Authenticated Byzantine Generals Strike Again

Anuj Gupta Prasant Gopal Piyush Bansal Kannan Srinathan
Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad, India
{anujgupta@research. prasant@research. piyush_bansal@research. srinathan@}iiit.ac.in

Abstract

Pease *et al.* introduced the problem of *Authenticated Byzantine Generals* (ABG) where players could use digital signatures (or similar tools) to thwart the challenge posed by Byzantine faults in distributed protocols for agreement. Subsequently it is well known that ABG among n players tolerating up to t faults is (efficiently) possible if and only if $n > t$ (which is a huge improvement over the $n > 3t$ condition in the absence of authentication for the same functionality). We initiate a study of ABG in a mixed adversary model where the adversary can corrupt up to any t_b players actively and control up to an other t_p players passively. We prove that ABG over a completely connected synchronous network of n nodes tolerating a (t_b,t_p)-adversary is possible if and only if $n > 2t_b$+min(t_b, t_p) when $t_p > 0$. For the case of t_p=0 and t_b=t, the existing result of $n > t$ holds. Our work attempts to unify the literature on Byzantine Generals Problems and Authenticated Byzantine Generals.

Keywords. Broadcast, Authenticated Byzantine General, Mixed adversary.

1 Introduction

Designing protocols for simulating a broadcast channel over a point to point network in presence of faults is a fundamental problem in theory of distributed computing. The problem is popularly referred to as the "Byzantine Generals problem"(BGP), introduced by Lamport *et al.* [18]. Informally, the challenge is to maintain a coherent view of the world among all the non-faulty players in spite of faulty players trying to disrupt the same. Specifically, in a protocol for BGP over a *synchronous* network of n players, the *General* starts with an input from a fixed set $V = \{0, 1\}$. At the end of the protocol (which may involve finitely many rounds of interaction), even if up to any t of the n players are faulty, all non-faulty players output the same value $u \in V$ and if the General is non-faulty and starts with input $v \in V$, then $u = v$. Over a completely connected synchronous network with no additional setup, classical results of [18, 21] show that reliable broadcast among n parties in presence of up to t number of malicious players is achievable if and only if $t < n/3$. Here a player is said to be non-faulty if and only if he faithfully executes the protocol delegated to him. Traditionally, the notion of failures in the system is captured via a fictitious entity called *adversary* that may control a subset of players. An adversary that controls up to any t of the n players is denoted by t-adversary. Note that, in the context of BGP, not all players under the control of the adversary need to be faulty. This is because the adversary may choose to *passively* control some of the players who, by virtue of correctly following the protocol, are non-faulty.

There exists a rich literature on the problem of BGP. After [18, 21], studies were initiated under various settings like asynchronous networks [11], partially synchronous networks [9], incomplete networks [8], hypernetworks [13], non-threshold adversaries [12], mixed-adversaries [1], mobile adversaries [14], and probabilistic correctness [22] to name a few. An important variant of BGP is the authenticated model proposed by Pease *et al.* [21], which as the title of this report suggests, is our main focus. In this model, which we hereafter refer to as *authenticated Byzantine General* (ABG), the players are supplemented with "magical" powers (say a Public Key Infrastructure (PKI) and digital signatures) using which the players can authenticate themselves and their messages. It is proved that in such a model, the tolerability against a t-adversary can be amazingly increased to as high as $t < n$. Dolev and Strong [7] presented efficient protocols thereby confirming the usefulness of authentication in both possibility as well as feasibility of distributed protocols. Subsequent papers on this subject include [3, 5, 24, 4, 17, 16, 23]. In essence, the state-of-the-art in ABG can be summarized by the following folklore (as noted by Nancy Lynch [20, page 116] too): "Protocols for agreement tolerating a *fail-stop* t-adversary, modified so that all messages are signed and only correctly signed messages are accepted, solve the agreement problem for the authenticated Byzantine fault model".

A large part of literature in the area of fault tolerant distributed computing considers adversary to have same amount of control over all the corrupt players. Mixed adversary model is motivated from a scenario where adversary has varied control over different corrupt players i.e. it controls some players passively, some others actively, another fraction as fail-stop and so on. Note that mixed adversary model not only generalizes the adversary models where only one type of corruption is considered but also facilitates a better understanding of the computability/complexity of the task at hand as a function of the adversary's power. With respect to BGP, mixed adversary model has been considered in the past, [15, 1] to name a few. Motivated from this, we initiate the study of ABG under the influence of a (t_b,t_p)-adversary where the adversary can corrupt up to any t_b players actively and control up to another t_p players passively. Adversary can make the actively corrupt players to behave in arbitrary manner and can read the internal state of the passively corrupt players.

2 Our Contributions and Results

The first contribution of this work is to argue that for the case of (t_b,t_p)-adversary, the problem definition of ABG itself needs to be modified. Note that any solution to the problem of ABG aims to simulate a broadcast channel over a point to point network. We show that a ABG protocol that does not facilitate the passivley corrupt players to reach an agreement along with honest players, *does not truely* simulate a broadcast channel, as originally intended.

From the result of $n > t$ [21], one might feel that in the presence of (t_b,t_p)-adversary, $n > t_b$ or $n > t_b$+t_p may be sufficient for possibility of ABG_{mix}. However we show that this is not the case, and neither of $n > t_b$ or $n > t_b$+t_p is sufficient to simulate a broadcast channel. We support our claim by studying a simple synchronous system consisting of three players (as illustrated by network $\mathcal{N}$ in Figure 1). In Section 5, we prove that *there does not exist any protocol guarantying consistency among the outputs of all the non-faulty players* over $\mathcal{N}$ tolerating a (1,1)-adversary. As a second contribution of this work we prove that over a completely connected synchronous graph of n nodes, ABG_{mix} tolerating a (t_b,t_p)-adversary is possible if and only if $n > 2t_b$+min(t_b,t_p), $t_p > 0$ (which explains our discussion as to why a $(1, 1)$-adversary is not tolerable over complete graph of three nodes). For the case of t_p=0 and t_b=t, the existing result of $n > t$ [21] holds. Further, note that the solution to the problem of ABG under influence of a (t_b,t_p)-adversary answers the question of simulating a broadcast channel for the entire gamut of adversaries between $t_b = t$ & $t_p = 0$ (ABG) and $t_b = t$ & $t_p = n - t$ (BGP). Thus, last but not least, our results attempt to *unify* the literature on ABG and BGP.

3 Our Model

We consider a set of n players, fully connected, denoted by $\mathbb{P}$. Communication over the network is assumed to be synchronous. That is, the protocol is executed in a sequence of *rounds* where in each round, a player can perform some local computation, send new messages to all the players, receive messages sent to him by players in the same round, (and if necessary perform some more local computation), in that order. During the execution, the adversary may corrupt up to any t_b+t_p players. Adversary can make t_b players to behave in any arbitrary fashion and read the internal states of another t_p players. We refer to such an adversary as (t_b,t_p)-adversary. W.l.o.g we assume that adversary always uses his full power, and hence $t_b \cap t_p = \emptyset$. We further assume that the communication channel between any two players is perfectly reliable and authenticated i.e. adversary cannot modify messages sent between non-malicious parties. We also assume existence of a (signature/authentication) scheme where the sender signs the message to be sent. This is modeled by all players also having an additional setup-tape that is generated during the preprocessing phase.[1] Typically in such a preprocessing phase, the signature keys are generated. That is, each player gets its own private key, and in addition, public verification keys for all other players. No player can forge any other player's signature and the receiver of a message can uniquely identify the sender of the message from the signature. However, adversary can forge the signature of all the (t_b+t_p) players under its control.

Organization of the report: In section 4, we argue as to why for mixed adversary one need to modify the definition of ABG. We follow it up by giving a motivating example in section 5, to study the problem of ABG under the influence of a mixed adversary. In section 6, we present rigorous definition for some of the terms used in the proofs. Section 7 focuses on the complete characterization of ABG_{mix}, followed by conclusion in section 8.

4 Quest for a Better Definition of ABG

We now present an argument debating as to why under the influence of mixed adversary the extant ABG defintion used in the literature does not suffice. As a preclude, we remark that literature considers a player to be *faulty* if and only if that player deviates from the designated protocol. Consequently, a player can be non-faulty in two ways – first the adversary is absent and (therefore) player follows the protocol and second the adversary is present passively and (therefore) player follows the protocol. (For the rest of this work we refer to the former kind of non-faulty player as *honest* and the latter as *passively corrupt.*)

Consider the following scenario: Given a physical broadcast channel among a set of n players, where a (t_b,t_p)-adversary corrupts t_b players actively and another t_p players passively. The General sends his input value $v \in V = \{0, 1\}$ through this physical broadcast channel. Then all the n players are guaranteed to receive value v. All the honest players will output v. By virtue of correctly following the protocol, all the passively corrupt players will also output v. Thus all non-faulty will output same value v. Adversary may make all the faulty (actively corrupt) players to output a value different from v.[2]

Preceding paragraph necessitates the following finding: any protocol aiming to *truly simulate* a broadcast channel in the presence of (t_b,t_p)-adversary, has to *ensure* that all non-faulty players (honest and passively corrupt, i.e. $n - t_b$) output *same* value. It is evident that a protocol for ABG that does not facilitate passively controlled players to agree too, does not *truly* simulate a broadcast channel as originally intended. Hereafter, we refer to the problem of ABG under the influence mixed adversary as ABG_{mix} and formally define the same in section 3. Based on the above discussion, we now formally define ABG_{mix}:

Definition 1 **(ABG_{mix})** *Given a set of n players $\mathbb{P}$=$\{p_1,p_2,\ldots,p_n\}$, a finite domain $V = \{0, 1\}$, and a predesignated player as General $\mathcal{G}$. $\mathcal{G}$ holds an input value $v \in V$. A protocol η among $\mathbb{P}$ solves ABG_{mix}, tolerating (t_b,t_p)-adversary, if for any t_b, t_p out of n, any $\mathbb{P}$ and V, at the end of the protocol the following three properties hold:*

[1]Note that keys cannot be generated with the system itself. It is assumed that the keys are generated using a trusted system and distributed to players prior to running of the protocol similar to [19].

[2]A similar argument can be given using a TTP (Trusted Third Party) [6]. The General sends his value v to TTP. TTP forwards it to all the players. All non-faulty players in the ideal process output v.

- Agreement: *All non-faulty players decide on the same value $u \in V$.*
- Validity: *If $\mathcal{G}$ is non-faulty and starts with the initial value $v \in V$, then $u = v$.*
- Termination: *All non-faulty players eventually decide.*

For a better clarity, we reiterate certain terms that have been used extensively in this work. A player is said to be *faulty* if and only if he deviates from the designated protocol. Consequently a *non-faulty* player is one who does not deviate from the designated protocol. Note that adversary may have certain amount of access to some(or all) non-faulty players such as reading their internal state. A *passively corrupt* player is one who follows the designated protocol diligently, but adversary has a complete (read only)access to his internal state. An *honest* player is one who follows the designated protocol, and over whom adversary has absolutely no control. For the purpose of this report, both *honest* and *passively corrupt* players are together referred as non-faulty players. The main aim of this work is to answer the following question: *when is it possible to design protocols for solving ABG_{mix} over completely connected synchronous networks, tolerating a (t_b,t_p)-adversary?*

5 Motivating Example

As a motivating example to study ABG_{mix} in presence of (t_b,t_p)-adversary, we show that there does not exists any protocol solving ABG_{mix}, tolerating $(1, 1)$-adversary, over a complete network $\mathcal{N}$ of three players, $\mathbb{P} = \{A, B, C\}$(as illustrated in Figure 1). This implies that neither $n > t_b$ nor $n > t_b + t_p$ is sufficient for solving ABG_{mix}. The proof for impossibility of any protocol is motivated from the work of Fischer *et al.* [10]. Here we only give a proof sketch, a detailed formal proof of the same is presented in Section 6.

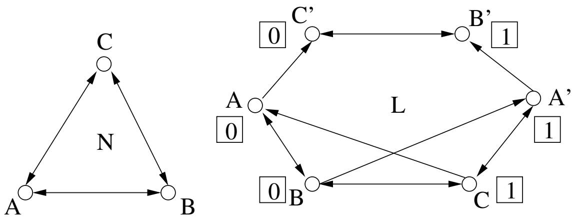

Figure 1: Network $\mathcal{N}$ and System L.

Theorem 1 *There does not exist any protocol that solves ABG_{mix} tolerating a $(1, 1)$-adversary over a completely connected synchronous network $\mathcal{N}$ of 3 nodes.*

Proof: We assume there exists a protocol Π that solves ABG_{mix} tolerating a $(1, 1)$-adversary over a completely connected network $\mathcal{N}$(as shown in Figure 1) of 3 nodes. Using Π we create a protocol π'[Definition 2] in such a way that if Π exists then so does π'(Lemma 2). Using two copies of π' we construct a system L, as shown in Figure 1. We then show that L must exhibit a contradictory behavior. It follows that the assumed protocol Π cannot exist.

Formally, L is a synchronous system with a well defined behavior. That is, system L has a well defined output distribution for any particular input assignment. We show that for a particular input assignment, no such well defined behavior is possible. Further, no player in L knows the complete system. Each player in aware of only his immediate neighbors. In reality a player may be connected to either a or a', but it cannot differentiate between the two. It knows its neighbor only by its local name which may be a. Further, in-neighborhood of any node a(or a') in L is same as in-neighborhood of the corresponding node a in $\mathcal{N}$.

Let α_1, α_2 and α_3 be three scenarios in an execution of Π. In α_1, A is the General starting with input 0. Adversary $\mathcal{A}$ corrupts C actively and controls A passively. In α_2, A is the General. $\mathcal{A}$ corrupts A and makes him to interact with B as if A started with input 0, and, interact with C as if A started with input 1. In α_3, A is the General starting with input 1. $\mathcal{A}$ corrupts B actively and controls A passively.

Further, let α be an execution of L where each player starts with input value as shown in Figure 1. All the players in α are honest and follow the designated protocol correctly.

We claim that no matter for how many rounds Π executes, for any round i, $\mathcal{A}$ can ensure that whatever *view* (informally view of a player means all the messages the player ever gets to see during the entire protocol execution. We formally define view in section 6) A, B get in α, $\mathcal{A}$ can generate the same view for A, B in α_1. This implies that the player A cannot ever differentiate between α_1 and α (dubbed $\alpha_1 \overset{A}{\sim} \alpha$). Similarly, player B cannot ever differentiate between α_1 and α ($\alpha_1 \overset{B}{\sim} \alpha$). Since Π solves ABG_{mix}, from the definition of ABG_{mix} [Definition 1], in α_1, both A, B should decide on value 0. Since view of A, B is same in α_1 and α, both A, B in α will also decide on value 0 (We are able to make claims regarding the outputs of A and B in α as their views in α are same as those in α_1. Thus by analyzing their outputs in α_1, we can determine there outputs in α.). Similarly, $\mathcal{A}$ can ensure that $\alpha_3 \overset{A'}{\sim} \alpha$ and $\alpha_3 \overset{C}{\sim} \alpha$. As per definition of ABG_{mix}, both A, C in α_3 should decide on value 1. Then so will both A', C in α. Similarly, we claim that $\mathcal{A}$ can ensure that $\alpha_2 \overset{B}{\sim} \alpha$ and $\alpha_2 \overset{C}{\sim} \alpha$. As per the definition of ABG_{mix}, B, C in α_2 should agree on same value, then so should B, C in α. But B,C in α have already decided upon values 0 and 1 respectively. This implies L must exhibit contradictory behavior.

To complete the proof sketch, we now give an idea as to how $\mathcal{A}$ can ensure that A, B get the same view in α_1 and α. Consider an execution Γ of L which is exactly same as α except that in Γ A' starts with input value 0. Since in Γ, no message from B' or C' can ever reach any of A,B,C or A', $\mathcal{A}$ can ensure that A and B get same messages in Γ and α_1 (all $\mathcal{A}$ has to do is to let C follow the designated protocol with input value 1). Now in α, all messages received by A and B respectively are same as those in Γ except those messages that have been processed by A' at least once(since A' in Γ starts with input value 0 where as A' in α starts with input value 1). If in α_1, $\mathcal{A}$ can simulate this difference between α and Γ, we can say that $\mathcal{A}$ can make view of A, B same in α and α_1. We now claim that for any round i, $i \geq 1$, it is always possible for $\mathcal{A}$ to do so. Note that owing to the typical construction of L, in α, A' can send a message to A or B only via C. This ensures that in α, any message from A' can reach A or B only after it has been processed by C. Now in α_1, C is faulty and $\mathcal{A}$ controls A passively. Thus whatever C sends to A and B in α, $\mathcal{A}$ can send the same to A and B in α_1. Similarly one can prove that whatever view B, C get in α, $\mathcal{A}$ can generate the same view for B, C in α_2 and whatever view C, A' get in α, $\mathcal{A}$ can generate the same view for C, A in α_3. ■

To complete the proof of theorem 1, we now define the protocol π'[Definition 2] and show that if Π exists then so does π'(Lemma 2).

Definition 2 (π') *Any statement in Π of the kind "b sends message m to a" is replaced by "b multicasts message m to all instances of a"(i.e. a,a') [3] in π'. Similarly any statement of the kind "c sends message m to a" in Π is replaced by "c multicasts message m to all instances of a" in π'. Rest all statements in π' are same as those in Π.*

Lemma 2 *If Π exists then π' exists.*

Proof: Implied from Definition 2. ■

Formally, one can prove the following lemma. Here $view_Z^{\phi}$ represents view of player Z during entire execution ϕ. Detailed proof of the same is given in Section 6.

Lemma 3 *Adversary can ensure the following:*
$view_A^{\alpha} \sim view_A^{\alpha_1}$ *and* $view_B^{\alpha} \sim view_B^{\alpha_1}$.

[3] a and a' are independent copies of a with same authentication key.

$view_B^{\alpha} \sim view_B^{\alpha_2}$ *and* $view_C^{\alpha} \sim view_C^{\alpha_2}$.
$view_{A'}^{\alpha} \sim view_A^{\alpha_3}$ *and* $view_C^{\alpha} \sim view_C^{\alpha_3}$.

6 Motivating Example: Detailed Proofs

As a prelude to detailed proof of Lemma 3, we mathematically define the term *view*. Intuitively, by *view* we want to capture all that a player ever gets to see during the entire execution of the protocol. Thus, the view of a player is formed by all the messages it ever sends and receives during the execution of the protocol. Let $msg_i^{\Omega}(a,b)_a$ denote the message sent by player a to player b in i^{th} round of execution Ω. The subscript a represents the last player who authenticated the message. W.l.o.g we assume that players always authenticate the message before sending.

Then view of a player a during execution Ω at the end of round i, denoted by $view_{a,i}^{\Omega}$, can be represented as a collection of all the messages it ever sends and receives. Formally:

$$view_{a,i}^{\Omega} = \bigcup_k (msg_k^{\Omega}(a,x)_a, msg_k^{\Omega}(x,a)_x),\ \forall k \in \{1 \dots i\},\ \forall x \in \mathbb{P} \tag{1}$$

The messages sent by player a in any round i of some execution say Ω depends on 4 parameters: input value with which a starts, secret key used by a for authentication, code(say π) being executed by a, and messages received by a up to round $i-1$ of Ω. Since the outgoing messages are a function of incoming messages, we can rewrite the equation 1 as:

$$view_{a,i}^{\Omega} = \bigcup_k (msg_k^{\Omega}(x,a)_x),\ \forall k \in \{1 \dots i\},\ \forall x \in \mathbb{P} \tag{2}$$

In order to show that the views of 2 different players a, b running in 2 different executions Ω, Γ respectively till round i are same, we use the following fact: If both players a, b start with same input, use the same secret key and run similar code [4], and if for every round $1 \dots i$ their corresponding incoming messages are same, then their views till round i will also be same [5]. Formally:

$$view_{a,k}^{\Omega} \sim view_{b,k}^{\Gamma},\ \text{iff},\ msg_k^{\Omega}(x,a) \sim msg_k^{\Gamma}(x,b),\ \forall k \in (1 \dots i),\ \forall x \in \mathbb{P} \tag{3}$$

In order to show that player A gets same view in α and α_1, we need to show is that whatever messages A receives from B, C in α, $\mathcal{A}$ can always ensure that A gets the same messages from B, C in α_1 too. Similarly, to show that B gets same view in α and α_1, one needs to prove that whatever messages B receives from A, C in α, $\mathcal{A}$ can always ensure that B gets the same messages in α_1 too. Our technique to show the same is as follows – note that what node A receives in round i of α(or α_1) depends on what nodes B and C send to it in round i of α(or α_1). So we need to argue that these messages sent in round i of α and α_1 respectively are same or *can be* made same by the adversary. Now the messages B,C send in round i of α and α_1 depend on what they them self receive in previous round $i-1$. This in turn depends on what A, C(or A, B) send to B(or C) in round $i-2$ of α and α_1 respectively. Thus we need to argue that adversary can ensure that whatever messages A, C(or A, B) send to B(or C) in round $i-2$ of α is same as whatever messages A, C(or A, B) send to B(or C) in round $i-2$ of α_1. Note that this continues in a recursive manner until recursion stops at round 1. The entire recursion can be visualized as trees T_{α}^{A} and $T_{\alpha_1}^{A}$ rooted at A for executions α and α_1 respectively as shown in Figure 2.

[4]Note that a, b may even run different codes say θ and θ', however message generated for a given player say c by θ for a given input $\mathcal{I}$ should be same as message generated for c by θ' for same input $\mathcal{I}$. For our proof Π and π' are similar in this respect, see definition 2

[5] [10] captured this via **Locality Axiom**. In ABG_{mix} a player may also use its private key to determine the outgoing messages. Thus in case of ABG_{mix}, both players having same secret key is must.

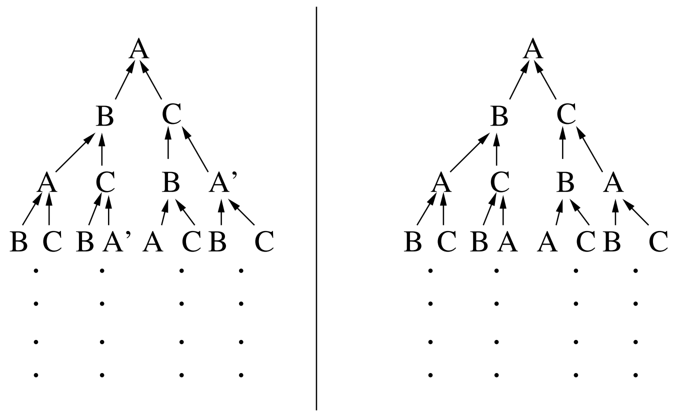

Figure 2: T_{α}^{A} and $T_{\alpha_1}^{A}$

We refer to these trees as *execution trees.* We now formally describe an execution tree T_{α}^{x}. We name the levels of tree in a bottom up manner. Let the lowest level of tree be 1, next level be 2 and so on. An edge from a node y at level j to another node z at level $j+1$ in the tree represents the message that y sends to z in round j of α. All edges are directed from child to parent and are between adjacent levels only.

For our proof to go through, we require the in-degree for any node y'(or y) in T_{α}^{x} to be same as in-degree of corresponding node y in $T_{\alpha_1}^{x}$. Also, if a node z at level $j+1$ has an incoming edge from node y at level j in $T_{\alpha_1}^{x}$, then correspondingly in T_{α}^{x} node z(or z') at level $j+1$ will also have an incoming edge from node y(or y') at level j. The above two points ensure that structurally both the trees $T_{\alpha}^{x'}$(or T_{α}^{x}) and $T_{\alpha_1}^{x}$ will be exactly same (a node y' in T_{α}^{x} is replaced by its corresponding node y in $T_{\alpha_1}^{x}$). Now consider some node, say b'(or b) at level j in T_{α}^{x}. Then its corresponding node at level j in $T_{\alpha_1}^{x}$ is b. Note that if the messages received by b'(or b) in T_{α}^{x} is same as those received by b in $T_{\alpha_1}^{x}$ and both b'(or b) and b start with same input value, same private key and run similar codes [6], then both will send same messages to their respective parents in their respective execution trees. One can then use induction on heights of executions trees, say T_{α}^{A} and $T_{\alpha_1}^{A}$, to argue that for any round i, A receives same messages in α and α_1.

For scenario α_1, we now specify the behavior of the adversary:

1. *Send outgoing messages of round i:* Based on the messages received during round $i-1$, $\mathcal{A}$ decides on the messages to be sent in round i. For round 1, $\mathcal{A}$ sends to B what an honest C would have sent to B in execution α_2. For $i \geq 2$, $\mathcal{A}$ authenticates $msg_{i-1}^{\alpha_1}(B,C)_B$ using C's key and sends it to A. For $msg_{i-1}^{\alpha_1}(A,C)_A$, $\mathcal{A}$ examines the message. If the message has not been authenticated by B even once, it implies that the message has not yet been seen by B. Then $\mathcal{A}$ authenticates and sends same message to B as C would have sent to B in round i of execution α_2. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\alpha_1}(A,C)_A$,($\mathcal{A}$ can construct $msg_{i-1}^{\alpha_1}(A,C)_A$, since it passively controls A and has messages received by A in previous rounds.) such that $msg_{i-1}^{\alpha_1}(A,C)_A \sim msg_{i-1}^{\alpha_2}(A,C)_A$, authenticates it using C's key and sends it to B. If the message has been authenticated by B even once, $\mathcal{A}$ simply authenticates $msg_{i-1}^{\alpha_1}(A,C)_A$ using C's key and sends it to B.

2. *Receive incoming messages of round i:* $\mathcal{A}$ obtains messages $msg_{i}^{\alpha_1}(A,C)_A$ and $msg_{i}^{\alpha_1}(B,C)_B$ via C. (These are round i messages sent by A and B respectively to C). Similarly via A, $\mathcal{A}$ obtains messages $msg_{i}^{\alpha_1}(B,A)_B$ and $msg_{i}^{\alpha_1}(C,A)_C$. (These are round i messages sent by B and C respectively to

[6]Refer to footnote 4.

A. Players respectively compute these messages according to their input, secret key, protocol run by them and the view they get up to round $i-1$).

Consider execution α from the perspective of A and B. We now show that messages received by A and B in round i of α are same as messages received by A and B respectively in round i of α_1.

Lemma 4 *$msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$ and $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$.*

We argue for $msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$. Argument for $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$ follows similarly. To prove that for any round i, A gets same messages in α and α_1, we use induction on height of T_{α}^{A} and $T_{\alpha_1}^{A}$(as shown in Figure 2). Only nodes present in T_{α}^{A} are A, B, C, A'. Corresponding nodes present in $T_{\alpha_1}^{A}$ are A, B, C, A respectively. Notice that since B' does not appear in T_{α}^{A}, any A' in T_{α}^{A} has an outgoing directed edge only and only to C. Similarly, since C' does not appear in T_{α}^{A}, any A in T_{α}^{A} has an outgoing directed edge only and only to B.

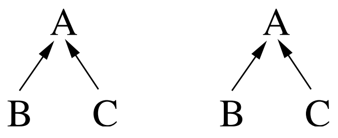

Figure 3: T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of round 1.

We analyze the executions trees T_{α}^{A} and $T_{\alpha_1}^{A}$ in a bottom up manner. Consider round 1 of executions α and α_1. Consider trees T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of round 1 as shown in Figure 3. We claim that A in α and α_1 receive similar messages at the end of round 1. B starts with same input, secret key and executes same code in α and α_1. Thus it will send same messages to A in round 1 of α and α_1 i.e. $msg_1^{\alpha}(B,A)_B \sim msg_1^{\alpha_1}(B,A)_B$. Using aforementioned adversary for α_1, $\mathcal{A}$ can ensure that $msg_1^{\alpha}(C,A)_C \sim msg_1^{\alpha_1}(C,A)_C$. Thus A gets same messages at the end of round 1 in α and α_1.

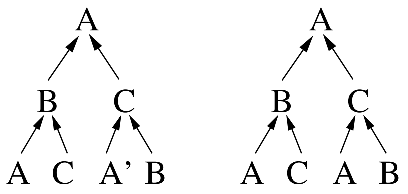

Figure 4: T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of round 2.

We now claim that the similarity holds in round 2 as well i.e. $msg_2^{\alpha}(x,A)_x \sim msg_2^{\alpha_1}(x,A)_x$. Consider trees T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of round 2 as shown in Figure 4. Node A starts with same input value, secret key and execute same code in both α and α_1 respectively, thus $msg_1^{\alpha}(A,B)_A \sim msg_1^{\alpha_1}(A,B)_A$. Similarly, since node B starts with same input value, secret key and execute same code in both α and α_1 respectively, we have $msg_1^{\alpha}(B,C)_B \sim msg_1^{\alpha_1}(B,C)_B$. From the aforementioned adversary for α_1, $\mathcal{A}$ can ensure that $msg_1^{\alpha}(C,B)_C \sim msg_1^{\alpha_1}(C,B)_C$. Now A and A' start with different inputs thus send different messages to C in round 1. However, since A is passively corrupt and C is Byzantine in α_1, $\mathcal{A}$ can construct message $msg_1^{\alpha_1}(A,C)_A$ such that $msg_1^{\alpha_1}(A,C)_A \sim msg_1^{\alpha}(A',C)_A$. Thus C can simulate to receive messages in α_1 same as those in α at the end of round 1. Now B receives same messages in α and α_1 and has same input value, secret key and executes same code, thus $msg_2^{\alpha}(B,A)_B \sim msg_2^{\alpha_1}(B,A)_B$. From the aforementioned adversary, $\mathcal{A}$ can ensure that $msg_2^{\alpha}(C,A)_C \sim msg_2^{\alpha_1}(C,A)_C$. Thus $msg_2^{\alpha}(x,A)_x \sim msg_2^{\alpha_1}(x,A)_x$, $\forall x \in \mathbb{P}$ holds.

Let the similarity be true till some round k i.e. $msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T_{α}^{A} and $T_{\alpha_1}^{A}$

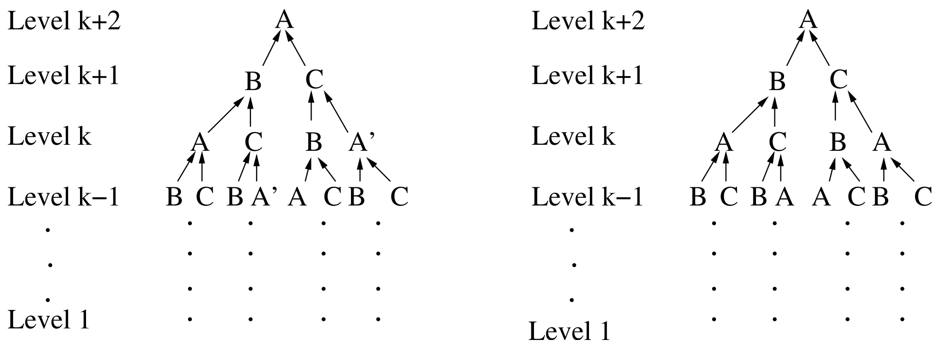

Figure 5: T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of $k+1$ rounds.

at the end of $k+1$ rounds as shown in Figure 5. For proving the induction step, we need to show that A at level $k+2$ receives same messages in both trees. Consider edges between level k and $k+1$. From induction hypothesis any node A up to level $k+1$ receives same messages in T_{α}^{A} and $T_{\alpha_1}^{A}$. Since A starts with same input value, secret key and executes same code in both α and α_1 respectively, thus will send same messages in round k i.e. $msg_k^{\alpha}(A,B)_A \sim msg_k^{\alpha_1}(A,B)_A$. Similarly one can argue that $msg_k^{\alpha}(B,C)_B \sim msg_k^{\alpha_1}(B,C)_B$. This is because from the induction hypothesis step on heights of T_{α}^{B} and $T_{\alpha_1}^{B}$, one gets $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. Now consider A' at level k in in T_{α}^{A} and corresponding A at level k in in $T_{\alpha_1}^{A}$. For time being assume A' up to level k in T_{α}^{A} receives same messages as corresponding A in $T_{\alpha_1}^{A}$. Since A' start with different input from A, they send different messages to C in round k. We now claim that $\mathcal{A}$ can ensure that C at level $k+1$ in $T_{\alpha_1}^{A}$ can simulate to receive same message from A' as C at level $k+1$ in T_{α}^{A}. This is because $\mathcal{A}$ controls A passively in α_1, thus can construct messages on behalf of A in α_1. Formally $\mathcal{A}$ can construct $msg_k^{\alpha_1}(A',C)_{A'}$ such that $msg_k^{\alpha_1}(A',C)_{A'} \sim msg_k^{\alpha}(A,C)_A$. Thus C a level $k+1$ receives same messages in both trees. Similarly one can argue that C at level k receives same messages in T_{α}^{A} and $T_{\alpha_1}^{A}$. Since C starts with same input value, secret key and executes same code in both α and α_1 respectively, thus it will send same messages in round $k+1$ to A i.e. $msg_{k+1}^{\alpha_1}(C,A)_C \sim msg_{k+1}^{\alpha}(C,A)_C$. Similarly one can argue that $msg_{k+1}^{\alpha_1}(B,A)_B \sim msg_{k+1}^{\alpha}(B,A)_B$. Thus induction holds for round $k+1$ too. The proof is based on a assumption that A' at level k in T_{α}^{A} receives same messages as corresponding A in $T_{\alpha_1}^{A}$. Note that A' in T_{α}^{A} and A in $T_{\alpha_1}^{A}$ receives messages from B and C. Using induction and arguments similar to those given above one can show that such an assumption indeed holds true. Thus $msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ holds true. ■

Lemma 5 *$view_A^{\alpha} \sim view_A^{\alpha_1}$ and $view_B^{\alpha} \sim view_B^{\alpha_1}$*

Proof: Follows from equation 3 and Lemma 4. ■

Adversary for α_2:

1. *Send outgoing messages of round i:* Based on the messages received during round $i-1$, $\mathcal{A}$ decides on the messages to be sent in round i. For round 1, $\mathcal{A}$ sends to B what an honest A would have sent to B in execution α_1. Similarly $\mathcal{A}$ sends to C what an honest A would have sent to C in execution α_3. For $i \geq 2$, $\mathcal{A}$ examines the message $msg_{i-1}^{\alpha_2}(C,A)_C$. If the message has not been authenticated by B even once, $\mathcal{A}$ authenticates and sends same message to B as A would have sent to B in round i of execution α_1. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\alpha_2}(C,A)_C$,($\mathcal{A}$ can construct $msg_{i-1}^{\alpha_2}(C,A)_C$, since it passively controls C and has messages received by C in previous round.) such that $msg_{i-1}^{\alpha_2}(C,A)_C \sim msg_{i-1}^{\alpha_1}(C,A)_C$, authenticates it using A's key and sends it to B. If the message has been authenticated by B even once, $\mathcal{A}$ simply authenticates $msg_{i-1}^{\alpha_2}(C,A)_C$ using A's key and sends it to B. Similarly $\mathcal{A}$ authenticates $msg_{i-1}^{\alpha_2}(B,A)_B$ using A's key and sends it to C.

2. *Receive incoming messages of round i:* $\mathcal{A}$ obtains messages $msg_i^{\alpha_2}(C,A)_C$ and $msg_i^{\alpha_2}(B,A)_B$ via A. (These are round i messages in α_2 sent by C and B respectively to A). Similarly via C, $\mathcal{A}$ obtains messages $msg_i^{\alpha_2}(A,C)_A$ and $msg_i^{\alpha_2}(B,C)_B$ in α_2. (These are also round i messages sent by A and B respectively to C. Players respectively compute these messages according to their input, secret key, protocol run by them and the view they get up to round $i-1$).

Consider execution α from the perspective of B and C. We now show that messages received by B and C in round i of α are same as messages received by B and C respectively in round i of α_2. The central idea is similar to proof of Lemma 4.

Lemma 6 *$msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$ and $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i>0$, $\forall x \in \mathbb{P}$*

Proof: We show that for any round i, adversary can ensure that B receives same messages in α and α_2 i.e. $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$, $\forall i>0$, $\forall x \in \mathbb{P}$. Argument for $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i>0$, $\forall x \in \mathbb{P}$ follows similarly. We prove the same using induction on height of T_{α}^{B} and $T_{\alpha_2}^{B}$(as shown in Figure 8). Note that only nodes present in T_{α}^{B} are A,B,C,A'. Corresponding nodes present in $T_{\alpha_2}^{B}$ are A,B,C,A respectively. Notice that since B' does not appear in T_{α}^{B}, any A' in T_{α}^{B} has an outgoing directed edge only and only to C. Similarly, since C' does not appear in T_{α}^{B}, any A in T_{α}^{B} has an outgoing directed edge only and only to B.

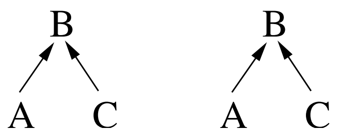

Figure 6: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 1.

We begin analyzing the executions trees T_{α}^{B} and $T_{\alpha_2}^{B}$ in a bottom up manner. Consider trees T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 1 as shown in Figure 3. C starts with same input, secret key and executes same code in α and α_2. Thus it will send same messages to B in round 1 of α and α_2 i.e. $msg_1^{\alpha}(C,B)_C \sim msg_1^{\alpha_2}(C,B)_C$. Since A is faulty in α_2, $\mathcal{A}$ can ensure that $msg_1^{\alpha}(A,B)_B \sim msg_1^{\alpha_2}(A,B)_B$. Thus B gets same messages at the end of round 1 in α and α_2 i.e. $msg_1^{\alpha}(x,B)_x \sim msg_1^{\alpha_2}(x,B)_x$, $\forall x \in \mathbb{P}$.

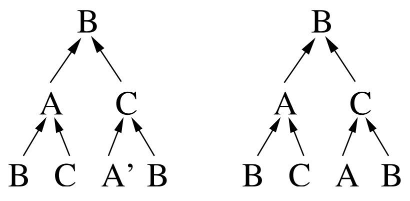

Figure 7: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 2.

We now claim that the similarity holds in round 2 as well i.e. $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_2}(x,B)_x$. Consider trees T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 2 as shown in Figure 7. B starts with same input value, secret key and execute same code in both α and α_2 respectively, thus we have $msg_1^{\alpha}(B,A)_B \sim msg_1^{\alpha_2}(B,A)_B$ and $msg_1^{\alpha}(B,C)_B \sim msg_1^{\alpha_2}(B,C)_B$. Similarly, C starts with same input value, secret key and execute same code in both α and α_2 respectively, thus $msg_1^{\alpha}(C,A)_C \sim msg_1^{\alpha_2}(C,A)_C$. $\mathcal{A}$ can ensure that $msg_1^{\alpha}(A',C)_{A'} \sim msg_1^{\alpha_2}(A,C)_A$. At the end of round 1, A receives same messages in α and α_2 and since it begins with same input value, secret key and executes same code, we have $msg_2^{\alpha}(A,B)_A \sim msg_2^{\alpha_2}(A,B)_A$. Similarly, since C also receives same messages in α and α_2 and it has same input value, secret key and executes same code, thus $msg_2^{\alpha}(C,B)_C \sim msg_2^{\alpha_2}(C,B)_C$. Thus, $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_2}(x,B)_x$, $\forall x \in \mathbb{P}$.

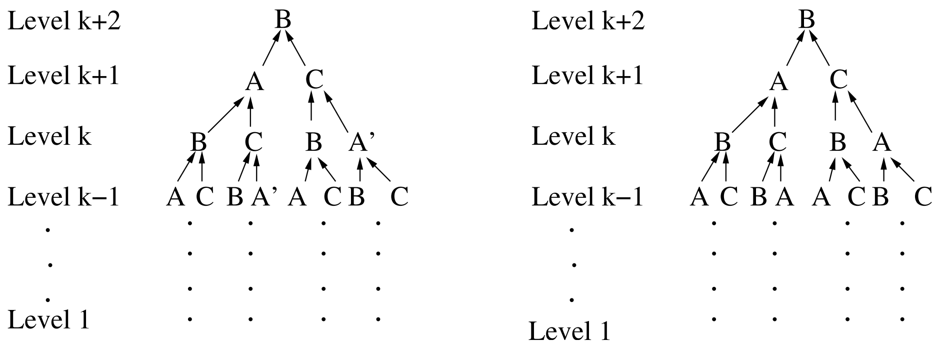

Figure 8: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of $k+1$ rounds.

Let the similarity be true till some round k i.e. $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$, $\forall i|1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of $k+1$ rounds as shown in Figure 8. For proving the induction step, we need to show that B at level $k+2$ receives same messages in both trees. Consider edges between level k and $k+1$. From induction hypothesis any node B up to level k receives same messages in T_{α}^{B} and $T_{\alpha_2}^{B}$. Since B starts with same input value, secret key and executes same code in both α and α_2 respectively, thus will send same messages in round k i.e. $msg_k^{\alpha}(B,A)_B \sim msg_k^{\alpha_2}(B,A)_B$ and $msg_k^{\alpha}(B,C)_B \sim msg_k^{\alpha_2}(B,C)_B$. Similarly one can argue that $msg_k^{\alpha}(C,A)_C \sim msg_k^{\alpha_2}(C,A)_C$. This is because from the induction hypothesis step on heights of T_{α}^{C} and $T_{\alpha_2}^{C}$, one gets $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i|1 \leq i \leq k$, $\forall x \in \mathbb{P}$. Thus A at level $k+1$ receives same messages in both T_{α}^{B} and $T_{\alpha_2}^{B}$. Since A starts with same input value, secret key and executes same code in both α and α_2 respectively, one gets $msg_{k+1}^{\alpha}(A,B)_A \sim msg_{k+1}^{\alpha_2}(A,B)_A$. Now consider A' at level k in in T_{α}^{B} and corresponding A at level k in in $T_{\alpha_2}^{B}$. For time being assume A' up to level k in T_{α}^{B} receives same messages as corresponding A in $T_{\alpha_2}^{B}$. Since A is corrupt in α_2, $\mathcal{A}$ can always ensure that in round k of α_2, A sends to C what A' sends to C in round k of α. One gets $msg_{k+1}^{\alpha}(C,B)_C \sim msg_{k+1}^{\alpha_2}(C,B)_C$. Thus induction holds for round $k+1$ too. The proof is based on a assumption that A' at level k in T_{α}^{B} receives same messages as corresponding A in $T_{\alpha_2}^{B}$. Note that A' in T_{α}^{B} and A in $T_{\alpha_2}^{B}$ receives messages from B and C. Using induction and arguments similar to those given above one can show that such an assumption indeed holds true. Thus $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ holds true. ■

Lemma 7 *$view_B^{\alpha} \sim view_B^{\alpha_2}$ and $view_C^{\alpha} \sim view_C^{\alpha_2}$*

Proof: Follows from equation 3 and Lemma 6. ■

Adversary for α_3:

1. *Send outgoing messages of round i:* Based on the messages received during round $i-1$, $\mathcal{A}$ decides on the messages to be sent in round i. For round 1, $\mathcal{A}$ sends to C what an honest B would have sent to C in α_2 and $\mathcal{A}$ sends to A what an honest B would have sent to A in α_2. For $i \geq 2$, $\mathcal{A}$ authenticates $msg_{i-1}^{\alpha_3}(C,B)_C$ using B's key and sends it to A. For $msg_{i-1}^{\alpha_3}(A,B)_A$, $\mathcal{A}$ examines the message. If the message has not been authenticated by C even once, then $\mathcal{A}$ authenticates and sends same message to C as an honest B would have sent to C in round i of execution α_2. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\alpha_3}(A,B)_A$,($\mathcal{A}$ can construct $msg_{i-1}^{\alpha_3}(A,B)_A$, since it passively controls A and has messages received by A in previous rounds.) such that $msg_{i-1}^{\alpha_3}(A,B)_A \sim msg_{i-1}^{\alpha_2}(A,B)_A$, authenticates it using B's key and sends it to C. If the message has been authenticated by C even once, $\mathcal{A}$ simply authenticates $msg_{i-1}^{\alpha_3}(A,B)_A$ using B's key and sends it to C.

2. *Receive incoming messages of round i:* $\mathcal{A}$ obtains messages $msg_i^{\alpha_3}(A,B)_A$ and $msg_i^{\alpha_3}(C,B)_C$ in α_3 via B. (These are round i messages sent by A and C respectively to B). Similarly via A, $\mathcal{A}$ obtains messages $msg_i^{\alpha_3}(B,A)_B$ and $msg_i^{\alpha_1}(C,A)_C$ in α_3. (These are also round i messages sent by B and C respectively to A. Players respectively compute these messages according to their input, secret key, protocol run by them and the view they get up to round $i-1$).

Owing to symmetry of system L, using the proof technique similar to one used in proof of Lemma 4, 5, one can prove the following:

Lemma 8 *$msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_3}(x,C)_x$ and $msg_i^{\alpha}(x,A')_x \sim msg_i^{\alpha_3}(x,A)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$*

Lemma 9 *$view_C^{\alpha} \sim view_C^{\alpha_3}$ and $view_{A'}^{\alpha} \sim view_A^{\alpha_3}$.*

As an interesting observation, it appears that the proof of Lemma 4, 6, 8 requires *directed* system, unlike *undirected* systems used in extant literature [10, 19]. This is because using directed edges one can restrict the paths through which messages are sent to some selected nodes. This is important because in order to make the views same, it is essential to ensure that whatever message is sent in L, adversary $\mathcal{A}$ can generate similar messages in different executions in $\mathcal{N}$. Specifically for the proof of above mentioned lemmas to go through, it is essential that in execution α A,B,C or A' do not ever receive any message from either of B' or C'.

7 Complete Characterization

We now give the necessary and sufficient conditions for ABG_{mix} tolerating a (t_b,t_p)-adversary over any completely connected synchronous network. As a prelude we first show that there does not exist any protocol solving ABG_{mix} over a complete network $\mathcal{N}'$(Figure 9) of four nodes $\mathbb{P} = \{A,B,C,D\}$, tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. For the rest of this work, $((x_1,\ldots,x_i),(y_1,\ldots,y_j))$ represents a single element of adversary basis such that adversary can corrupt $x_1,\ldots,x_i$ actively and simultaneously control $y_1,\ldots,y_j$ passively. The proof technique is similar to one used in proof of Theorem 1.

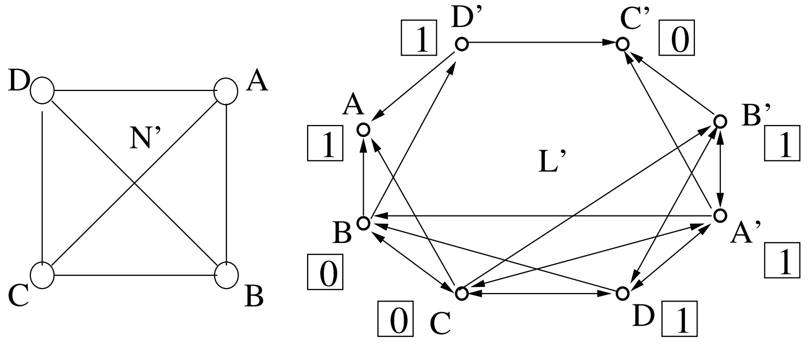

Figure 9: Network $\mathcal{N}'$ and System L'.

Lemma 10 *There does not exist any protocol solving ABG_{mix} over a complete network $\mathcal{N}'$ of four nodes $\mathbb{P} = \{A,B,C,D\}$, tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$.*

Proof: We assume that there exists a protocol η that solves ABG_{mix} over a completely connected network $\mathcal{N}'$ of four nodes, tolerating a adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. Using η we create a protocol η'[Definition 3] in such a way that if η exists then so does η'(Lemma 11). Using two copies of η' we construct a system L' (as shown in Figure 9), and show that L' must exhibit contradictory behavior. It follows that our assumption about existence of η is wrong.

We do not know what η' solves. Formally, system L' is a synchronous system with a well defined behavior. That is, L' has a well defined output distribution for any particular input assignment. We show that for a particular input assignment, no such well defined behavior is possible. Further, no player in L' knows the complete system. Each player in aware of only his immediate neighbors. In reality a player may be connected to either a or a', but it cannot differentiate between the two. It knows its neighbor only by its local name which may be a. Further, in-neighborhood of any node a(or a') in L' is same as in-neighborhood of the corresponding node a in $\mathcal{N}'$.

Let β_1, β_2 and β_3 be three scenarios in execution of η over $\mathcal{N}'$. In β_1, B is the General starting with input 0. Adversary $\mathcal{A}$ corrupts A,D actively and controls B passively. In β_2, B is the General. $\mathcal{A}$ corrupts B and controls A passively. Adversary interacts with C as if B started with input 0 and interact with A, D as if B started with input 1. In scenario β_3, B is the General starting with input 1. $\mathcal{A}$ corrupts C actively and controls B passively. Further, let β be an execution of L' where each player starts with input value as shown in Figure 9. All the players in β are honest and follow the designated protocol correctly.

We now claim that no matter for how many rounds η executes, for any round i, $\mathcal{A}$ can ensure that whatever view [equation 2] B,C get in β, $\mathcal{A}$ can generate the same view for B,C in β_1. Similarly we prove that whatever view C,D,A' get in β, $\mathcal{A}$ can generate the same view for C,D,A respectively in β_2. Similarly, whatever view A',B',D get in β, $\mathcal{A}$ can generate the same view for A,B,D respectively in β_3. We prove our claims in Lemma 12 to Lemma 17.

From the definition of ABG_{mix} [Definition 1], in β_1, both B, C should decide on value 0. Since view of B, C is same in β_1 and β, both A, B in β will also decide on value 0 (We are able to make claims regarding the outputs of B and C in β as their views are same as those in β_1. Thus by analyzing their outputs in β_1, we can determine there outputs in β.). Similarly, $\mathcal{A}$ can ensure that view A', B', D in β is same as view of A, B, D in β_3. A, B, D in β_3 will eventually decide upon value 1. Then so should A', B', D in β. Now C, D have same view in β and β_2. As per Definition 1 C, D in β_2 should agree on same value. Then so should C, D in β. But C,D have already decided upon values 0 and 1 respectively in β. This implies L' must exhibit contradictory behavior. ■

To complete the above proof we show that – B, C get same view in β and β_1; C, D get same view in β and β_2 and view of A', B', D in β is same as view of A, B, D respectively in β_3. We prove the same in Lemmas 12 - 17. The proof technique is similar to one used in proofs of Lemma 4- 9. As a prelude, we define the protocol η'[Definition 3] and show that if η exists then so does η'(Lemma 11). ■

Definition 3 (η') *All statements in η' are same as those in η except the following:*

- *any statement in η of the kind "A sends message m to B" is replaced by "A multicasts message m to all instances of B"(i.e. B,B′)* [7]*. Similarly, "A sends message m to C" is replaced by "A multicasts message m to all instances of C"(i.e. C,C′).*

- *any statement in η of the kind "C sends message m to A" is replaced by "C multicasts message m to all instances of A"(i.e. A,A′). Similarly, "C sends message m to B" is replaced by "C multicasts message m to all instances of B"(i.e. B,B′).*

- *any statement in η of the kind "D sends message m to B" is replaced by "D multicasts message m to all instances of B"(i.e. B,B′).*

Lemma 11 *If η exists then η' exists.*

[7] B and B' are independent copies of B with same authentication key.

Proof: Implied from Definition 3. ■

For scenario β_1, we now specify the behavior of the adversary:

1. *Send outgoing messages of round i:* Based on the messages received during round $i-1$, $\mathcal{A}$ decides on the messages to be sent in round i. In round 1, $\mathcal{A}$ sends to C what an honest A and D would have sent to C in round 1 of β_2. For $i \geq 2$, $\mathcal{A}$ authenticates $msg_{i-1}^{\beta_1}(C,A)_C$ using A's secret key and sends it to B,D. Similarly, $\mathcal{A}$ authenticates $msg_{i-1}^{\beta_1}(C,D)_C$ using D's secret key and sends it to A,B. For $msg_{i-1}^{\beta_1}(B,A)_B$, $\mathcal{A}$ examines the message. If the message has not been authenticated by C even once then $\mathcal{A}$ authenticates and sends same message to C as an honest A would have sent to C in β_2. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\beta_1}(B,A)_B$, such that $msg_{i-1}^{\beta_1}(B,A)_B \sim msg_{i-1}^{\beta_2}(B,A)_B$, authenticates it using A's key and sends it to C. If $msg_{i-1}^{\beta_1}(B,A)_B$ has been authenticated by C even once, $\mathcal{A}$ simply authenticates the message using A's key and sends it to C. Likewise $\mathcal{A}$ examines $msg_{i-1}^{\beta_1}(B,D)_B$. If the message has not been authenticated by C even once $\mathcal{A}$ authenticates and sends same message to C as an honest D would have sent to C in execution β_2. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\beta_1}(B,D)_B$ such that $msg_{i-1}^{\beta_1}(B,D)_B \sim msg_{i-1}^{\beta_2}(B,D)_B$, authenticates it using D's key and sends it to C. If $msg_{i-1}^{\beta_1}(B,D)_B$ has been authenticated by C even once, $\mathcal{A}$ authenticates the message using D's key and sends it to C.

2. *Receive incoming messages of round i:* $\mathcal{A}$ obtain messages $msg_i^{\beta_1}(B,A)_A$, $msg_i^{\beta_1}(C,A)_C$ and $msg_i^{\beta_1}(D,A)_D$ via A. Similarly via D $\mathcal{A}$ gets $msg_i^{\beta_1}(A,D)_A$, $msg_i^{\beta_1}(B,D)_B$ and $msg_i^{\beta_1}(C,D)_C$. (These are round i messages sent by B,C, D to A and A,B,C to D respectively). Similarly, $\mathcal{A}$ obtains $msg_i^{\beta_1}(A,B)_A$, $msg_i^{\beta_1}(C,B)_C$ and $msg_i^{\beta_1}(D,B)_D$ via B. (These are round i messages sent by A,C,D to B. A,C,D respectively compute these messages according to their input value, secret key, protocol run by them and the view they get up to receive phase of round $i-1$.)

We now show that the messages received by B,C in round i of β are same as the messages received by B,C respectively in round i of β_1. Our technique is same as one used in proof of Lemma 4

Lemma 12 *$msg_i^{\beta}(x,B)_x \sim msg_i^{\beta_1}(x,B)_x$ and $msg_i^{\beta}(x,C)_x \sim msg_i^{\beta_1}(x,C)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$.*

Proof: We show that for any round i, adversary can ensure that B receives same messages in β and β_1 i.e. $msg_i^{\beta}(x,B)_x \sim msg_i^{\beta_1}(x,B)_x$, $\forall x \in \mathbb{P}$. Argument for $msg_i^{\beta}(x,C)_x \sim msg_i^{\beta_1}(x,C)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ follows similarly. We apply induction on heights of T_{β}^{B} and $T_{\beta_1}^{B}$ (as shown in Figure 12). Note that only nodes present in T_{β}^{B} are B, C, D, A', B'. Corresponding nodes present in $T_{\beta_1}^{B}$ are B, C, D, A, B respectively.

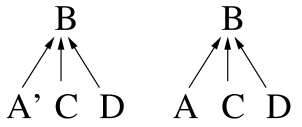

Figure 10: Execution trees T_{β}^{B} and $T_{\beta_1}^{B}$ at the end of round 1.

We analyze these trees in bottom up manner. Consider trees T_{β}^{B} and $T_{\beta_1}^{B}$ at the end of round 1 as shown in Figure 10. C starts with same input, secret key and executes same code in β and β_1. Thus it will send same messages to B in round 1 of β and β_1 i.e. $msg_1^{\beta}(C,B)_C \sim msg_1^{\beta_1}(C,B)_C$. Since A and D are faulty in β_1, aforementioned adversary $\mathcal{A}$ can ensure that $msg_1^{\beta}(A',B)_{A'} \sim msg_1^{\beta_1}(A,B)_A$ and $msg_1^{\beta}(D,B)_D \sim msg_1^{\beta_1}(D,B)_D$. Thus B gets same messages at the end of round 1 in β and β_1 i.e. $msg_1^{\beta}(x,B)_x \sim msg_1^{\beta_1}(x,B)_x$, $\forall x \in \mathbb{P}$.

We now claim that the similarity holds for round 2 as well i.e. $msg_2^{\beta}(x,B)_x \sim msg_2^{\beta_1}(x,B)_x$, $\forall x \in \mathbb{P}$. Consider trees T_{β}^{B} and $T_{\beta_1}^{B}$ at the end of round 2 as shown in Figure 11. Consider node B at level

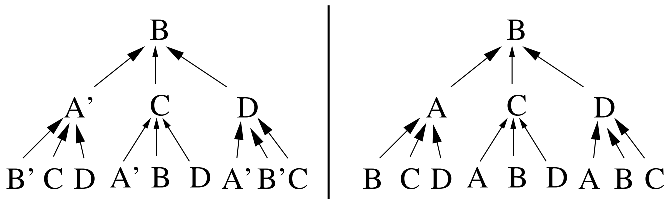

Figure 11: Execution trees T^B_β and $T^B_{\beta_1}$ at the end of round 2.

1 in T^B_β and $T^B_{\beta_1}$. Node B starts with same input value, secret key and execute same code in both β and β_1 respectively, thus $msg^{\beta}_1(B,C)_B \sim msg^{\beta_1}_1(B,C)_B$. Since A,D are faulty, $\mathcal{A}$ can ensure that $msg^{\beta_1}_1(A,C)_A \sim msg^{\beta}_1(A',C)_{A'}$ and $msg^{\beta_1}_1(D,C)_D \sim msg^{\beta}_1(D,C)_D$. Thus C receives same messages at the end of round 1 in β and β_1. Since C starts with same input value, secret key and execute same code in both β and β_1 respectively, it sends same message to B in round 2 i.e. $msg^{\beta}_2(C,B)_C \sim msg^{\beta_1}_2(C,B)_C$. Now consider A' at level 2 in T^B_β and corresponding A at level 2 in $T^B_{\beta_1}$. B' in β starts with a different input from B in β_1, thus $msg^{\beta}_1(B',A')_{B'} \nsim msg^{\beta_1}_1(B,A)_B$. However since A is faulty and B is passively corrupt in β_1, $\mathcal{A}$ on behalf of B can construct $msg^{\beta_1}_1(B,A)_B$ such that $msg^{\beta_1}_1(B,A)_B \sim msg^{\beta}_1(B',A')_{B'}$. C starts with same input value, secret key and execute same code in both β and β_1 respectively, thus $msg^{\beta}_1(C,A')_C \sim msg^{\beta_1}_1(C,A)_C$. Since D is faulty, $\mathcal{A}$ can ensure that $msg^{\beta_1}_1(D,A)_D \sim msg^{\beta}_1(D,A')_D$. Thus A' in β receives same messages at the end of round 1 as A in β_1. Since A is faulty in β_1, $\mathcal{A}$ can ensure that A in β_1 sends message to B in round 2 same as what A' in β sends to B in round 2 i.e. $msg^{\beta_1}_2(A,B)_A \sim msg^{\beta}_2(A',B)_{A'}$. Similarly one can show that $msg^{\beta_1}_2(D,B)_D \sim msg^{\beta}_2(D,B)_D$. Thus $msg^{\beta}_2(x,B)_x \sim msg^{\beta_1}_2(x,B)_x$, $\forall x \in \mathbb{P}$.

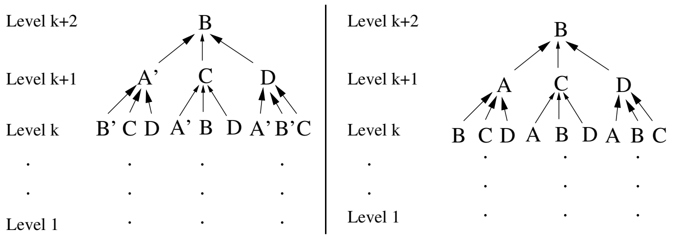

Figure 12: T^B_β and $T^B_{\beta_1}$ at the end of $k+1$ rounds.

Let the similarity be true till some round k i.e. $msg^{\beta}_i(x,B)_x \sim msg^{\beta_1}_i(x,B)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T^B_β and $T^B_{\beta_1}$ at the end of $k+1$ rounds as shown in Figure 12. To prove the induction step we need to show that B at level $k+2$ receives same messages in both trees. Consider node D at level $k+1$. One can argue that C till round k also gets same messages in β and β_1. This is because from similar induction hypothesis step on heights of T^C_β and $T^C_{\beta_1}$, one gets $msg^{\beta}_i(x,C)_x \sim msg^{\beta_1}_i(x,C)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. Now, since C starts with same input value, secret key and execute same code in both β and β_1 respectively, it sends same messages to D in round k i.e. $msg^{\beta}_k(C,D)_C \sim msg^{\beta_1}_k(C,D)_C$. For time being assume A' receives messages till round k in β_1 same as what A receives till round k in β. Since A is faulty in β_1, $\mathcal{A}$ can ensure that A sends same message to D in β_1 as A' sends to D in β i.e. $msg^{\beta}_k(A',D)_{A'} \sim msg^{\beta_1}_k(A,D)_A$. Similarly assume that B' receives messages till round k in β_1 same as what B receives messages till round k in β. But B in β_1 starts with a different input from B' in β, thus they send different messages to D in β and β_1. However since D is faulty and B is passively corrupt in β_1, $\mathcal{A}$ can ensure that $msg^{\beta}_k(B',D)_{B'} \sim msg^{\beta_1}_k(B,D)_B$. Thus D at level $k+1$ receives same messages in T^B_α and $T^B_{\alpha_1}$. Since D

is faulty in β_1, $\mathcal{A}$ can ensure that $msg^{\beta}_{k+1}(D,B)_D \sim msg^{\beta_1}_{k+1}(D,B)_D$. Using similar arguments one can show that $msg^{\beta}_{k+1}(C,B)_C \sim msg^{\beta_1}_{k+1}(C,B)_C$ and $msg^{\beta}_{k+1}(A',B)_{A'} \sim msg^{\beta_1}_{k+1}(A,B)_A$. Thus B receives same messages in round $k+1$ of β and β_1. Thus induction hypothesis holds for round $k+1$ too. Thus $msg^{\beta}_i(x,B)_x \sim msg^{\beta_1}_i(x,B)_x, \forall x \in \mathbb{P}$ holds true. The above proof is based on the assumption that A' up to level k in T^B_α receives same messages as corresponding A in $T^B_{\beta_1}$. Using induction and arguments similar to given above one can show easily that both assumptions indeed holds true. Similarly one can prove that B' up to level k in T^B_α receives same messages as corresponding B in $T^B_{\beta_1}$. ■

Lemma 13 *$view^{\beta}_B \sim view^{\beta_1}_B$ and $view^{\beta}_C \sim view^{\beta_1}_C$*

Proof: Follows from equation 3 and Lemma 12. ■

Adversary for β_2:

1. *Send outgoing messages of round i:* Based on the messages received in round $i-1$, $\mathcal{A}$ decides on the messages to be sent in round i. In round 1, $\mathcal{A}$ sends to C what an honest B would have sent to C in round 1 of β_1. Similarly $\mathcal{A}$ sends to D what an honest B would have sent to D in round 1 of β_3 and $\mathcal{A}$ sends to A what an honest B would have sent to A in round 1 of β_3. For $i \geq 2$, $\mathcal{A}$ authenticates $msg^{\beta_2}_{i-1}(C,B)_B$ using B's secret key and sends it to A,D. Similarly, $\mathcal{A}$ authenticates $msg^{\beta_2}_{i-1}(D,B)_D$ using B's secret key and sends it to A,C. For $msg^{\beta_2}_{i-1}(A,B)_A$, $\mathcal{A}$ examines the message. If the message has not been authenticated by either C or D even once, then $\mathcal{A}$ authenticates and sends same message to C as an honest B would have sent to C in β_1. Similarly $\mathcal{A}$ authenticates and sends same message to D as an honest B would have sent to D in β_3. Formally, $\mathcal{A}$ constructs $msg^{\beta_2}_{i-1}(A,B)_A$, such that $msg^{\beta_2}_{i-1}(A,B)_A \sim msg^{\beta_1}_{i-1}(A,B)_A$, authenticates it using B's key and sends it to C. Similarly $\mathcal{A}$ constructs $msg^{\beta_2}_{i-1}(A,B)_A$, such that $msg^{\beta_2}_{i-1}(A,B)_A \sim msg^{\beta_3}_{i-1}(A,B)_A$, authenticates it using B's key and sends it to D. If $msg^{\beta_2}_{i-1}(A,B)_A$ has been authenticated by either C or D even once, $\mathcal{A}$ simply authenticates the message using B's key and sends it to C and D.
2. *Receive incoming messages of round i:* $\mathcal{A}$ obtains messages $msg^{\beta_2}_i(A,B)_A$, $msg^{\beta_2}_i(C,B)_C$ and $msg^{\beta_2}_i(D,B)_D$ from B in β_2 (These are round i messages sent by A,C,D to B. They respectively compute these messages according to their input, protocol run by them and the view they get up to receive phase of round $i-1$.). Similarly $\mathcal{A}$ obtains $msg^{\beta_2}_i(B,A)_B$, $msg^{\beta_2}_i(C,A)_C$ and $msg^{\beta_2}_i(D,A)_D$ from A in β_2 (These are round i messages sent by B,C,D to A).

Similar to Lemma 12,13 one can prove the following lemmas.

Lemma 14 *$msg^{\beta}_i(x,C)_x \sim msg^{\beta_2}_i(x,C)_x$, $msg^{\beta}_i(x,D)_x \sim msg^{\beta_2}_i(x,D)_x$ and $msg^{\beta}_i(x,A')_x \sim msg^{\beta_2}_i(x,A)_x$ $\forall i > 0$, $\forall x \in \mathbb{P}$.*

Lemma 15 *$view^{\beta}_C \sim view^{\beta_2}_C$, $view^{\beta}_D \sim view^{\beta_2}_D$, $view^{\beta}_{A'} \sim view^{\beta_2}_A$*

Adversary for β_3:

1. *Send outgoing messages of round i:* Based on the messages received in round $i-1$, $\mathcal{A}$ decides on the messages to be sent in i. In round 1, $\mathcal{A}$ sends to D what an honest C would have sent to D in round 1 of β_2. For $i \geq 2$ $\mathcal{A}$ authenticates $msg^{\beta_3}_{i-1}(A,C)_A$ using secret key of C and sends it to B,D. Similarly it authenticates $msg^{\beta_3}_{i-1}(D,C)_D$ using C's secret key and sends it to A,B. For $msg^{\beta_3}_{i-1}(B,C)_B$, $\mathcal{A}$ examines the message. If the message has not been authenticated by either A or D even once, then $\mathcal{A}$ authenticates and sends same message to A as an honest C would have sent to A in β_2 and sends same to D as an honest C would have sent to D in execution β_2. Formally, $\mathcal{A}$

constructs $msg^{\beta_3}_{i-1}(B,C)_B$, such that $msg^{\beta_3}_{i-1}(B,C)_B \sim msg^{\beta_2}_{i-1}(B,C)_B$ authenticates it using C's key and sends it to A,D. If $msg^{\beta_3}_{i-1}(B,C)_B$ has been authenticated by either of A or D even once, $\mathcal{A}$ simply authenticates the message using C's key and sends it to A,D.

2. *Receive incoming messages of round i:* $\mathcal{A}$ obtains messages $msg^{\beta_3}_i(A,C)_A$, $msg^{\beta_3}_i(B,C)_B$ and $msg^{\beta_3}_i(D,C)_D$ via C. (These are round i messages sent by A,B and D to C). Similarly $\mathcal{A}$ obtains $msg^{\beta_3}_i(A,B)_A$, $msg^{\beta_3}_i(C,B)_C$ and $msg^{\beta_3}_i(D,B)_D$ via B. (These are round i messages sent by A,C and D to B. A,C and D respectively compute these messages according to the protocol run by them and the view they get receive phase of round $i-1$.)

Similar to Lemma 12,13 one can prove the following lemmas.

Lemma 16 *$msg^{\beta}_i(x,A')_x \sim msg^{\beta_3}_i(x,A)_x$, $msg^{\beta}_i(x,B')_x \sim msg^{\beta_3}_i(x,B)_x$, and $msg^{\beta}_i(x,D)_x \sim msg^{\beta_3}_i(x,D)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$.*

Lemma 17 *$view^{\beta}_{A'} \sim view^{\beta_3}_{A}$, $view^{\beta}_{B'} \sim view^{\beta_3}_{B}$, $view^{\beta}_{D} \sim view^{\beta_3}_{D}$.*

We now present the main theorem of this work.

Theorem 18 (Main Theorem) *ABG_{mix} over a completely connected synchronous network $\mathcal{N}'$ of n nodes tolerating (t_b,t_p)-adversary is possible if and only if if $n > 2t_b + min(t_b,t_p)$, $t_p > 0$.*

Proof: We first give necessity proof followed by proof for sufficiency.

`Necessity:` We first prove that there does not exist any protocol solving ABG_{mix} over a completely connected synchronous network $\mathcal{N}'$ of n nodes tolerating (t_b,t_p)-adversary when $n \leq 2t_b + min(t_b,t_p)$, for $t_p > 0$. We present the proof separately for $t_b > t_p$ and $t_b \leq t_p$.

`Case of` $t_b > t_p$: We assume that there exists a protocol η solving ABG_{mix} complete network $\mathcal{N}'$ tolerating (t_b,t_p)-adversary when $n \leq 2t_b + min(t_b,t_p)$. Using η, we construct a protocol η' which solves ABG_{mix} over a completely connected graph of four nodes, tolerating $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. However, from Lemma 10, we know that there cannot exist any such η'. This contradicts our assumption that there exists a solution η solving ABG_{mix} for $n \leq 2t_b + min(t_b,t_p)$, $t_p > 0$.

We now show as to how to transform η into a solution η' which solves ABG_{mix} over a completely connected graph of four nodes, tolerating $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. Divide n players in η into sets I_A, I_B, I_C, I_D, such that their respective sizes are $min(t_b,t_p), min(t_b,t_p), t_b$, $(t_b - min(t_b,t_p))$. $\mathbb{A}$ can corrupt all the players in any of the following sets $I_A, I_B, I_C, I_D, (I_A \cup I_D), (I_B \cup I_D)$ actively and players in I_A, I_B, I_D passively. Note that the players from the set I_C cannot be corrupted passively. Each of the four players A,B,C and D in η' simulate players in I_A, I_B, I_C, I_D respectively. Each player i in η' keeps track of the states of all the players in I_i. Player i assigns its input value to every member of I_i, and simulates the steps of all the players in I_i as well as the messages sent and received between pairs of players in I_i. Messages from players in I_i to players in I_j are simulated by sending same messages from player i to player j. If any player in I_i terminates then so does player i. If any player in I_i decides on value v, then so does player i.

We now prove that if η solving ABG_{mix} tolerating (t_b,t_p)-adversary when $n \leq 2t_b + min(t_b,t_p)$, then η' solves ABG_{mix} tolerating $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. For simplicity we assign any actively and passively corrupted players of η to be exactly those that are simulated by actively and passively corrupted player in η'. Let ψ' be an execution of η' with the faults characterized by $\mathbb{A}$ = $\{((A,D),(B)),((B),(A)),((C),(B))\}$. Let ψ be an execution of η. As per our assumption ψ solves ABG_{mix}, thus ψ satisfies Definition 1. We now show that same holds for ψ' if it holds for ψ. W.l.o.g in

ψ, let the general be from set I_i, then in ψ', player i acts as the general. Note that in ψ if I_i is controlled actively or passively by the adversary, then so is i is ψ'. Let j,k ($j \neq k$) be two non-faulty players in ψ'. j and k simulates at least one player each in ψ. w.l.o.g let them simulate players in I_j, I_k. Since j and kl are non-faulty, so are all players in I_j, I_k. For ψ, all players in I_j, I_k must terminate, then so should j and k. In ψ, all non-faulty players including I_j, I_k should agree on same value say u, then in ψ', j, k also agree on u. In ψ, if the general is non-faulty and starts with value v, then in ψ' too, general will be non-faulty and starts with value v. In such a case in ψ, all non-faulty players including I_j, I_k should have $u = v$, then in ψ', j, k will also have $u = v$. Thus ψ' also satisfies Definition 1. Then, η' solves ABG_{mix} tolerating $\mathbb{A} = \{((A, D), (B)), ((B), (A)), ((C), (B))\}$.

`Case of` $t_b \leq t_p$: In this case, the expression $n \leq 2t_b + min(t_b, t_p)$ reduces to $n \leq 3t_b$. We assume that there exists a protocol λ solving ABG_{mix} complete network $\mathcal{N}'$ tolerating (t_b,t_p)-adversary when $n \leq 3t_b$. Using λ, we construct a protocol λ' which solves ABG_{mix} over a completely connected graph of three nodes, tolerating a (1,1)-adversary. However, from Theorem 1, we know that there cannot exist any such λ'. This contradicts our assumption that there exists a solution λ solving ABG_{mix} for $n \leq 3t_b$, when $t_b \leq t_p$.

Given λ, we now show as to how can one construct λ'. Divide the n players into three sets I_1, I_2 and I_3 such each is of size at max t_b i.e. $|I_i| \leq t_b$ (Since $n \leq 3t_b$ such a division is always possible). Further, since $t_b \leq t_p$, it vacuously implies $|I_i| \leq t_p$. Thus, adversary can corrupt any of I_i actively and I_j passively, $i \neq j$. Let player i in λ' simulate all the players in I_i. For simplicity we assign any actively and passively corrupted players in an execution of λ to be exactly those that are simulated by actively and passively corrupted player in corresponding execution of λ'. Similar to the argument presented for $t_b > t_p$, one can show that if λ satisfies Definition 1, then so does λ'. This completes the necessity proof.

`Sufficiency -` For sufficiency we present protocol for $n > 2t_b$+min(t_b, t_p), $t_p > 0$. We present the protocol separately for $t_b > t_p$ and $t_b \leq t_p$.

`Case of` $t_b > t_p$: $n > 2t_b$+min(t_b, t_p) reduces to $n > 2t_b + t_p$. For this we present a protocol and prove its correctness in section 7.1.

`Case of` $t_b \leq t_p$: $n > 2t_b$+min(t_b, t_p) reduces to $n > 3t_b$. Here any protocol for unauthenticated Byzantine Generals Problem works (such as *EIGByz* protocol given in [20, page 119]). This is because for unauthenticated setting $t_p = n - t_b$. This completes the sufficiency proof. We remark that for t_p=0, the result reduces to $n > t_b$ [21]. ∎

7.1 Protocol for $n > 2t_b + t_p$

The proposed protocol is obtained by a sequence of transformations on EIG tree [2]. A detailed description of the construction of EIG tree is available in [20, page 108]. Our protocol *EIGPrune* is given in Figure 13.

Definition 4 (Prune(EIG)) *This method that takes an EIG tree as an input and deletes subtrees say $subtree_j{}^i$ ($subtree_j{}^i$ refers to a subtree in i's EIG tree such that the subtree is rooted at node whose's label is j) of i's EIG tree as given in the sequel. For each subtree $subtree_j{}^i$, where label $j \in \mathbb{P}$, a set W_j is constructed which contains all distinct values that ever appears in $subtree_j{}^i$. If $|W_j| > 1$, $subtree_j{}^i$ is deleted and modified EIG tree is returned.*

We prove the correctness of *EIGPrune* via Lemma 19 – 22.

Lemma 19 *The $subtree_j{}^i$, where j is an honest player and i is a non-faulty player, will never be deleted during* **Prune**(EIG) *operation.*

***EIGPrune* Algorithm**

General $\mathcal{G}$ send his value to every player. Every player assumes this value from the $\mathcal{G}$ as his input value and and exchanges messages with others as per *EIGStop* protocol in [20, page 110] for $t_b + t_p + 1$ rounds.

At the end of t_b+t_p+1 rounds of *EIGStop* protocol, player p_i invokes **Prune**(EIG) [Definition 4]. Player p_i applies the following decision rule – take majority of the values at the first level [8] of its *EIG* tree (note that he does not need to take a majority over the entire *EIG* tree). If a majority exists player, p_i decides on that value; else, p_i decides on *default value*, v_0.

Figure 13: *EIGPrune* algorithm

Proof: This Lemma stems from the fact that any message signed by an honest player cannot be changed in the course of the protocol. Thus, a ${subtree_j}^i$, j being an honest player will never be deleted in **Prune**(*EIG*) and will be consistent throughout for all non-faulty players. ■

Lemma 20 *After $t_b + t_p + 1$ rounds, if a ${subtree_j}^i$ has more than one value then $\forall$ k, ${subtree_j}^k$ also has more than one value, there by ensuring that all $\forall$ k, ${subtree_j}^k$ are deleted (i, j, k are not necessarily distinct), where i, k are non-faulty.*

Proof: Any message sent in $(t_b + t_p)^{th}$ round has a label of length $t_b + t_p$ and hence we are sure to have either an honest player already having signed on it or in $(t_b + t_p + 1)^{th}$ round an honest player would broadcast it. This ensures that a value cannot be changed/reintroduced in the $(t_b + t_p + 1)^{th}$ round. In other words, a faulty player can either send different initial values in round one or change a value in Round k, $2 \leq k \leq t_b + t_p$, if and only if all players who have signed so far on that message are under the control of adversary. In any case, the non-faulty players send these values in the next round and hence the Lemma. ■

Lemma 21 *${subtree_j}^i$ and ${subtree_j}^k$ in the* EIG *trees of any two players i, k will have same values after the subjecting the tree to* **Prune**(EIG)*, where i, k are non-faulty players.*

Proof: This follows from previous Lemma 20 as, if subtrees had different values; then as per the protocol they would have broadcasted the values in their *EIG* tree in the next round and thus the subtrees would have more than one different value resulting in their deletion during **Prune**(*EIG*) step. ■

Lemma 22 *For $n > 2t_b + t_p$,* EIGPrune *algorithm solves ABG_{mix}.*

Proof: Termination is obvious, by the decision rule. $n - (t_b + t_p)$ represents the number of honest players and according to $n > 2t_b + t_p$, $n - (t_b + t_p) > t_b$. Thus honest majority is guaranteed which vacuously implies non-faulty majority. Now if General is non-faulty and starts with v, then all the non-faulty players also start with v. The decision rule ensures that in such a case v is the only possible decision value. Thus validity also holds. For agreement, let i and j be any two non-faulty players. Since, decisions only occur at the end, and by previous lemma we see that $\forall i, {subtree_j}^i$ can have only one value which consistent throughout all $subtree^i_j, \forall i \in \mathbb{P}$. This implies they have the same set of values. The decision rule then simply implies that i and j make the same decision. In case of General being faulty, the agreement simply implies that all non-faulty players decide on same value. ■

8 Conclusion

The folklore has been that use of authentication reduces the problem of simulating a broadcast in presence to Byzantine faults to fail-stop failures. Thus, the protocols designed for fail-stop faults can be quickly adapted to solve ABG. However in this work, we have shown that this does not hold true for the case of ABG under the influence of mixed adversary. In a way, the problem of ABG_{mix} covers the entire range of problems between ABG and BGP. Consequentially, the protocols for ABG_{mix} take ideas from both ABG and BGP. From our result of $n > 2t_b + \min(t_b, t_p)$, it appears that studying this problem over general networks will be interesting in its own right.

References

[1] 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] Amotz Bar-Noy, Danny Dolev, Cynthia Dwork, and H. Raymond Strong. Shifting gears: changing algorithms on the fly to expedite byzantine agreement. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 42–51, New York, NY, USA, 1987. ACM Press.

[3] 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.

[4] 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.

[5] 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.

[6] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.

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

[8] Danny Dolev. The byzantine generals strike again. Technical report, Stanford, CA, USA, 1981.

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

[10] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In *PODC '85: Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 59–70, New York, NY, USA, 1985. ACM.

[11] 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.

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

[13] 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.

[14] 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.

[15] J. A. Garay and K. J. Perry. A Continuum of Failure Models for Distributed Computing. In *Proceedings of the 6th International Workshop on Distributed Algorithms*, volume 647 of *Lecture Notes in Computer Science (LNCS)*, pages 153–165. Springer-Verlag, 1992.

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

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

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

[19] Y. Lindell, A. Lysysanskaya, and T. Rabin. On the Composition of Authenticated Byzantine Agreement. In *Proceedings of the 34th Symposium on Theory of Computing (STOC)*, pages 514–523. ACM Press, 2002.

[20] N. Lynch. *Distributed Algorithms.* Morgan Kaufmann, San Mateo, CA, USA, 1996.

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

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

[23] Ulrich Schmid and Bettina Weiss. Synchronous byzantine agreement under hybrid process and link failures. Research Report 1/2004, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

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