub-routine for the cases when only this weaker condition is met.

Lemma 3. *If node i is honest, then all non-faulty nodes agree on the sender's input and i's execution is clean for any non-faulty node.*

Proof. Since j is an honest node, whose signature cannot be forged by an adversary, and the graph is $(t+1)$-connected it simply implies that j's location in Ω is consistent across all non-faulty i. □

Theorem 4. *Flood-Set Protocol, given as Algorithm 1, is a (t, k)-BA protocol, given $n > (2t + k)$ and the graph is $(t + 1)$-connected.*

Proof. Termination is obvious. For validity, since $n > 2t + k$ this implies $(n - t - k) > t$. Notice that $(n - t - k)$ denotes the number of honest nodes(that is, nodes with a good signature scheme). And hence, we can conclude that the honest nodes out number the corrupt nodes. Since, all honest nodes start with the same value v, from Lemma 3, the decision rule simply implies that v is the only possible decision. For agreement, Lemma 2 implies that all for any two non-faulty nodes i, j Ω is consistent across both of them. And since the same decision rule is applied across the nodes, agreement is guaranteed. □

Node x deems the execution/invocation of the flood-set protocol by node j as *dirty* if he detects the Byzantine influence(i.e., if x received two different inputs from j with a valid signature of j or never receives any messages with valid signature of j); otherwise we say that the execution is *clean* for i. (Note that in our setting an execution can be *dirty* when either j is either faulty or passively corrupt.)

At the end of Flood-Set protocol, x modifies Ω in the following way: the j^{th} location of this tuple is changed to a $\perp$ if node j's flood-set execution is *dirty*, else if it is *clean* (and thus, had a single value v), then $\Omega[j] = v$. x takes a majority over Ω and outputs it as its decision value; otherwise, it outputs a default value. The proof of correctness hinges on the fact that $n > 2t + k$. Rearranging the terms, $(n - t - k) > t$, means that the honest nodes are in strict majority. Thus, the clean runs of the honest nodes carries us through. □

5 The Bad: When faulty outnumber the honest

We now take a detour and limit our focus to 2-connected graphs. Specifically, we first construct $(1, \psi)$-BA protocol on a 2-connected graph, where $\psi \in [2, n - 2]$. Our approach can be outlined as follows: we design a protocol Π for the weakest case, that is to say the adversary *always* uses his full power and corrupts exactly 1 node actively and $(n - 2)$ nodes do not have the power to authenticate themselves. It is straightforward to see that such a protocol would also work for the stronger assumptions in which more nodes have the power to authenticate themselves or the adversary does not corrupt anyone. Finally, we extend Π to a more general setting where the adversary controls $t \geq 1$ nodes.

5.1 Π: The baby protocol

Designing Π*:* Nodes exchange messages as per the Flood-Set protocol, given as Algorithm 1. Node i applies a "modified" decision rule, which is as follows: If a majority exists over all the clean runs, then i outputs it as his decision and halts. Else, if the number of nodes which had a clean run is more than 2, i outputs 0 and halts. However, if the number of clean runs is only 2: Say, only runs of nodes a and b are clean. Notice that, one of the nodes a or b must be honest while the other node may be corrupt.

We now make a big assumption (and later show how to get rid of it): We assume that both a and b are non-faulty. Node x, $x \in \mathbb{P} - \{a, b\}$, sends his input to node a using the following routing strategy: (a) through the direct edge(if it exists), (b) otherwise a and b have at least 2 vertex disjoint paths between them and all the nodes in these paths must use these paths only. (c) else let $\chi = \{p_1, p_2..., p_j\}$ be the set of paths to a from x, if any such path includes b, x chooses it otherwise it chooses the shortest path to a. x also sends its input to b along an analogous set of rules. Nodes a and b are required to sign with their signature and send them back to x along the same path. If a and b receive more than one value from a node or not along the routing protocol given, a and b do not respond. Suppose, x receives, say, α and β from a and b respectively. If either a or b do not match with his input bit, x drops that message (Note that he cannot infer anything whether the node who signed on the toggled bit is corrupt or not).

Now, every node tries to get its input signed by a and b. After that, every node runs the Flood-Set protocol, given as Algorithm 1, twice. The first time with its input bit signed by a and the next time with its input signed by b. If both these runs turn out to be clean but the values contradict each other – both runs of i are overruled to be dirty. And if i has both its runs clean and consistent – then i's run is declared clean. At the end of Flood-Set protocol, i take a majority among all clean runs including the two runs of a and b; otherwise it decides on a default value(say, 0) and the protocol terminates. This would work as along as one of the node's execution in the flood-set protocol is clean. We, now, prove that if both a and b are non-faulty, then at least one node will have a clean run.

Lemma 5. *If both a and b are non-faulty, at least one node shall have a clean run.*

Proof. Since $\mathcal{N}$ is 2-connected, there can be two cases: (a) a, b are a vertex cut-set[5] in $\mathcal{N}$. (b) a, b are not a vertex cut-set. In the former, it is easy to see that the claim is maintained as there shall be at-least two components upon removing a and b and it is clear that the adversary may be present only in one component. Thereby, the nodes in the other component can get the signature of a and b and hence, they will be able to sent their value to all other nodes successfully - one of a or b is honest and the other is non-faulty - so either both the runs will be clean with a consistent value or one of the runs will be clean and the other will be dirty - and from the protocol, both these cases are deemed to be clean runs. The case of both runs being clean with a contradictory value can happen only if one of a, b and the node is faulty.

In the later, there is only one component upon removing a and b. If a and b are adjacent, notice that the both a as well as b are sure to have a neighbors which is other than b and a respectively (as $\mathcal{N}$ is 2-connected, neighborhood of each node is at least 2). In this case, one of them is guaranteed to be non-faulty and hence from the routing method it is easy to see that, one of these node's will have a *clean* run (arguments go similar to previous case). If a and b are neither vertex cut-sets nor adjacent then there are at least 2 vertex disjoint paths between a and b. And active adversary resides in only one of them. Hence, at least of the nodes in the other path will send its input bit to get signed from a, b as per the routing algorithm. Hence, it easy to see that such a node *always* exists and hence at least one of the nodes modulo a, b will have a *clean* run. And, thus the Lemma. □

Observe that, all nodes would have agreed, under the big assumption that both a and b were non-faulty. However, if the protocol has not had the clean run - Lemma 5 implies that either a or b is faulty. Depending on whether nodes a and b are a vertex cut-set or not, nodes do the following.

[5] A vertex cut-set in a graph is a set of vertices whose removal from the graph makes it disconnected.

5.1.1 The shallow side

If a and b are *not* a vertex cut-set, a publicly chosen (say, the node with the least UID outside a, b) non-faulty node i may send his input to everyone using paths outside a and b. i sends his input to a through a path avoiding b and to b via path avoiding a. Note that these paths are bound to exist as it is a 2-connected graph. This completes the construction of Π when a and b are not a cut-set.

5.1.2 The far side: Diamond protocol

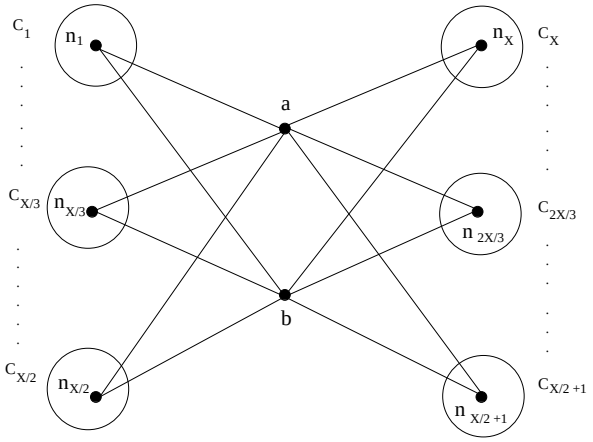

Figure 2: Graph $\mathcal{N}'$

If a and b are a vertex cut-set, say a and b partition the graph $\mathcal{N}$ into x components $c_1, c_2, \ldots c_x$. Now, nodes choose a representative from each of these components, say n_i from is chosen from c_i (via some function on UID's). Nodes, now, create a (virtual) overlay-ed graph $\mathcal{N}'$ whose vertices include n_i's, a and b. An edge appears between any two vertices only if there is an edge between the components represented by them. Note that we are not looking for a direct edge between n_i's, we are looking for edge between C_i's and a, b (each edge may have to be simulated via one or more nodes). Notice that every n_i has to be connected to both a and b (follows from 2-connectivity of $\mathcal{N}$). Thus, $\mathcal{N}'$ (ref. Figure 2) has to be 2-connected (follows from $\mathcal{N}$ being 2-connected and that a, b are a cut). Now, we pair every node in $\mathcal{N}'$ (other than a and b) with another node (pairings are chosen via a pre-decided strategy)[6]. Consider one such pairing (n_x, n_y). Now, nodes a, b, n_x and n_y simulate graph G in Figure 1. Specifically, a, b, n_x and n_y simulate nodes 1, 3, 2 and 4 respectively. They execute rounds 1 and 2 of the Diamond protocol (Table 1). It is assumed that nodes sign on all the messages they send and discard any received message with an invalid signature. The rest of the nodes, merely, act as routers relying messages. The only difference w.r.t. the Diamond protocol is that in lieu of rounds 3 and 4, node 4(that is, n_y) executes the Flood-Set protocol given in Algorithm 1 on graph $\mathcal{N}$ with the tuple containing inputs ψ_1 and ψ_3. If it is a *clean* run, nodes decide on that value and halt.

Claim 6. *If node* 4 *does not receive the input value of node* 2 *by the end of this protocol, node* 2 *can identify the Byzantine faulty node.*

Proof. Let node 2 starts with a input bit $\alpha \in \{0, 1\}$ and let node 4 receive values α, β[7] through the two vertex disjoint paths. *w.l.o.g*, β can take the values either $\alpha, \overline{\alpha}$ [8] or a *null*. In case β is $\overline{\alpha}$, then 4 does not receive 2's input. However, as per protocol 4 now sends the value $\overline{\alpha}$ via the other path, which is non-faulty and thus, sends them across to 2. While, the value that is received along the other path cannot be toggled to $\overline{\alpha}$ as it has the signature of a non-faulty node which cannot be forged. Upon receiving this, 2 can easily see either of 1 or 3 is faulty by looking at the signatures on the message. In the other two cases, 4 would have received 2's input bit. □

Claim 7. *Node* 2 *can sense if his input value has been reliably communicated to node* 4.

Proof. If 2's input is reliably transmitted to 4, we will show that the adversary cannot prevent 2 from being ignorant of the same. Notice that as per the protocol, 4 sends the values received from

[6] A node can be involved in multiple pairings

[7] Note that since of the two paths is non-faulty, at least one of the values that node 4 receives is α.

[8] we use the notation $\overline{\alpha}$ to denote the complement of input value of node 2.

Code for node 1:

Let node 1 start with an input $\sigma_1 \in \{0,1\}$.

1. Receives the values sent by nodes 2 and 4 and them across to 4 and 2 respectively.
2. Do nothing.
3. Receive ψ from node 4 and send it to 2.
4. Do nothing.
5. Receive from 2 and output the same.

Code for node 3:

Let node 3 start with an input $\sigma_3 \in \{0,1\}$.

1. Receives the values sent by nodes 2 and 4 and them across to 4 and 2 respectively.
2. Do nothing.
3. Receive ψ from node 4 and send it to 2.
4. Do nothing.
5. Receive from 2 and output the same.

Code for node 2:

Let node 2 start with an input $\sigma_2 \in \{0,1\}$.

1. Sends σ_2 to nodes 1 and 3.
2. Receive ψ_1 and ψ_3 from nodes 1 and 3 respectively.
3. Do nothing.
4. Receive ψ'_{31} from node 1 and ψ'_{13} from node 2. If any of these values is a $\perp$ then replace it with his input value, σ_2.
   If $\psi'_{13} = \psi'_{31} = \sigma_2$, then decide on σ_2.
   Else if $\psi_{ij} \neq \sigma_2$, then decide on ψ_j.
5. Send the value decided upon to 1 and 3.

Code for node 4:

Let node 4 start with an input $\sigma_4 \in \{0,1\}$.

1. Sends σ_4 to nodes 1 and 3.
2. Receive ψ_1 and ψ_3 from node 1 and node 3 respectively.
3. Send the value ψ_3 to node 1 and ψ_1 to node 3.
4. Create a set W from ψ_1 and ψ_3. If $|W| = 1$ decide on that element, else output σ_4.
5. Do nothing.

Table 1: Diamond protocol.

1 via 3 and vice-versa. By arguments similar to those in Claim 6, we can see that if 4 received 2's value, then at the end of protocol, 2 will not receive a toggled value of his input. So, he either receives either $\perp$ via both paths or a $\perp$ along one of the paths. Notice that in both these cases, 2 can always be assured that 4 received his input. □

Theorem 8. *Protocol given in Table 1 accomplishes BA protocol on graph G, when either* 1 *or* 3 *is corrupt by the Byzantine adversary and* 2, 4 *have no authentication schemes.*

Proof. Termination is obvious. For validity - observe that at the end of the BA protocol in Table 1, nodes output either the input value of node 2 or 4 (both these nodes are non-faulty) and hence, validity will always hold. For Agreement, by Claim 7, it easy to see that node 2 can sense whether 4 has reliably received its input. If 4 does not receive node 2's input, by Claim 6 2 can find out the adversary. *w.l.o.g* say 1 is the adversary, and hence node 4's input value (it needs to consider the input value of node 4 received from node 2). Once, node 2 knows what value to be agreed upon, it sends this value to nodes 1 and 3 and thus, agreement is attained. □

Notice that if nodes have not agreed on any value, we can invoke Claim **??** to infer that node 2(here it is, n_x) can identify the adversary (between a, b). If nodes haven't decided yet: nodes n_x and n_y swap their codes, that is, n_x deploys the code of 4 and vice-versa and then, they re-execute the Diamond protocol given above. If nodes still did not agree, both n_x and n_y can identify the faulty node is (out of a and b). The executions proceed until nodes have decided and halted or

when all the pairings have tried their luck. When all the pairings are exhausted, notice that all n_i's can identify the adversary and this as good as the adversary making himself public!

All the n_i's agree on the input of the non-faulty node (out of a and b). Now, each of these n_i's send the decision to the nodes in the connected component represented by n_i and also to a and b. This completes the construction of Π ($(1, n-2)$-BA protocol).

Lemma 9. *For every 2-connected graph on n nodes, Π is a $(1, n-2)$-BA protocol.*

Proof. Termination is obvious. For agreement, the use of Flood-Set (Algorithm 1) and the Diamond protocol for communication ensures that all the non-faulty nodes have consistent values and hence the decision rule simply implies that all of them *agree* on the same value. If all non-faulty nodes start with same input σ, every node's input (modulo the faulty ones) has to be σ and the protocol decides only upon receiving at least inputs from three nodes. Thus, if all the non-faulty nodes start with the same input, σ is the only possible output. Hence, by definition, it is a $(1, n-2)$-BA protocol over a 2-connected graph. □

5.2 Beyond 2-connected networks

We, now, extend Π to an arbitrarily connected network. Before that, we introduce new machinery. It is convenient to model the threshold adversary as a non-threshold adversary [HM97, FM98, HM00]. Informally, a non-threshold adversary captures the faults by a fault structure, that is, an enumeration of all the possible "snapshots" of faults in the network. Note that a single snapshot can be described by an ordered pair (B, K), where $B, K \subseteq \mathcal{I}$ and $B \cap K = \emptyset$, [9] which means that the nodes in the set B are Byzantine faulty while the nodes in the set K are passively corrupt. A fault structure is a collection of such pairs. More precisely, we define the fault structure by $\mathcal{A}$, where $\mathcal{A} \subseteq 2^{|\mathcal{I}|}$. The adversary is allowed to corrupt any pair from the fault structure. The fault structure is *monotone* in the sense that if $(B_1, K_1) \in \mathcal{A}$, then $\forall (B_2, K_2)$ such that $B_2 \subseteq B_1$ and $K_2 \subseteq K_1$, $(B_2, K_2) \in \mathcal{A}$. We note that $\mathcal{A}$ can be uniquely represented by listing the elements in its *maximal basis* $\overline{\mathcal{A}}$ which we define below. In what follows, unless specified otherwise, we work with only the maximal basis of $\mathcal{A}$.

Definition 4 (Maximal Basis of $\mathcal{A}$)**.** *For any monotone fault structure $\mathcal{A}$, its maximal basis $\overline{\mathcal{A}}$ is defined as $\overline{\mathcal{A}} = \{(B, K) | (B, K) \in \mathcal{A}, \nexists (X, Y) \in \mathcal{A}, (X, Y) \neq (B, K), X \supseteq B \text{ and } Y \supseteq K\}$*

Another way of representing the adversary is via an Fault Basis $\mathcal{A}$ given in definition 4 with the understanding that the any one pair (B_i, K_i) from $\mathcal{A}$ is under the control of adversary and it may corrupt the nodes in B_i in Byzantine fashion and nodes in K_i passively. It is evident that a (t, k)-fault is characterized by a fault basis $\mathcal{A} = \{(B, K) || B| \leq t, |K| \leq k, B \cap K = \emptyset\}$. We define the size of the fault-basis to be the number of (B, K) pairs in the set $\mathcal{A}$. From now on, we work with the maximal basis of $\mathcal{A}$.

5.2.1 The case of 3-sized structures.

We first give the characterization for the case of 3-sized structures and then extend it to any adversary structure. We begin by setting the stage for constructing protocols on $\mathcal{N}$ tolerating $\mathcal{A}$. Since $(t + k) < n$ (and thus, $|B_i| + |K_i| < |\mathcal{I}|$), there is at least one honest node. Let us denote the honest node when the adversary corrupts (B_i, K_i) by h_i.

[9] This is not a serious assumption as if there some nodes common to both sets, such nodes can *w.l.o.g* placed solely in set B.

Assume that upon removing the nodes in $(B_1 \cup B_2 \cup B_3)$, say $\mathcal{N}$ is partitioned into x components, namely, $c_1, c_2, \ldots c_x$. Let us denote the honest node when the adversary corrupts (B_i, K_i) by h_i. Now, we choose a representative from each of B_1, B_2, B_3 and each of the components in the following fashion: If any of them have a h_i's, it is chosen as the representative; otherwise, the node with the lowest UID is picked. Note that, at most two of the three h_i's, say h_α and h_β, may lie within a B_i (this follows from the definition of h_i). Our goal is to ensure the presence of an honest representative. Consider the case when h_2 and h_3 lie inside B_1. In this case, if B_1 is corrupt, h_1 is honest and will have an honest representative and our target is achieved. However, when B_1 is not corrupt, one of h_2 or h_3 is honest (follows from definition of h_i's), but we are not sure which of them is honest. So, we need to be a little smarter in picking up a representative. Hence, we create a virtual node and use it as a representative in case both h_2 and h_3 lie inside B_1. The virtual node we create has the following property: Either it is honest or faulty but never passively corrupt. As with every virtual node, it is crucial to define its simulation, the notion of send/receive for the virtual node and its signature. For an exposition on virtual nodes, we refer the readers to [HM00].

Simulation of virtual node. Nodes h_2 and h_3 combine to simulate the virtual node. Since, they may not be adjacent, we need to specify how they communicate. If h_2 and h_3 have a path consisting of nodes exclusively from B_1 and those outside the B_2 and B_3, they use this path to communicate and agree[10]. However, if all paths between h_2 and h_3 have a node either from B_2 or B_3, communication is carried out as follows: They send the values to each other via any two paths (chosen deterministically) such that one of them avoids nodes in B_2 and the other avoids B_3(such paths are guaranteed to exist as the network is $(t+1)$-connected). h_2 and h_3, now, take a majority over these values among the values obtained in the *clean* runs[11]. If they share a path exclusively in B_1 - when B_1 is not corrupt, the objective of creating an honest is achieved as h_2, h_3 are non-faulty and they share a good path and when it is corrupt, the simulation is allowed to fail. When the construction uses 2 paths (one from B_2 and the other from B_3), if either h_2 or h_3 has a signature scheme which the adversary cannot replicate (in other words, honest) - the only value that can be received consistently is the value of the honest node. Thus, the simulation is consistent. While if one of them is faulty, the protocol does not rely on the simulation and we are allowed to fail.

Signature. The notion of signature for a virtual node is a natural extension of the simulation. Any message which has to be signed by the virtual node contains a sequence of signatures from h_2, h_3 and the nodes along the communication path. The verification relies on the fact that every node knows h_2, h_3 and the nodes involved in the simulation.

Send/Receive. The virtual node sending a message to node i is defined as follows: h_2 and h_3 exchange messages using the aforementioned paths and then, they run the Flood-Set protocol to send the transcripts to i. i, then, takes a majority among the *clean* runs of the Flood-Set to extract the transcript and thus, the message of the virtual node. The Flood-set works when the virtual node is honest. If it is not honest, the flooding is allowed to fail. i sending a message to the virtual node is equivalent to - i sending message separately to h_2 and h_3 and they exchange the messages they received from i and then, take a majority over the clean runs. The consistency of send, receive rely on the paths chosen and the fact that one of the paths is always good. The signatures of the nodes along the path play a crucial role in allowing i to verify the transcripts.

[10] which is only possible when both are non-faulty. However, when they are a faulty they cannot agree and the simulation fails. However, in this case the protocol does not rely on this virtual node to provide a honest representative.

[11] Clean runs are those in which only one message with a valid signature is received.

Lemma 10. *(t,k)-BA protocol over any general network $\mathcal{N}=(\mathbb{P},\mathcal{E})$, when $(t+k)<|\mathbb{P}|\leq(2t+k)$, tolerating a 3-sized fault basis $\mathcal{A}=\{(B_1,K_1),\ (B_2,K_2),\ (B_3,K_3)\}$, $\forall(B_i,K_i)\in\mathcal{A},\ |B_i|=t$ and $|K_i|=k$, exists if and only if $\mathcal{N}$ is $(2t,t)$-connected.*

Proof *Sufficiency:* Say n_i is chosen from c_i (components formed after removing B_1, B_2 and B_3 from $\mathcal{N}$) and b_i from $B_i, i\in\{1,2,3\}$. We, now, create a overlay-ed network $\mathcal{N}'$(this construction is similar to Section 5.1.2) on n_i's and b_i's in which an edge appears between any two nodes(representatives) only if there is a edge between the components represented by them. Each of these n_i's is connected to at least two b_i's (Since, $\mathcal{N}$ is $(t+1)$-connected). $\mathcal{N}'$ is a 2-connected with at most one of the b_i's being faulty and at least one of the representatives being honest. Nodes in $\mathcal{N}'$ now execute Π and by Lemma 9 all the (non-faulty) representatives agree. After the representatives agree, they distribute their decision across their components and to the representatives of B_i's. For nodes inside a B_i, since the network $\mathcal{N}$ is $(2t,t)$-connected (Definition 2), every node has a degree at least $2t$. This implies that any node in B_i has a direct edge to a node outside all B_i's OR has a direct edge to one node from each of the B_i's. In the former, it can decide on the value obtained from outside B_i's. In the latter, it takes a majority among the three values received from each of the B_i's. This is bound to work as two of the three B_x's are non-faulty and would have agreed in the execution of Π. The proof of correctness stems from the fact that once the representatives agree (similar to Lemma 9), it is easy that all the nodes in their respective components also agree. This completes the sufficiency.
Necessity: We shall now prove that a (t,k)-BA protocol over a graph $\mathcal{N}$ on n nodes, $(t+k)<n\leq(2t+k)$, does not exist if $\mathcal{N}$ is not $(2t,t)$-connected. If a graph is not $(2t,t)$-connected it implies that either $\mathcal{N}$ is not $(t+1)$-connected or $\mathcal{N}$ has a node, call it u, with a neighborhood of at most $(2t-1)$ nodes. We let B_0 and B_1 to be any partition of u and the neighbours of u into two equal halves.

The necessity of the $(t+1)$-connectivity is straight forward as there can be as many as t Byzantine faults. For the latter, we shall prove the impossibility for a 2-sized fault basis $\mathbb{A}=\{(B_0,\mathbb{P}\setminus(B_1\cup B_0)),\ (B_1,\mathbb{P}\setminus(B_1\cup B_0))\}$. Note the sizes of both B_0 and B_1 are still bounded by t and $|\mathbb{P}|=2t+k$, where $k=|\mathbb{P}-\{B_0\cup B_1\}|$. Define two executions $\mathbf{E}_0$ and $\mathbf{E}_1$ as follows. In the execution $\mathbf{E}_\alpha\in\{\mathbf{E}_0,\mathbf{E}_1\}$, the set B_α is Byzantine corrupt and all nodes *except* those in $B_\alpha\cup B_{\overline{\alpha}}$[12] are passively corrupt. In both executions, u starts with an input 0 and is not corrupted by the adversary. In $\mathbf{E}_\alpha$, all the nodes in $\mathbb{P}-\{B_\alpha\cup\{u\}\}$ start with input α. The behavior of the Byzantine set B_α in the execution $\mathbf{E}_\alpha$ is to send no message whatsoever to anyone other than u. And to u, it sends exactly the same messages that are sent by an honest B_α in the execution $\mathbf{E}_{\overline{\alpha}}$. In order for the Byzantine corrupt B_α to behave as specified in the execution $\mathbf{E}_\alpha$, it needs to simulate the behavior of $\mathbb{P}-B_{\overline{\alpha}}$ in the execution $\mathbf{E}_{\overline{\alpha}}$. To this end, the adversary simulates round-by-round the behavior of the vertices in $\mathbb{P}-B_{\overline{\alpha}}$ using their respective inputs in the execution $\mathbf{E}_{\overline{\alpha}}$. Notice that in $\mathbf{E}_{\overline{\alpha}}$ the set $B_{\overline{\alpha}}$ does not any send messages to anyone except u. Thus, during $\mathbf{E}_\alpha$, B_α neither needs to simulate $B_{\overline{\alpha}}$ nor forge the signature of nodes in $B_{\overline{\alpha}}$. Also, the adversary can sign on behalf of the rest of the nodes $(\mathbb{P}-B_{\overline{\alpha}})$. So, the adversary has all the details to carry out the simulation and all that remains is to construct the simulation.

The adversary simulates the behaviour of a node round by round. At the beginning of each round, each simulated node has a history of messages that it got in the simulation of the previous rounds. It uses them to simulate the messages for the next round. The real messages sent by u to B_α are added to the history and used in simulation whenever the need arises. The simulated

[12] $\overline{\alpha}$ is used to denote the complement of α

node sends the same messages that the node would have sent during the same round in $\mathbf{E}_{\overline{\alpha}}$. The messages that the simulation sends to u are the real messages sent by B_α to u. By definition, these messages are exactly the same as the ones sent by B_α in the execution $\mathbf{E}_{\overline{\alpha}}$. The rest of the simulation is only used/maintained to update the history for the next round. The history of messages of each simulated vertex in execution $\mathbf{E}_\alpha$ is the same as the history of the vertex in execution $\mathbf{E}_{\overline{\alpha}}$. Therefore, the messages sent by B_0 and B_1 to u in both executions are exactly the same. Hence, u cannot distinguish whether it is in $\mathbf{E}_0$ or $\mathbf{E}_1$ and remains in a bivalent state. Thus, the adversary can ensure that some non-faulty node will remain bivalent for as long as it wants! This completes the necessity and thus, the proof of Lemma 10. □

We, now, extend the characterizations over a 3-sized fault basis to an n-sized fault basis.

Lemma 11. *(t, k)-BA protocol over a graph $\mathcal{N}$ tolerating a n-sized fault basis $\mathbb{A}$ exists if and only if there exists (t, k)-BA protocols for every 3-sized fault basis $\mathbb{B}$, $\mathbb{B} \subseteq \mathbb{A}$.*

Proof. This extension is based on the works of [HM97, HM00].
Sufficiency: We construct a protocol using induction on the size on the fault basis.
Base Case: Lemma 10 gives us a protocol for every 3-sized fault basis. So, induction starts when size is 3.
Inductive hypothesis: Assume that protocols tolerating fault sizes less than n are given.
Inductive step: We construct protocols for n-sized fault structure $\mathbb{A}$. We partition $\mathbb{A}$ into four non-empty sets $\mathbb{A}_1$, $\mathbb{A}_2$, $\mathbb{A}_4$ and $\mathbb{A}_4$.[13] Now, consider the fault basis $\mathcal{A}_i = \{\mathbb{A} - \mathbb{A}_i\}$. By inductive hypothesis, we have protocols for all $(n-1)$-sized fault basis. Denote the protocol for BA tolerating $\mathcal{A}_1$ by ψ_1. Define ψ_2, ψ_3, ψ_4 analogously. It is easy to see that every element of $\mathbb{A}$ belongs to at least three of the $\mathcal{A}_i$'s. Hence, any $x \in \mathbb{A}$ is tolerated by at least *three* of the four ψ's. Using the ψ_i's, we create four virtual nodes with the guarantee that at least three of them are honest.

The four virtual nodes, call them $v_1, ...v_4$, are constructed as follows: All nodes take part in the simulation. Node v_i sending its input to v_j is same as the nodes running ψ_j on the output of ψ_i. This establishes the notion of sending a message between any two virtual nodes. The virtual nodes, now, simulate a $(1, 3)$-BA protocol [Lyn96], call it Π_4^1. Informally, it is the BA protocol tolerating a single Byzantine fault in the *non-authenticated setting* on 4 nodes. At the end of the simulation of Π_4^1 protocol, each node in $\mathbb{P}$ takes a majority on the decision value of these four virtual nodes (Each node was a part of 4 virtual players and hence, it has one decision value for each of the virtual player).

A comment on the majority voting is due. Since, only one of the ψ's can fail for every x *in*$\mathbb{A}$, one of the virtual nodes is faulty. This virtual node may deviate in a arbitrary way from the protocol and produce bizarre outputs. However, note that the other three virtual nodes are non-faulty w.r.t x and hence, they must agree. Now, every node simulating the four virtual players will have the same decision value for 3 out of 4 virtual nodes. Hence, a majority vote will satisfy the agreement and validity conditions of BA. Thus, we have successfully tolerated a n-sized fault basis. This completes the construction of a protocol tolerating a n-sized fault basis from protocols tolerating a $n-1$-sized fault basis.

Necessity: If BA tolerating a 3-sized fault basis is impossible, then it remains impossible w.r.t a n-sized fault basis. □

Lemma 12. *(t, k)-BA protocol over a graph $\mathcal{N} = (\mathbb{P}, \mathcal{E})$, $(t + k) < n \leq (2t + k)$, exists if and only if $\mathcal{N}$ is $(2t, t)$-Connected.*

Proof. By invoking Lemma 10 and Lemma 11. □

[13] Notice that $\forall i, j \in \{1, 2, 3, 4\}$ $\mathbb{A}_i \cap \mathbb{A}_j = \emptyset$ and $|\mathbb{A}_i| \neq 0$.

6 The Ugly: When only faulty can authenticate

Lemma 13. *(t,k)-BA Protocol over $\mathcal{N}$, $n \leq (t+k)$, exists if and only if $\mathcal{N}$ is $(2t+1)$-connected.*

Proof. Since $n \leq (t+k)$, it basically means that the signatures schemes of all nodes outside adversary's control can be forged and hence the power of authentication is entailed useless. Hence, proofs from the standard unauthenticated model of BA [Lyn96, Dol82] will lead us here. □

Theorem 14 (Main Theorem)**.** *(t,k)-BA protocol over a graph $\mathcal{N} = (\mathbb{P}, \mathcal{E})$, $|\mathbb{P}| = n$, exists if and only if $n > 2t+ min(t,k)$ and $\mathcal{N}$ is*

$$\begin{array}{rlrcll} (t+1)\text{-connected} & if & & & n & > (2t+k) \\ (2t,t)\text{-connected} & if & (t+k) & < & n & \leq (2t+k) \\ (2t+1)\text{-connected} & if & & & n & \leq (t+k) \end{array}$$

Proof. By invoking Lemma 1, Lemma 12 and Lemma 13 and the result of Gupta *et al.* [GGBS10], we establish the theorem. □

7 Concluding Remarks

Possibly, for the first time in literature we show that there are graphs over which agreement is possible even though not all non-faulty nodes can reliably communicate with each other. In essence, *all-node global consistency is strictly easier than all-pairs point-to-point communication.* In this perspective, it appears that the problem of *agreement* could be a more fundamental primitive to general distributed computing than what (even the ubiquitous problem of) reliable communication is.

The focus of this work has been, primarily, to establish (im)possibility results for BA. Hence, the protocols presented in this paper are sub-optimal and there is a definite scope for improving the same. Further, it will be interesting to study this problem in more generic settings such as directed graphs, asynchronous networks and may be under the influence of a non-threshold adversary.

References

[AFM99] Bernd Altmann, Matthias Fitzi, and Ueli M. Maurer. Byzantine agreement secure against general adversaries in the dual failure model. In *Proceedings of the 13th International Symposium on Distributed Computing*, pages 123–137, London, UK, 1999. Springer-Verlag. 2

[Bor95] Malte Borcherding. On the number of authenticated rounds in byzantine agreement. In *WDAG '95: Proceedings of the 9th International Workshop on Distributed Algorithms*, pages 230–241, London, UK, 1995. Springer-Verlag. 2

[Bor96a] Malte Borcherding. Levels of authentication in distributed agreement. In *WDAG '96: Proceedings of the 10th International Workshop on Distributed Algorithms*, pages 40–55, London, UK, 1996. Springer-Verlag. 2

[Bor96b] Malte Borcherding. Partially authenticated algorithms for byzantine agreement. In *ISCA: Proceedings of the 9th International Conference on Parallel and Distributed Computing Systems*, pages 8–11, 1996. 2

[DDS87] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. *J. ACM*, 34(1):77–97, 1987. 2

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *Journal of the Association for Computing Machinery (JACM)*, 40(1):17–47, January 1993. 1, 4

[Dol82] D. Dolev. The Byzantine Generals Strike Again. *Journal of Algorithms*, 3(1):14–30, March 1982. 1, 2, 14

[DS83] D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983. 5

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985. 2

[FM98] Matthias Fitzi and Ueli M. Maurer. Efficient byzantine agreement secure against general adversaries. In *International Symposium on Distributed Computing*, pages 134–148, 1998. 10

[FM00] Mattias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 494–503, New York, NY, USA, 2000. ACM. 2

[Gar94] J. A. Garay. Reaching (and Maintaining) Agreement in the Presence of Mobile Faults. In *Proceedings of the 8th International Workshop on Distributed Algorithms – WDAG '94*, volume 857 of *Lecture Notes in Computer Science (LNCS)*, pages 253–264, 1994. 2

[GGBS10] Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. Authenticated Byzantine Generals in Dual Failure Model. In *ICDCN '10: In proceedings of the 11th International Conference on Distributed Computing and Networking*, Kolkata, India, 2010. 1, 2, 14

[GLR95] L. Gong, P. Lincoln, and J. Rushby. Byzantine agreement with authentication: Observations and applications in tolerating hybrid and link faults, 1995. 2

[HM97] M. Hirt and U. Maurer. Complete Characterization of Adversaries Tolerable in Secure Multi-party Computation. In *Proceedings of the 16th Symposium on Principles of Distributed Computing (PODC)*, pages 25–34. ACM Press, August 1997. 10, 13

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000. 2, 10, 11, 13

[KK07] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. 2007. 2

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. 4

[Lyn96] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Mateo, CA, USA, 1996. 2, 13, 14

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. 1, 2, 4

[Rab83] M. O. Rabin. Randomized byzantine generals. In *Proc. of the 24th Annu. IEEE Symp. on Foundations of Computer Science*, pages 403–409, 1983. 2

[ST87] T. K. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987. 2