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.* [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 $u = v$. In 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 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, 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 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

[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.

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 $n > t$ [21], 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 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 $n > t$ [21] 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 (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 $u = v$.*

[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 [19].

- 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.

4 Motivating Example

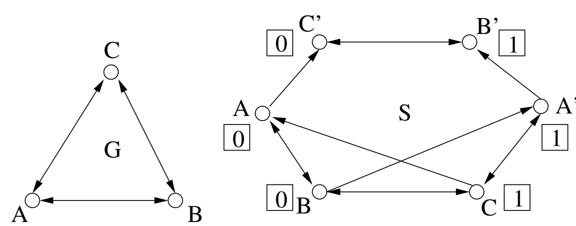

Figure 1: System G and S.

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 be $n > t_b+t_p$ as is the case with $n > t$ result for ABG [21]. 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.

We start by assuming that there exists a protocol π 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 π we create a protocol π' (executed by each player in S) in such a way that if π exists then so does π'. Further, let α_1 be an execution of π 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 α_2 be the execution of π 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 α_3 be an execution of π 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 α be an execution of π' in S in which each player starts with input value as shown in Figure 1. All the players in α are honest and follow the prescribed protocol(π') 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 α, $\mathcal{A}$ can generate the same view for A, B in α_1. Since $\mathcal{A}$ can ensure that view of both A, B is same in α and α_1, we show that A, B in α will decide on value 0. Similarly one can prove that A', C in α will decide on value 1. On similar lines we then prove that since B, C in α_2 should agree on same value, then so should B, C in α, but B,C have already decided upon values 0 and 1 respectively in α, leading to a contradiction in π'. This contradicts our original assumption about existence of π.

To complete the proof sketch, we now give an idea as to how $\mathcal{A}$ can ensure that A, B gets same view in α_1 and α. Consider an execution Γ of π' in S which is exactly same as α except that in Γ A' starts with input value 0. Since in α, 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 start with input value 1 and follow the designated protocol). 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' starts with input value 0 in Γ and input value 1 in α). 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 and 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 S, 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 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.

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

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, 19]. 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 α. In Appendix A we elaborate as to why the proof for Lemmas 1, 2, 3 breaks down, in case this constraint in execution α 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 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 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 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 [3], and if for every round $1 \dots i$ their corresponding incoming messages are same, then their views till round i will also be same. [4] Formally:

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

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.

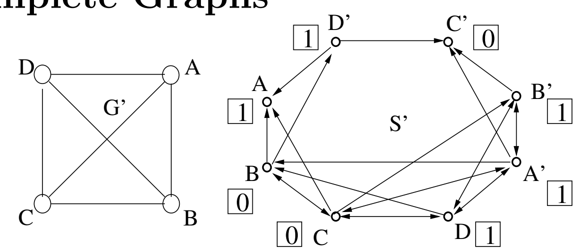

Figure 2: System G' and S'.

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 ϖ 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 ϖ'. We now formally define ϖ' and further prove that if ϖ exists then so does ϖ'.

Definition 2 **(ϖ')** *For all players $a, b \in \mathbb{P}$, any statement of kind "b sends message m to a" in ϖ 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 ϖ'. Rest all statements in ϖ' are same as ϖ.*

Lemma 4 *If ϖ exists then ϖ' 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

[3]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

[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.

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(ϖ). 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 ϖ' 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 ϖ'. All we know is that S' is a synchronous system and ϖ' has a well defined behavior.

Let β_1 be an execution of ϖ 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 β_2 be the execution of ϖ 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 β_3 be an execution of ϖ 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 β be an execution of ϖ' in S' in which each player starts with input value as shown in Figure 2. All the players in β are honest and follow the designated protocol correctly. We now show 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 in β_2 and whatever view A',B',D get in β, $\mathcal{A}$ can generate the same view for A,B,D in β_3.

We now give the adversary strategy in executions β_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 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 upto receive phase of round $i-1$.)

We now argue 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.

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. We prove that for any round i, B and C receive same messages in executions β and β_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^B_β and $T^B_{\beta_1}$ as one shown in Figure 5. We now formally describe tree T^x_β. 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 β. 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 $T^{x'}_\beta$ (or T^x_β) and $T^x_{\beta_1}$ will be exactly same (A node y' in T^x_β is replaced by its corresponding node y in $T^x_{\beta_1}$). Now consider a node b'(or b) at level j in T^x_β. Then its corresponding node at level j in $T^x_{\beta_1}$ is b. Note that if the messages received by b' in T^x_β is same as those received by b in $T^x_{\beta_1}$ 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 β and β_1 we apply induction on heights of T^B_β and $T^B_{\beta_1}$. Similarly using T^C_β and $T^C_{\beta_1}$, we show view of C is same in β and β_1. Note that only nodes present in T^B_β are B, C, D, A', B'. Corresponding nodes present in $T^B_{\beta_1}$ are B, C, D, A, B respectively. We analyze these trees in bottom up manner. Consider trees T^B_β, T^C_β and $T^B_{\beta_1}$, $T^C_{\beta_1}$ at the end of round 1 as shown in Figure 3. We claim that B in β and β_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 β and β_1. Thus it will send same messages to B in round 1 of β and β_1 i.e. $msg^\beta_1(C,B)_C \sim msg^{\beta_1}_1(C,B)_C$. Since A and D are faulty in β_1, using aforementioned adversary strategy $\mathcal{A}$ can ensure that $msg^\beta_1(A',B)_{A'} \sim msg^{\beta_1}_1(A,B)_A$ and $msg^\beta_1(D,B)_D \sim msg^{\beta_1}_1(D,B)_D$. Thus B gets same messages at the end of round 1 in β and β_1. Similarly one can show that C also gets same messages at the end of round 1 in β and β_1.

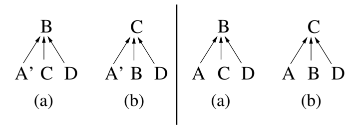

Figure 3: T^B_β, T^C_β and $T^B_{\beta_1}$, $T^C_{\beta_1}$ at the end of round 1.

We now claim that the similarity holds for round 2 as well i.e. $msg^\beta_2(x,B)_x \sim msg^{\beta_1}_2(x,B)_x$ and $msg^\beta_2(x,C)_x \sim msg^{\beta_1}_2(x,C)_x$, $\forall x \in \mathbb{P}$.

Consider trees T^B_β, T^C_β and $T^B_{\beta_1}$, $T^C_{\beta_1}$ at the end of round 2 as shown in Figure 4. Consider node C at level 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 $msg^\beta_1(A',C)_{A'} \sim msg^{\beta_1}_1(A,C)_A$ and $msg^\beta_1(D,C)_D \sim msg^{\beta_1}_1(D,C)_D$. 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 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 $msg^\beta_1(B',A')_{B'} \sim msg^{\beta_1}_1(B,A)_B$. 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 $msg^\beta_1(D,A')_D \sim msg^{\beta_1}_1(D,A)_D$. 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. $msg^\beta_2(A',B)_{A'} \sim msg^{\beta_1}_2(A,B)_A$. Similarly one can show that $msg^\beta_2(D,B)_D \sim msg^{\beta_1}_2(D,B)_D$. Thus $msg^\beta_2(x,B)_x \sim msg^{\beta_1}_2(x,B)_x$, $\forall x \in \mathbb{P}$. Similarly one can argue for

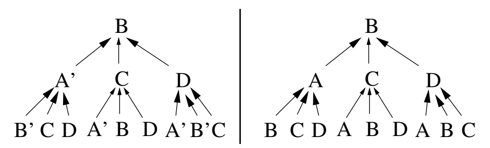

Figure 4: T^B_β and $T^B_{\beta_1}$ at the end of round 2.

$msg_2^{\beta}(x, C)_x \sim msg_2^{\beta_1}(x, C)_x$, $\forall x \in \mathbb{P}$.

Let the similarity be true till some round k i.e. $msg_i^{\beta}(x, B)_x \sim msg_i^{\beta_1}(x, B)_x$ and $msg_i^{\beta}(x, C)_x \sim msg_i^{\beta_1}(x, C)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T_{α}^{B} and $T_{\alpha_1}^{B}$ 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 β and β_1 respectively, it sends same messages to D in round k i.e. $msg_k^{\beta}(C, D)_C \sim msg_k^{\beta_1}(C, D)_C$. 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. $msg_k^{\beta}(A', D)_{A'} \sim msg_k^{\beta_1}(A, D)_A$. 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 $msg_k^{\beta}(B', D)_{B'} \sim msg_k^{\beta_1}(B, D)_B$. Thus D at level $k+1$ receives same messages in T_{α}^{B} and $T_{\alpha_1}^{B}$. Since D is faulty in β_1, $\mathcal{A}$ can ensure that $msg_{k+1}^{\beta}(D, B)_D \sim msg_{k+1}^{\beta_1}(D, B)_D$. Using similar arguments one can show that $msg_{k+1}^{\beta}(C, B)_C \sim msg_{k+1}^{\beta_1}(C, B)_C$ and $msg_{k+1}^{\beta}(A', B)_{A'} \sim msg_{k+1}^{\beta_1}(A, B)_A$. Thus B receives same messages in round $k+1$ of β and β_1. Thus induction hypothesis holds for round $k+1$ too. Thus $msg_i^{\beta}(x, B)_x \sim msg_i^{\beta_1}(x, B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ holds true. Similarly one can argue for $msg_i^{\beta}(x, C)_x \sim msg_i^{\beta_1}(x, C)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$. The above proof is based on assumptions that A' upto level k in T_{α}^{B} receives same messages as corresponding A in $T_{\beta_1}^{B}$. 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_{α}^{B} receives same messages as corresponding B in $T_{\beta_1}^{B}$. ■

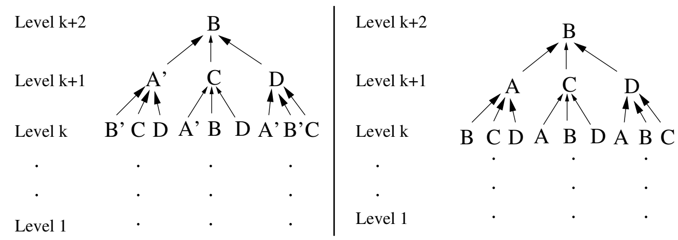

Figure 5: T_{β}^{B} and $T_{\beta_1}^{B}$ at the end of $k+1$ rounds.

Lemma 6 *$view_B^{\beta} \sim view_B^{\beta_1}$ and $view_C^{\beta} \sim view_C^{\beta_1}$*

Proof: Follows from equation 3 and Lemma 5. ■

We now give adversary strategy for β_2 and β_3 respectively. 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 $msg_{i-1}^{\beta_2}(C, B)_B$ using B's secret key and sends it to A,D. Similarly, $\mathcal{A}$ authenticates $msg_{i-1}^{\beta_2}(D, B)_D$ using B's secret key and sends it to A,C. For $msg_{i-1}^{\beta_2}(A, B)_A$, $\mathcal{A}$ 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 $msg_{i-1}^{\beta_2}(A, B)_A$, such that $msg_{i-1}^{\beta_2}(A, B)_A \sim msg_{i-1}^{\beta_1}(A, B)_A$, authenticates it using B's key and sends it to C. Similarly $\mathcal{A}$ constructs $msg_{i-1}^{\beta_2}(A, B)_A$, such that $msg_{i-1}^{\beta_2}(A, B)_A \sim msg_{i-1}^{\beta_3}(A, B)_A$, 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 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 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 β_2 (These are round i messages sent by B,C,D to A).

Adversary strategy 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 $msg_{i-1}^{\beta_3}(A,C)_A$ 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 β_2 and sends same to D as an honest C would have sent to D in execution β_2. Formally, $\mathcal{A}$ constructs $msg_{i-1}^{\beta_3}(B,C)_B$, 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 *$view_C^{\beta} \sim view_C^{\beta_2}$, $view_D^{\beta} \sim view_D^{\beta_2}$andview$_{A'}^{\beta} \sim view_A^{\beta_2}$*

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 ϖ solving ABG_{mix} over a complete graph of four nodes tolerating adversary basis $\mathbb{A} = \{((A,D),(B)),((B),(A)),((C),(B))\}$. From ϖ we construct a protocol ϖ' [Definition 2] for system S'. β is an execution of ϖ' as shown in Figure 2. β_1 is an execution of ϖ in which A,D are faulty, C is honest and B is passively corrupt. Both B,C start with input value 0, and since ϖ solves ABG_{mix}, from validity condition both B,C must eventually output 0. From Lemma 6, for B,C, β_1 is indistinguishable from β i.e. $\beta \overset{B}{\sim} \beta_1$ and $\beta \overset{C}{\sim} \beta_1$. Thus, B,C in β will eventually decide on value 0 (We are able to make claims regarding player's outputs in β as views of players are same in β and β_1. Thus by analyzing player's outputs in β_1, we can determine their outputs in β). Similarly using Lemma 10 A',B',D cannot distinguish between β and β_3 i.e. $\beta \overset{A'}{\sim} \beta_3$, $\beta \overset{B'}{\sim} \beta_3$ and $\beta \overset{D}{\sim} \beta_3$. Thus in β, A',B' and D eventually agree on value 1. Now consider execution β_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, ϖ 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 β as in β_2. However in β, C has already decided on 0 and A',D have already decided on 1. This leads to a contradiction in ϖ'. Using Lemma 4, we can say that our assumption that there exists a protocol ϖ 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 η solving ABG_{mix} tolerating (t_b,t_p) adversary when $n \leq 2t_b + min(t_b, t_p)$. We show how to transform η into a solution η' which solves ABG_{mix} for four players completely connected, 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 $min(t_b, t_p), min(t_b, t_p), t_b, ($ $min(t_b, t_p))$. $\mathbb{A}$ 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 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 show that 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 termination, agreement and validity conditions [Definition 1]. We now show that same holds for ψ' if it holds for ψ. In ψ, let the general be from set I_k, then in ψ', player k acts as the general. Note that in ψ if I_k is controlled actively or passively by the adversary, then so is k is ψ'. Let j,l ($j \neq l$) be two non-faulty players in ψ'. j and l simulates at least one player each in ψ. 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 ψ, all players in I_j, I_l must terminate, then so should j and l. In ψ, all non-faulty players including I_j, I_l should agree on same value say u, then in ψ', j, l 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_l should have $u = v$, then in ψ', j, l should have $u = v$. Thus ψ' also satisfies termination, validity and agreement conditions. Then η' 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 η 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 [20, 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 [20, 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 round one or change a value in Round k, $2 \le k \le t_b + t_p$, 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 $subtree^i_j, \forall i \in \mathbb{P}$. This implies they have the same set of

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

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 [20, 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] 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.

A Appendix

A.1 Impossibility of ABG_{mix} over G tolerating $(1,1)$-adversary

In section 4 we gave a proof sketch for impossibility of ABG_{mix} over a complete graph of three nodes tolerating a $(1,1)$-adversary. We now formally prove the same. We start by assuming that there exists a protocol π that solves ABG_{mix} for three players, $\mathbb{P} = \{A, B, C\}$, tolerating $(1,1)$-adversary. Let original graph of 3 players be G as shown in Figure 1. We construct a new system S, shown in Figure 1, using *two* copies of each player where each player runs some algorithm π'. We first formally define π' then prove that π' exists if π exists.

Definition 4 (π') *For all players $a, b \in \mathbb{P}$, 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') [1] which are connected by a directed edge from b to a" in π'. Rest all statements in π' are same as those in π.*

Lemma 17 *If π exists then π' exists.*

Proof: Implied from Definition 4. ■

Construction of S: Take two copies of each player in G and construct a hexagonal system S as shown in Figure 1. Player A is connected to B,C,C'; player B is connected to A,C,A'; C is connected to A,B,A'; A' is connected to B,C,B'; B' is connected to A',C' and C' is connected to A,B'. 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'. a and a' are independent copies of the player a with same authentication key. What we want to ensure is that 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. It is evident 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 algorithm(π). Also note that each player in S knows only its immediate neighbors and not the complete graph. Also, 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. Here we neither know what system S is supposed to do nor what π' solves. Since S does not form ABG_{mix} setting, therefore the definition of ABG_{mix} [Definition 1] does not tell us anything directly about the players' output in S. All we know is that S is a synchronous system and π' has a well defined behavior.

Let α_1 be an execution of π 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 α_2 be the execution of π in G in which B is an honest player. $\mathcal{A}$ corrupts C passively and A in byzantine fashion. Here A acts as the general. A sends 0 to B and 1 to C. Let α_3 be an execution of π 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 α be an execution of π' in S in which each player starts with input value as shown in Figure 1. Notice that all the players in α are honest and follow the prescribed protocol correctly.

We will show that some players in α do not always show a well defined behavior thus leading to a contradiction in π'. To do so we will prove that whatever *view* A, B get in α, $\mathcal{A}$ can generate the same view for A, B in α_1. On similar lines we 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. We define *view* mathematically as in equation **??**. We first formally give the adversary strategy in α_1:

[1] a and a' are independent copies of the player a with same authentication key.

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 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 upto 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 18 $msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$ *and* $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$.

Proof: We prove using induction. We prove that for any round i, whatever messages A,B receive in α $\mathcal{A}$ can ensure that A,B receive same messages in α_1 respectively. Note that what node A receives in round i of α depends on what nodes B and C send to it in round i of α. Similarly what node A receives in some round i of α_1 depends on what nodes B and C send to it in round i of α_1. So we need to argue that these messages sent in round i of α and α_1 are same or *can be* made same by adversary. In turn what B,C send in round i of α and α_1 depends on what they receive in previous round $i-1$. Thus we we need to argue that these messages sent in round $i-1$ of α and α_1 are same or can be made same by adversary. But what these send in round $i-1$ depends on what they receive respectively in round $i-2$. 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 8. In general this holds for any node x'(or x) in execution α of S and corresponding node x in execution α_1 of G.

We now formally describe 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. 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 $T_{\alpha}^{x'}$(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 a node 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' in T_{α}^{x} is same as those received by b in $T_{\alpha_1}^{x}$ and both b' and b start with same input value, same private key and run same code then both will send same messages.

We prove above theorem using induction on height of T_{α}^{B} and $T_{\alpha_1}^{B}$. Only nodes present in T_{α}^{B} are A, B, C, A'. Corresponding nodes present in $T_{\alpha_1}^{B}$ are A, B, C, A respectively. Notice that since B' does not appear in T_{α}^{B}, any A' in T_{α}^{A} or T_{α}^{B} has an outgoing directed edge only and only to C. We analyze these trees in bottom up manner. Consider round 1 of executions α and α_1. Consider trees T_{α}^{A}, T_{α}^{B} and $T_{\alpha_1}^{A}$, $T_{\alpha_1}^{B}$ at the end of round 1 as shown in

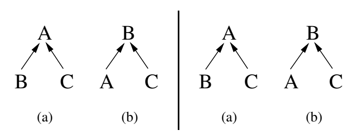

Figure 6: T_{α}^{A}, T_{α}^{B} and $T_{\alpha_1}^{A}$, $T_{\alpha_1}^{B}$ at the end of round 1.

Figure 6. We claim that A in α and α_1 receive similar messages at the end of round 1. Likewise B in α and α_1 respectively also receive similar messages at the end of round 1. Consider (a) in Figure 6. 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 strategy 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. Using arguments similar to those for (a), one can show that for (b), B also gets same messages at the end of round 1 in α and α_1.

We now claim that the similarity holds in round 2 as well i.e. $msg_2^{\alpha}(x,A)_x \sim msg_2^{\alpha_1}(x,A)_x$ and $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_1}(x,B)_x$, $\forall x \in \mathbb{P}$. Consider trees T_{α}^{A}, T_{α}^{B} and $T_{\alpha_1}^{A}$, $T_{\alpha_1}^{B}$ at the end of round 2 as shown in Figure 7.

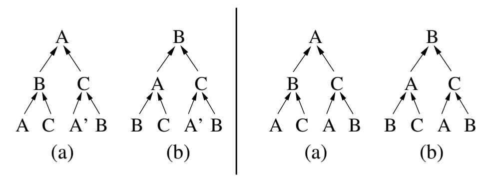

Figure 7: T_{α}^{A}, T_{α}^{B} and $T_{\alpha_1}^{A}$, $T_{\alpha_1}^{B}$ at the end of round 2.

Consider T_{α}^{A} and $T_{\alpha_1}^{A}$. Node A as well as B start with same input value, secret key and execute same code in both α and α_1 respectively, thus $msg_1^{\alpha}(A,B)_A \sim msg_1^{\alpha_1}(A,B)_A$ and $msg_1^{\alpha}(B,C)_B \sim msg_1^{\alpha_1}(B,C)_B$. Using aforementioned adversary strategy 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 A 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 $msg_2^{\alpha}(B,A)_B \sim msg_2^{\alpha_1}(B,A)_B$. Using aforementioned adversary strategy $\mathcal{A}$ 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. Similarly one can argue for $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_1}(x,B)_x$, $\forall x \in \mathbb{P}$.

Let the similarity be true till some round k i.e. $msg_i^{\alpha}(x,A)_x \sim msg_i^{\alpha_1}(x,A)_x$ and $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$, $\forall i | 1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of $k+1$ rounds as shown in Figure 8.

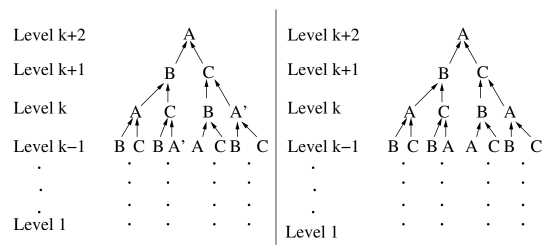

Figure 8: T_{α}^{A} and $T_{\alpha_1}^{A}$ at the end of $k+1$ rounds.

For proving induction 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 upto 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 $msg_k^{\alpha}(B,C)_B \sim msg_k^{\alpha_1}(B,C)_B$. 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' upto 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. Using similar ideas as used above one can show that $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_1}(x,B)_x$, $\forall i>0$, $\forall x \in \mathbb{P}$. ∎

Lemma 19 *$view_A^{\alpha} \sim view_A^{\alpha_1}$ and $view_B^{\alpha} \sim view_B^{\alpha_1}$*

Proof: Recall from equation 3, to show that view of A in α and α_1 are same, it is sufficient to show that for any round i messages received by A in α and α_1 respectively are same. This follows from Lemma 18 . Thus $view_A^{\alpha} \sim view_A^{\alpha_1}$ and $view_B^{\alpha} \sim view_B^{\alpha_1}$. ∎

We now formally give the adversary strategy in 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 $msg_{i-1}^{\alpha_2}(C,A)_A \sim msg_{i-1}^{\alpha_1}(C,A)_C$, 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 upto round $i-1$).

Lemma 20 *$msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$ and $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i>0$, $\forall x \in \mathbb{P}$*

Proof: Using adversary strategy in α_2, similar to proof of Lemma 18. Proof omitted. ∎

Lemma 21 *$view_B^{\alpha} \sim view_B^{\alpha_2}$ and $view_C^{\alpha} \sim view_C^{\alpha_2}$.*

Proof: Using Equation 3 and Lemma 20. ∎

Adversary strategy 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 upto round $i-1$).

Lemma 22 $msg_i^{\alpha}(x, C)_x \sim msg_i^{\alpha_3}(x, C)_x$ *and* $msg_i^{\alpha}(x, A')_x \sim msg_i^{\alpha_3}(x, A)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$

Proof: Using adversary strategy in α_3, similar to proof of Lemma 18. Proof omitted. ∎

Lemma 23 $view_{A'}^{\alpha} \sim view_{A}^{\alpha_3}$ *and* $view_{C}^{\alpha} \sim view_{C}^{\alpha_3}$.

Proof: Follows from Equation 3 and Lemma 22. ∎

Theorem 24 *There does not exists any protocol solving* ABG_{mix} *over a complete graph on 3 players tolerating* $(1, 1)$*-adversary.*

Proof: Proof by contradiction. We assume there exists a protocol π solving ABG_{mix} over a complete graph on 3 players influenced by a $(1, 1)$-adversary. Now consider execution α in system S where each player executes π'[Definition 4] . In α_1, C is faulty, B is honest and A is passively corrupt, and A is the general and starts with input 0, and since π solves ABG_{mix}, from the validity condition both A,B must eventually decide on 0. From Lemma 19, for A,B, α and α_1 are indistinguishable i.e. $\alpha \stackrel{A}{\sim} \alpha_1$ and $\alpha \stackrel{B}{\sim} \alpha_1$. Thus A,B in α will eventually decide on 0. (We are able to make claims regarding the outputs of A and B in α as their views are same as those in α_1. Thus by analyzing their outputs in α_1, we can determine there outputs in α.) Similarly in α_3, A is the general and starts with input 1, thus both A and C should output 1. Using Lemma 23, α and α_3 are indistinguishable to C,A' i.e. $\alpha \stackrel{C}{\sim} \alpha_3$ and $\alpha \stackrel{A'}{\sim} \alpha_3$. Thus C,A' in α should agree on 1. Now consider α_2. A is faulty, C is honest and B is passively corrupt, and A acts as general and sends different values to B and C. Since π solves ABG_{mix}, from agreement condition[Definition 1], both B and C should output the same value. Using Lemma 21, B,C in α should output same value, but B and C have already decided on values 0 and 1 respectively. This leads to a contradiction in π'. Thus there cannot exists a π' leading to impossibility of existence of π(from Lemma 17). ∎