# 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 General* (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 study the problem of ABG in ($t_b$,$t_p$)-mixed adversary model where adversary can corrupt up to any $t_b$ players actively and control up to any 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.

**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.* [19]. 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 $u = v$. In a completely connected synchronous network with no additional setup, classical results of [19, 22] 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 [19, 22], 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 [23] to name a few. An important variant of BGP is the authenticated model proposed by Pease *et al.* [22], which as the title of this paper 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, 25, 4, 18, 16, 24]. In essence, the state-of-the-art in ABG can be summarized by the following folklore (as noted by Nancy Lynch [21, 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 permits to understand the effects on 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 aim to study the problem of ABG under influence of a ($t_b$,$t_p$)-adversary where adversary can corrupt up to any $t_b$ players actively and control another up to any $t_p$ players passively. Adversary can make actively corrupt players to behave in arbitrary manner and can read the internal state of passively corrupt players. Note that the solution to the problem of ABG with authentication under influence of a ($t_b$,$t_p$)-adversary answers the question of simulating a broadcast channel for the entire gamut of adversary strategies between $t_b = t$ & $t_p = 0$ (ABG) and $t_b = t$ & $t_p = n - t$ (BGP).

## 2 Our Contributions and Results

The first contribution of this paper is to argue that for the case of ($t_b$,$t_p$)-adversary, the problem definition of ABG itself needs to be modified. As a preclude to the argument, 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 the paper 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, the General sends a value on this physical broadcast channel. If the General is honest and sends a value say $v$, then all the $n$ players are guaranteed to receive value $v$. Then all the honest players will output $v$. By virtue of correctly following the protocol, all passively corrupt players will also output $v$. Thus all non-faulty will output same value $v$. Note that adversary can make all the actively corrupt players to output a value different from what they receive.[1] In case the General himself is faulty all non-faulty players will output same value.

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 (honest and passively corrupt, i.e. $n - t_b$) players output *same* value. Note that in an authenticated setting such as ABG, passive control also models situations where a player executes the designated protocol faithfully but is unaware of the fact that his private key has been compromised. In such a case, from the arguments presented in preceding paragraph, 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. For the rest of the paper we call this model as "ABG under mixed adversary" ($ABG_{mix}$). We formally define $ABG_{mix}$ in section 3.

From the result of $n > t$ [22], one might feel that in the presence of ($t_b$,$t_p$)-adversary, $n > t_b$+$t_p$ (using $t_b$+$t_p$=$t$) is sufficient for possibility of $ABG_{mix}$. However we show that this is not the case, and $n > t_b$+$t_p$ is necessary but not sufficient to simulate a broadcast channel as originally intended. We support our claim by studying a simple synchronous system consisting of three players (as illustrated in graph $G$ in Figure 1). For $n > t_b$+$t_p$ to be a sufficient condition for $ABG_{mix}$ over any complete graph tolerating ($t_b$,$t_p$)-adversary, there should exist at a protocol solving $ABG_{mix}$ tolerating ($t_b$,$t_p$)-adversary, whenever $n > t_b$+$t_p$ is satisfied. However, in Section 4 we prove that for the case of three players over a completely connected graph $G$ if the strategy of the $(1, 1)$-adversary is to actively control one of the players and passively control another one player, then *there cannot exist any protocol that can guarantee consistency among the outputs of all the non-faulty players.* In light of this observation we initiate the study of $ABG_{mix}$ in presence of ($t_b$,$t_p$)-adversary. As a second contribution of this paper we formally prove that $ABG_{mix}$ 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 $n > t$ [22] still holds.

# 3 Our Model and Notations

We consider a set of $n$ players, computationally unbounded, denoted by $\mathbb{P}$, fully connected. Player $P_i$ is modeled as an interactive Turing Machine with $n - 1$ pairs of incoming and outgoing communication tapes. 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. In the send phase of each round, players write messages onto their outgoing communication tapes, and in the receive phase, players read the content of their incoming communication tapes. During the execution, the adversary may take control of 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. 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 i.e. adversary cannot modify messages sent between non-malicious parties. We also assume existence of a

[1] A similar argument can be given using a TTP (Trusted Third Party) [6]. The designated 'General' sends its value $v$ to TTP. TTP forwards it to all the players. Since all honest and passively corrupt players follow the ideal protocol diligently, they all output $v$.

(signature/authentication) scheme where the sender signs the message to be sent. This is modeled by all parties also having an additional setup-tape that is generated during the preprocessing phase.[2] Typically in such a preprocessing phase, the signature keys are generated. That is, each party 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 can uniquely identify the sender of the message using the signature. However, the adversary can forge the signature of all the ($t_b$+$t_p$) players under its control. W.l.o.g we assume that players authenticate themselves and their messages with the help of a private key. Based on the discussion in the previous section, we now formally define $ABG_{mix}$:

**Definition 1** ($ABG_{mix}$) *A designated General starts with an input from a fixed set $V = \{0, 1\}$. The goal is for the players to eventually output decisions from the set $V$ upholding the following conditions, even in the presence of a $(t_b, t_p)$-adversary:*

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

For clarity of the reader, we reiterate certain terms that have been used extensively in the paper. A player is said to be *faulty* if and only if he deviates from the designated protocol. Consequently *non-faulty* players are ones who do not deviate from the designated protocol. Note that adversary may have some 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 complete 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 paper, both *honest* and *passively corrupt* players are non-faulty. Rest of the paper focuses on complete characterization of $ABG_{mix}$ over complete graphs.

**Organization of the paper:** We start the technical exposition by giving a motivating example to study the problem of $ABG_{mix}$ in section 4. In section 5, we give mathematically rigorous definitions of some terms used in the formal proofs. Section 6 gives the complete characterization of $ABG_{mix}$ tolerating $(t_b, t_p)$-adversary over complete graphs, followed by the conclusion in section 7. Owing to space constraints the formal proof of the motivating example considered in section 4 is given in Appendix A of [17].

# 4 Motivating Example

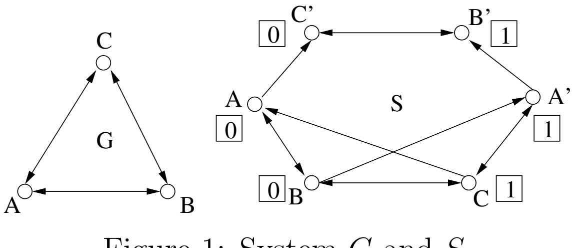

Figure 1: System $G$ and $S$.

As a motivating example to study $ABG_{mix}$ in presence of ($t_b$,$t_p$)-adversary, we first show that there does not exists any protocol solving $ABG_{mix}$ over a complete graph of 3 players influenced by a $(1, 1)$-adversary. This essentially shows that result of $ABG_{mix}$ cannot

[2] 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 [20].

be $n > t_b+t_p$ as is the case with $n > t$ result for ABG [22]. The proof for impossibility of protocol is motivated from [10]. Here we only give a proof sketch, a detailed formal proof is available in Appendix A of [17].

We start by assuming that there exists a protocol $\pi$ that solves $ABG_{mix}$ for three players, $\mathbb{P} = \{A, B, C\}$ tolerating $(1, 1)$-adversary. We then construct a system $S$ as shown in Figure 1. Using $\pi$ we create a protocol $\pi'$ (executed by each player in $S$) in such a way that if $\pi$ exists then so does $\pi'$. Further, let $\alpha_1$ be an execution of $\pi$ in $G$ in which $B$ is an honest player, adversary $\mathcal{A}$ corrupts $C$ in byzantine fashion and $A$ in passive manner. Here $A$ is the General and starts with input 0. Similarly let $\alpha_2$ be the execution of $\pi$ in $G$ in which $B$ is an honest player. $\mathcal{A}$ corrupts $C$ passively and controls $A$ in byzantine fashion. Here $A$ acts as the general. $A$ sends 0 to $B$ and 1 to $C$. Let $\alpha_3$ be an execution of $\pi$ in $G$ in which $C$ is an honest player. $\mathcal{A}$ corrupts $A$ passively, and $B$ in byzantine fashion. Here $A$ acts as the General and starts with input 1. Let $\alpha$ be an execution of $\pi'$ in $S$ in which each player starts with input value as shown in Figure 1. All the players in $\alpha$ are honest and follow the prescribed protocol($\pi'$) correctly. We then prove 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 5) $A, B$ get in $\alpha$, $\mathcal{A}$ can generate the same view for $A, B$ in $\alpha_1$. Since $\mathcal{A}$ can ensure that view of both $A, B$ is same in $\alpha$ and $\alpha_1$, we show that $A, B$ in $\alpha$ will decide on value 0. Similarly one can prove that $A', C$ in $\alpha$ will decide on value 1. On similar lines we then prove that since $B, C$ in $\alpha_2$ should agree on same value, then so should $B, C$ in $\alpha$, but $B$,$C$ have already decided upon values 0 and 1 respectively in $\alpha$, leading to a contradiction in $\pi'$. This contradicts our original assumption about existence of $\pi$.

To complete the proof sketch, we now give an idea as to how $\mathcal{A}$ can ensure that $A, B$ gets same view in $\alpha_1$ and $\alpha$. Consider an execution $\Gamma$ of $\pi'$ in $S$ which is exactly same as $\alpha$ except that in $\Gamma$ $A'$ starts with input value 0. Since in $\alpha$, 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 $\Gamma$ and $\alpha_1$ (All $\mathcal{A}$ has to do is to start with input value 1 and follow the designated protocol). Now in $\alpha$, all messages received by $A$ and $B$ respectively are same as those in $\Gamma$ except those messages that have been processed by $A'$ at least once(since $A'$ starts with input value 0 in $\Gamma$ and input value 1 in $\alpha$). If in $\alpha_1$, $\mathcal{A}$ can simulate this difference between $\alpha$ and $\Gamma$, we can say that $\mathcal{A}$ can make view of $A$ and $B$ same in $\alpha$ and $\alpha_1$. 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 $S$, in $\alpha$ $A'$ can send a message to $A$ or $B$ only via $C$. This ensures that in $\alpha$, any message from $A'$ can reach $A$ or $B$ only after it has been processed by $C$. Now in $\alpha_1$, $C$ is faulty and $\mathcal{A}$ controls $A$ passively. Thus whatever $C$ sends to $A$ and $B$ in $\alpha$, $\mathcal{A}$ can send the same to $A$ and $B$ in $\alpha_1$. Similarly one prove that whatever view $B, C$ get in $\alpha$, $\mathcal{A}$ can generate the same view for $B, C$ in $\alpha_2$ and whatever view $C, A'$ get in $\alpha$, $\mathcal{A}$ can generate the same view for $C, A$ in $\alpha_3$.

Formally one can prove the following lemmas. Here $view_Z^{\phi}$ represents view of player $Z$ during entire execution $\phi$. Detailed formal proofs of these lemmas are given in Appendix A of [17].

**Lemma 1** *$view_A^{\alpha} \sim view_A^{\alpha_1}$ and $view_B^{\alpha} \sim view_B^{\alpha_1}$*

**Lemma 2** *$view_B^{\alpha} \sim view_B^{\alpha_2}$ and $view_C^{\alpha} \sim view_C^{\alpha_2}$.*

**Lemma 3** *$view_{A'}^{\alpha} \sim view_A^{\alpha_3}$ and $view_C^{\alpha} \sim view_C^{\alpha_3}$.*

**Note**: We remark that *undirected* systems does not seem to work in proving above lemmas. An interested reader is encouraged to try proving Lemmas 1, 2, 3 using undirected system's technique used in extant literature [10, 20]. Curiously though, the impossibility can be proved using a *directed* system. 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 $S$, adversary $\mathcal{A}$ can generate similar messages in different executions in $G$. Specifically for the proof of above mentioned lemmas to go through, it is essential that $A$,$B$,$C$ or $A'$ donot ever get any message from either of $B'$ or $C'$ in execution $\alpha$. In Appendix A of [17] we elaborate as to why the proof for Lemmas 1, 2, 3 breaks down, in case this constraint in execution $\alpha$ is not satisfied.

# 5 Mathematical Definitions

Prior to giving the complete characterization of $ABG_{mix}$ tolerating ($t_b$,$t_p$)-adversary, we mathematically define certain terms which are used in this work. We start with *view*. Intuitively, by *view* we want to capture all that a player can ever 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 $\Omega$. 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 $\Omega$ at the end of round $i$, denoted by $view_{a,i}^{\Omega}$, can be represented as collection of all the messages it ever send and receives. Formally:

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

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

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

In order to show that the views of 2 different players $a, b$ running in 2 different executions $\Omega, \Gamma$ 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 [3], and if for every round $1 \ldots i$ their corresponding incoming messages are same, then their views till round $i$ will also be same. [4] Formally:

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

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

[4] [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.

# 6 Characterization of $ABG_{mix}$ over Complete Graphs

We now give the necessary and sufficient conditions for possibility of $ABG_{mix}$ over completely connected synchronous networks. We show that $ABG_{mix}$ over a complete graph is possible if and only if $n > 2t_b + \min(t_b, t_p)$. We first give the necessity proof followed by sufficiency.

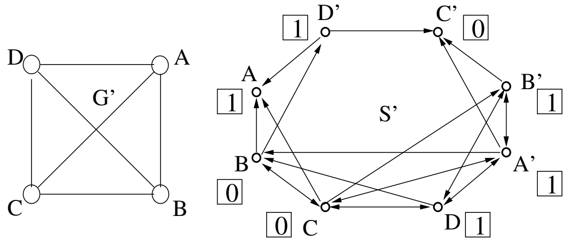

Figure 2: System $G'$ and $S'$.

## 6.1 Necessity

We first show that there does not exists any protocol solving $ABG_{mix}$ over a complete graph of four nodes tolerating adversary basis $\mathbb{A} = \{((A, D), (B)), ((B), (A)), ((C), (B))\}$. For the rest of paper, $((x_1 \dots x_i)(y_1 \dots y_j))$ represents a single element of adversary basis such that adversary can corrupt $x_1 \dots x_i$ actively and simultaneously control $y_1 \dots y_j$ passively. The proof technique used and the intuition behind impossibility is same as that in section 4. Formally we prove the impossibility by contradiction. We assume there exists a protocol $\varpi$ that solves $ABG_{mix}$ over a complete graph of four nodes $G'$ tolerating adversary basis $\mathbb{A} = \{((A, D), (B)), ((B), (A)), ((C), (B))\}$. We then construct a new system $S'$ as shown in Figure 2. Each player in $S'$ runs $\varpi'$. We now formally define $\varpi'$ and further prove that if $\varpi$ exists then so does $\varpi'$.

**Definition 2** ($\varpi'$) *For all players $a, b \in \mathbb{P}$, any statement of kind "b sends message m to a" in $\varpi$ is replaced by "b multicasts message m to all instances of a(i.e. a, a′)* [1] *which are connected by a directed edge from b to a" in $\varpi'$. Rest all statements in $\varpi'$ are same as $\varpi$.*

**Lemma 4** *If $\varpi$ exists then $\varpi'$ exists.*

*Proof*: Implied from Definition 2. ■

**Construction of** $S'$: Take two copies of each player in $G'$, construct a octagonal system $S'$ as shown in Figure 2. Player $A$ is connected to $B$,$C$,$D'$; $B$ is connected to $A$,$C$,$D$,$A'$,$D'$; player $C$ is connected to $A$,$B$,$D$,$A'$,$B'$; player $D$ is connected $B$,$C$,$A'$,$B'$ and so on. Connectivity in $S'$ is shown using directed edges. A node $a$ behaving in a byzantine fashion with a pair of honest nodes, is captured by connecting one of the honest nodes to $a$ and other to $a'$ [4]. Note that connectivity in $S'$ is not same as in $G'$. To be precise, in-neighborhood of any node $a$(or $a'$) in $S'$ is same as in-neighborhood of corresponding node $a$ in $G'$, however out-neighborhood of some nodes in $S'$ is not same as out-neighborhood of corresponding nodes in $G'$. This would make a difference if players in both systems were running same protocol($\varpi$). $S'$ is constructed in a such a way that whatever messages are sent to some selected players in $S'$, same messages can be ensured by adversary to those very selected players in $G'$. Each player in $S'$ knows only its immediate neighbors and not the complete graph $S'$. 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 $a$. We neither know what system $S'$ does nor what $\varpi'$ solves. Since, $S'$ does not form an $ABG_{mix}$ setting, therefore the definition of $ABG_{mix}$ [Definition 1] does not tell us anything directly about the output of players in $\varpi'$. All we know is that $S'$ is a synchronous system and $\varpi'$ has a well defined behavior.

Let $\beta_1$ be an execution of $\varpi$ in $G'$ where $C$ is an honest player. $\mathcal{A}$ corrupts $A$,$D$ in byzantine fashion and controls $B$ passively. Here $B$ is the general and starts with input value 0. Similarly let $\beta_2$ be the execution of $\varpi$ in which $C$,$D$ are honest players. $\mathcal{A}$ corrupts $A$ passively and $B$ in byzantine fashion. Here $B$ is the general. $B$ sends a 1 to $A$,$D$ and a 0 to $C$. Let $\beta_3$ be an execution of $\varpi$ in which $A$,$D$ are honest players. $\mathcal{A}$ controls $B$ passively, corrupts $C$ in byzantine fashion. Here $B$ is the general and starts with input value 1. Let $\beta$ be an execution of $\varpi'$ in $S'$ in which each player starts with input value as shown in Figure 2. All the players in $\beta$ are honest and follow the designated protocol correctly. We now show that whatever view [equation 2] $B$,$C$ get in $\beta$, $\mathcal{A}$ can generate the same view for $B$,$C$ in $\beta_1$. Similarly we prove that whatever view $C$,$D$,$A'$ get in $\beta$, $\mathcal{A}$ can generate the same view for $C$,$D$,$A$ in $\beta_2$ and whatever view $A'$,$B'$,$D$ get in $\beta$, $\mathcal{A}$ can generate the same view for $A$,$B$,$D$ in $\beta_3$.

We now give the adversary strategy in executions $\beta_1$:

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 $\beta_2$. 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 $\beta_2$. 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 $\beta_2$. 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 upto receive phase of round $i-1$.)

We now argue that the messages received by $B$,$C$ in round $i$ of $\beta$ are same as the messages received by $B$,$C$ respectively in round $i$ of $\beta_1$.

**Lemma 5** *$msg_i^{\beta}(x,B)_x \sim msg_i^{\beta_1}(x,B)_x$ 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 prove using induction. Basic technique is similar to one used in Appendix A of [17]. We prove that for any round $i$, $B$ and $C$ receive same messages in executions $\beta$ and $\beta_1$ respectively. Note that we have to show that adversary can same messages can be sent no matter for how many rounds protocol is run. Note that what players send in round $k$ is also dependent on what they recieve in round $k-1$ which in turn in also dependent on they in turn recieve in round $k-2$ and so on. Note that this continues in a recursive manner until recursion stops at round 1. The entire recursion can be visualized as trees which we refer to as execution trees $T_{\beta}^{B}$ and $T_{\beta_1}^{B}$ as one shown in Figure 5. We now formally describe tree $T_{\beta}^{x}$. 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 $\beta$. All edges are directed from child to parent and are between adjacent levels only. Observe that for the proof to go through, in-degree for any node $y'$(or $y$) in system $S'$ has to be same as in-degree of corresponding node $y$ in $G'$. Thus structurally both trees $T_{\beta}^{x'}$(or $T_{\beta}^{x}$) and $T_{\beta_1}^{x}$ will be exactly same (A node $y'$ in $T_{\beta}^{x}$ is replaced by its corresponding node $y$ in $T_{\beta_1}^{x}$). Now consider a node $b'$(or $b$) at level $j$ in $T_{\beta}^{x}$. Then its corresponding node at level $j$ in $T_{\beta_1}^{x}$ is $b$. Note that if the messages received by $b'$ in $T_{\beta}^{x}$ is same as those received by $b$ in $T_{\beta_1}^{x}$ and both $b'$ and $b$ start with same input value, same private key and run same code then both will send same messages.

To prove that the view [Definition 2] of $B$ is same in $\beta$ and $\beta_1$ we apply induction on heights of $T_{\beta}^{B}$ and $T_{\beta_1}^{B}$. Similarly using $T_{\beta}^{C}$ and $T_{\beta_1}^{C}$, we show view of $C$ is same in $\beta$ and $\beta_1$. Note that only nodes present in $T_{\beta}^{B}$ are $B, C, D, A', B'$. Corresponding nodes present in $T_{\beta_1}^{B}$ are $B, C, D, A, B$ respectively. We analyze these trees in bottom up manner. Consider trees $T_{\beta}^{B}$, $T_{\beta}^{C}$ and $T_{\beta_1}^{B}$, $T_{\beta_1}^{C}$ at the end of round 1 as shown in Figure 3. We claim that $B$ in $\beta$ and $\beta_1$ receive similar messages at the end of round 1. Consider (a) in Figure 3. $C$ starts with same input, secret key and executes same code in $\beta$ and $\beta_1$. Thus it will send same messages to $B$ in round 1 of $\beta$ and $\beta_1$ i.e. $msg_1^{\beta}(C,B)_C \sim msg_1^{\beta_1}(C,B)_C$. Since $A$ and $D$ are faulty in $\beta_1$, using aforementioned adversary strategy $\mathcal{A}$ 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 $\beta$ and $\beta_1$. Similarly one can show that $C$ also gets same messages at the end of round 1 in $\beta$ and $\beta_1$.

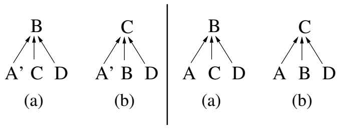

Figure 3: $T_{\beta}^{B}$, $T_{\beta}^{C}$ and $T_{\beta_1}^{B}$, $T_{\beta_1}^{C}$ at the end of round 1.

We now claim that the similarity holds for round 2 as well i.e. $msg_2^{\beta}(x,B)_x \sim msg_2^{\beta_1}(x,B)_x$ and $msg_2^{\beta}(x,C)_x \sim msg_2^{\beta_1}(x,C)_x$, $\forall x \in \mathbb{P}$.

Consider trees $T_{\beta}^{B}$, $T_{\beta}^{C}$ and $T_{\beta_1}^{B}$, $T_{\beta_1}^{C}$ at the end of round 2 as shown in Figure 4. Consider node $C$ at level 1 in $T_{\beta}^{B}$ and $T_{\beta_1}^{B}$. Node $B$ starts with same input value, secret key and execute same code in both $\beta$ and $\beta_1$ respectively, thus $msg_1^{\beta}(B,C)_B \sim msg_1^{\beta_1}(B,C)_B$. Since $A$,$D$ are faulty, $\mathcal{A}$ can ensure that $msg_1^{\beta}(A',C)_{A'} \sim msg_1^{\beta_1}(A,C)_A$ and $msg_1^{\beta}(D,C)_D \sim msg_1^{\beta_1}(D,C)_D$. Thus $C$ receives same messages at the end of round 1 in $\beta$ and $\beta_1$. Since $C$ starts with same input value, secret key and execute same code in both $\beta$ and $\beta_1$ respectively, it sends same message to $B$ in round 2 i.e. $msg_2^{\beta}(C,B)_C \sim msg_2^{\beta_1}(C,B)_C$. Now consider $A'$ at level 2 in $T_{\beta}^{B}$ and $A$ at level 2 in $T_{\beta_1}^{B}$. $B'$ in $\beta$ starts with a different input from $B$ in $\beta_1$, thus $msg_1^{\beta}(B',A')_{B'} \nsim msg_1^{\beta_1}(B,A)_B$. However since $A$ is faulty and $B$ is passively corrupt in $\beta_1$, $\mathcal{A}$ on behalf of $B$ can construct $msg_1^{\beta_1}(B,A)_B$ such that $msg_1^{\beta}(B',A')_{B'} \sim msg_1^{\beta_1}(B,A)_B$. $C$ starts with same input value, secret key and execute same code in both $\beta$ and $\beta_1$ respectively, thus $msg_1^{\beta}(C,A')_C \sim msg_1^{\beta_1}(C,A)_C$. Since $D$ is faulty, $\mathcal{A}$ can ensure that $msg_1^{\beta}(D,A')_D \sim msg_1^{\beta_1}(D,A)_D$. Thus $A'$ in $\beta$ receives same messages at the end of round 1 as $A$ in $\beta_1$. Since $A$ is faulty in $\beta_1$, $\mathcal{A}$ can ensure that $A$ in $\beta_1$ sends message to $B$ in round 2 same as what $A'$ in $\beta$ sends to $B$ in round 2 i.e. $msg_2^{\beta}(A',B)_{A'} \sim msg_2^{\beta_1}(A,B)_A$. Similarly one can show that $msg_2^{\beta}(D,B)_D \sim msg_2^{\beta_1}(D,B)_D$. Thus $msg_2^{\beta}(x,B)_x \sim msg_2^{\beta_1}(x,B)_x$, $\forall x \in \mathbb{P}$. Similarly one can argue for $msg_2^{\beta}(x,C)_x \sim msg_2^{\beta_1}(x,C)_x$, $\forall x \in \mathbb{P}$.

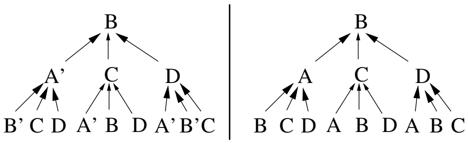

Figure 4: $T_{\beta}^{B}$ and $T_{\beta_1}^{B}$ at the end of round 2.

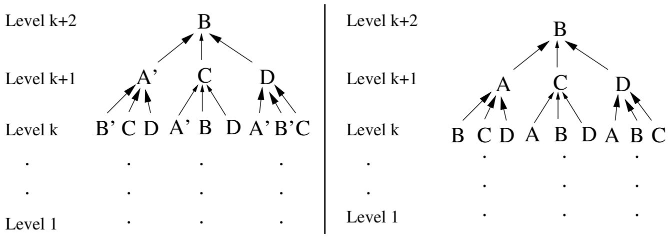

Figure 5: $T^B_\beta$ and $T^B_{\beta_1}$ at the end of $k+1$ rounds.

Let the similarity be true till some round $k$ i.e. $msg^{\beta}_i(x, B)_x \sim msg^{\beta_1}_i(x, B)_x$ and $msg^{\beta}_i(x, C)_x \sim msg^{\beta_1}_i(x, C)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider $T^B_\alpha$ and $T^B_{\alpha_1}$ at the end of $k+1$ rounds as shown in Figure 5. To prove induction we need to show that $B$ at level $k+2$ receives same messages in both trees. Consider node $D$ at level $k+1$. From induction hypothesis $C$ receive same messages till round $k$ in both trees. Also since $C$ starts with same input value, secret key and execute same code in both $\beta$ and $\beta_1$ 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 $\beta_1$ same as what $A$ receives till round $k$ in $\beta$. Since $A$ is faulty in $\beta_1$, $\mathcal{A}$ can ensure that $A$ sends same message to $D$ in $\beta_1$ as $A'$ sends to $D$ in $\beta$ 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 $\beta_1$ same as what $B$ receives messages till round $k$ in $\beta$. But $B$ in $\beta_1$ starts with a different input from $B'$ in $\beta$, thus they send different messages to $D$ in $\beta$ and $\beta_1$. However since $D$ is faulty and $B$ is passively corrupt in $\beta_1$, $\mathcal{A}$ 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 $T^B_\alpha$ and $T^B_{\alpha_1}$. Since $D$ is faulty in $\beta_1$, $\mathcal{A}$ 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 $\beta$ and $\beta_1$. Thus induction hypothesis holds for round $k+1$ too. Thus $msg^{\beta}_i(x, B)_x \sim msg^{\beta_1}_i(x, B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ holds true. Similarly one can argue for $msg^{\beta}_i(x, C)_x \sim msg^{\beta_1}_i(x, C)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$. The above proof is based on assumptions that $A'$ upto level $k$ in $T^B_\alpha$ 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 $B'$ upto level $k$ in $T^B_\alpha$ receives same messages as corresponding $B$ in $T^B_{\beta_1}$. ■

**Lemma 6** *$view^{\beta}_B \sim view^{\beta_1}_B$ and $view^{\beta}_C \sim view^{\beta_1}_C$*

*Proof*: Follows from equation 3 and Lemma 5. ■

We now give adversary strategy for $\beta_2$ and $\beta_3$ respectively. For $\beta_2$:

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 $\beta_1$. Similarly $\mathcal{A}$ sends to $D$ what an honest $B$ would have sent to $D$ in round 1 of $\beta_3$ and $\mathcal{A}$ sends to $A$ what an honest $B$ would have sent to $A$ in round 1 of $\beta_3$. 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 $\beta_1$. Similarly $\mathcal{A}$ authenticates and sends same message to $D$ as an honest $B$ would have sent to $D$ in $\beta_3$. 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 $msg_{i-1}^{\beta_2}(A,B)_A$ 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 $msg_i^{\beta_2}(A,B)_A$, $msg_i^{\beta_2}(C,B)_C$ and $msg_i^{\beta_2}(D,B)_D$ from $B$ in $\beta_2$ (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 upto receive phase of round $i-1$.). Similarly $\mathcal{A}$ obtains $msg_i^{\beta_2}(B,A)_B$, $msg_i^{\beta_2}(C,A)_C$ and $msg_i^{\beta_2}(D,A)_D$ from $A$ in $\beta_2$ (These are round $i$ messages sent by $B$,$C$,$D$ to $A$).

Adversary strategy for $\beta_3$:

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 $\beta_2$. For $i \geq 2$ $\mathcal{A}$ authenticates $msg_{i-1}^{\beta_3}(A,C)_A$ using secret key of $C$ and sends it to $B$,$D$. Similarly it authenticates $msg_{i-1}^{\beta_3}(D,C)_D$ using $C$'s secret key and sends it to $A$,$B$. For $msg_{i-1}^{\beta_3}(B,C)_B$, $\mathcal{A}$ 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 $\beta_2$ and sends same to $D$ as an honest $C$ would have sent to $D$ in execution $\beta_2$. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\beta_3}(B,C)_B$, such that $msg_{i-1}^{\beta_3}(B,C)_B \sim msg_{i-1}^{\beta_2}(B,C)_B$ authenticates it using $C$'s key and sends it to $A$,$D$. If $msg_{i-1}^{\beta_3}(B,C)_B$ 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 $msg_i^{\beta_3}(A,C)_A$, $msg_i^{\beta_3}(B,C)_B$ and $msg_i^{\beta_3}(D,C)_D$ via $C$. (These are round $i$ messages sent by $A$,$B$ and $D$ to $C$). Similarly $\mathcal{A}$ obtains $msg_i^{\beta_3}(A,B)_A$, $msg_i^{\beta_3}(C,B)_C$ and $msg_i^{\beta_3}(D,B)_D$ 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 $i-1$.)

Using aforementioned adversary strategies and technique similar to one used in proof of Lemma 5, 6 one can prove the following four lemmas. Due to space constraints proofs are omitted.

**Lemma 7** *$msg_i^{\beta}(x,C)_x \sim msg_i^{\beta_2}(x,C)_x$, $msg_i^{\beta}(x,D)_x \sim msg_i^{\beta_2}(x,D)_x$ and $msg_i^{\beta}(x,A')_x \sim msg_i^{\beta_2}(x,A)_x$ $\forall i > 0$, $\forall x \in P$.*

**Lemma 8** *$view_C^{\beta} \sim view_C^{\beta_2}$, $view_D^{\beta} \sim view_D^{\beta_2} and view_{A'}^{\beta} \sim view_A^{\beta_2}$*

**Lemma 9** *$msg_i^{\beta}(x,A')_x \sim msg_i^{\beta_3}(x,A)_x$, $msg_i^{\beta}(x,B')_x \sim msg_i^{\beta_3}(x,B)_x$, and $msg_i^{\beta}(x,D)_x \sim msg_i^{\beta_3}(x,D)_x$, $\forall i > 0$, $\forall x \in P$.*

**Lemma 10** *$view_{A'}^{\beta} \sim view_A^{\beta_3}$, $view_{B'}^{\beta} \sim view_B^{\beta_3}$, $view_D^{\beta} \sim view_D^{\beta_3}$.*

**Lemma 11** *There does not exists any protocol solving $ABG_{mix}$ over a complete graph of four nodes(G') tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$.*

*Proof*: Proof by contradiction. Let there exists a protocol $\varpi$ solving $ABG_{mix}$ over a complete graph of four nodes tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. From $\varpi$ we construct a protocol $\varpi'$ [Definition 2] for system $S'$. $\beta$ is an execution of $\varpi'$ as shown in Figure 2. $\beta_1$ is an execution of $\varpi$ in which $A$,$D$ are faulty, $C$ is honest and $B$ is passively corrupt. Both $B$,$C$ start with input value 0, and since $\varpi$ solves $ABG_{mix}$, from validity condition both $B$,$C$ must eventually output 0. From Lemma 6, for $B$,$C$, $\beta_1$ is indistinguishable from $\beta$ i.e. $\beta \overset{B}{\sim} \beta_1$ and $\beta \overset{C}{\sim} \beta_1$. Thus, $B$,$C$ in $\beta$ will eventually decide on value 0 (We are able to make claims regarding player's outputs in $\beta$ as views of players are same in $\beta$ and $\beta_1$. Thus by analyzing player's outputs in $\beta_1$, we can determine their outputs in $\beta$). Similarly using Lemma 10 $A'$,$B'$,$D$ cannot distinguish between $\beta$ and $\beta_3$ i.e. $\beta \overset{A'}{\sim} \beta_3$, $\beta \overset{B'}{\sim} \beta_3$ and $\beta \overset{D}{\sim} \beta_3$. Thus in $\beta$, $A'$,$B'$ and $D$ eventually agree on value 1. Now consider execution $\beta_2$. $B$ is byzantine corrupt, $A$ is passively corrupt and $C$,$D$ are honest. $A$,$C$ and $D$ start with input values 1,0,1 respectively. Since, $\varpi$ solves $ABG_{mix}$, from agreement condition all three should decide on same value. From Lemma 8, $\beta \overset{A'}{\sim} \beta_2$, $\beta \overset{C}{\sim} \beta_2$ and $\beta \overset{D}{\sim} \beta_2$. Thus $A'$,$C$ and $D$ should output same value in $\beta$ as in $\beta_2$. However in $\beta$, $C$ has already decided on 0 and $A'$,$D$ have already decided on 1. This leads to a contradiction in $\varpi'$. Using Lemma 4, we can say that our assumption that there exists a protocol $\varpi$ solving $ABG_{mix}$ over a complete graph of four nodes tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$ is wrong. ■

We now give the main theorem of this paper.

**Theorem 12** *There does not exists any protocol solving $ABG_{mix}$ over a complete graph $G$ of $n$ nodes tolerating ($t_b$,$t_p$)-adversary if $n \leq 2t_b + min(t_b,t_p)$, for $|t_p| > 0$.*

*Proof*: Proof by contradiction. We assume there exists a protocol $\eta$ solving $ABG_{mix}$ tolerating ($t_b$,$t_p$) adversary when $n \leq 2t_b + min(t_b,t_p)$. We show how to transform $\eta$ into a solution $\eta'$ which solves $ABG_{mix}$ for four players completely connected, tolerating $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. Divide $n$ players in $\eta$ 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 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 $\eta'$ simulate players in $I_A, I_B, I_C, I_D$ respectively. Each player $i$ in $\eta'$ 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 show that $\eta'$ solves $ABG_{mix}$ tolerating $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. For simplicity we assign any actively and passively corrupted players of $\eta$ to be exactly those that are simulated by actively and passively corrupted player in $\eta'$. Let $\psi'$ be an execution of $\eta'$ with the faults characterized by $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. Let $\psi$ be an execution of $\eta$. As per our assumption $\psi$ solves $ABG_{mix}$, thus $\psi$ satisfies termination, agreement and validity conditions [Definition 1]. We now show that same holds for $\psi'$ if it holds for $\psi$. In $\psi$, let the general be from set $I_k$, then in $\psi'$, player $k$ acts as the general. Note that in $\psi$ if $I_k$ is controlled actively or passively by the adversary, then so is $k$ is $\psi'$. Let $j$,$l$ ($j \neq l$) be two non-faulty players in $\psi'$. $j$ and $l$ simulates at least one player each in $\psi$. w.l.o.g let them simulate players in $I_j$, $I_l$.

Since $j$ and $l$ are non-faulty, so are all players in $I_j$, $I_l$. For $\psi$, all players in $I_j$, $I_l$ must terminate, then so should $j$ and $l$. In $\psi$, all non-faulty players including $I_j$, $I_l$ should agree on same value say $u$, then in $\psi'$, $j$, $l$ also agree on $u$. In $\psi$, if the general is non-faulty and starts with value $v$, then in $\psi'$ too, general will be non-faulty and starts with value $v$. In such a case in $\psi$, all non-faulty players including $I_j$, $I_l$ should have $u = v$, then in $\psi'$, $j$, $l$ should have $u = v$. Thus $\psi'$ also satisfies termination, validity and agreement conditions. Then $\eta'$ should solve $ABG_{mix}$ tolerating $\mathbb{A} = \{((A, D), (B)), ((B), (A)), ((C), (B))\}$. But from Lemma 11, we know that there does not exists any protocol solving $ABG_{mix}$ tolerating $\mathbb{A} = \{((A, D), (B)), ((B), (A)), ((C), (B))\}$. Thus our assumption that there exists a solution $\eta$ solving $ABG_{mix}$ for $n \leq 2t_b + min(t_b, t_p)$ is wrong. ■

## 6.2 Sufficiency

We now give protocol for $n > 2t_b$+min$(t_b, t_p)$. Given $(t_b, t_p)$, one can always find if $t_p < t_b$ or $t_p \geq t_b$. First consider the case of $t_p < t_b$. Then $n > 2t_b$+min$(t_b, t_p)$ reduces to $n > 2t_b + t_p$. The proposed protocol for this is obtained by a sequence of transformations on *EIG* [2]. A detailed description of the construction of *EIG* tree is available in [21, page 108]. The General sends his input to every player. Each player starts with this as input value and exchanges messages with others as per *EIGStop* protocol in [21, page 110] for $t_b + t_p + 1$ rounds.

**Definition 3 (Prune(*EIG*)) Prune**(EIG) *is a 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.*

At the end of $t_b + t_p + 1$ rounds of *EIGStop* protocol, we invoke **Prune**(*EIG*). Player $i$ applies the following decision rule. Namely, Player $i$ takes a majority of the values at the first level [5] of its *EIG* tree (note that he does not need to take a majority over the entire *EIG* tree). If a majority exists player $i$ decides on that value; otherwise, $i$ decides on a *default value*, $v_0$.

**Lemma 13** *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.*

*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 14** *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

[5] all nodes with labels $l$ such that $l \in \mathbb{P}$.

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 15** *$subtree_j{}^i$ 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 14 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. ■

**Theorem 16** *For $n > 2t_b + t_p$,* **EIG** *algorithm given above solves $ABG_{mix}$.*

*Proof:* Termination is obvious, by the decision rule. Note that $n - (t_b + t_p)$ represent 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 starts with $v$, then all the non-faulty players also start with $v$. The decision rule ensures that in case the General starts with $v$, $v$ is the only possible decision value. Thus validity also holds. For agreement, let $i$ and $j$ be any two non-faulty players that decide. 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 $subtree_j^i, \forall i \in \mathbb{P}$. 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 source being faulty, the agreement simply implies that all non-faulty players decide on same value. ■

For the case when $t_p \geq t_b$, $n > 2t_b$+min$(t_b, t_p)$ reduces to $n > 3t_b$. For such a case any protocol for unauthenticated Byzantine agreement (such as one given on [21, page 119]) works. This is because for unauthenticated setting $t_p = n - t_b$. This completes the sufficiency proof.

# 7 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 paper, 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] Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. Authenticated Byzantine Generals Strike Again. Technical report, Center for Security, Theory and Algorithmic Research (CSTAR), International Institute of Information Technology, Hyderabad, India, 2008. A complete version is available at http://research.iiit.ac.in/~anujgupta/BA_with_authentication_web_version.pdf.

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

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

[20] 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.

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

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

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

[24] 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.

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