Authenticated Byzantine Generals in Dual Failure Model

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 Byzantine Generals (BGP) to study the effects of Byzantine faults in distributed protocols for reliable broadcast. It is well known that BGP among n players tolerating up to t faults is (efficiently) possible if and only if $n > 3t$. To overcome this severe limitation, Pease *et al.* introduced a variant of BGP, *Authenticated Byzantine General* (ABG). Here players are supplemented with digital signatures (or similar tools) to thwart the challenge posed by Byzantine faults. Subsequently, they proved that with the use of authentication, fault tolerance of protocols for reliable broadcast can be amazingly increased to $n > t$ (which is a huge improvement over the $n > 3t$).

Byzantine faults are the most generic form of faults. In a network not *all* faults always malicious. Some faulty nodes may only leak their data while others are malicious. Motivated from this, we study the problem of ABG in (t_b,t_p)-mixed adversary model where the adversary can corrupt up to any t_b players actively and control up to any other t_p players passively. We prove that in such a setting, ABG over a completely connected synchronous network of n nodes tolerating a (t_b,t_p)-adversary is possible iff $n > 2t_b+\min(t_b, t_p)$ when $t_p > 0$. Interestingly, our results can also be seen as an attempt to unify the extant literature on BGP and ABG.

Keywords. Reliable broadcast, Authenticated Byzantine General, Mixed adversary.

1 Introduction

Fault tolerance of a distributed system is a highly desirable property, and has been the subject of intensive research for many years. A fundamental problem in the theory of distributed systems is that of designing protocols for simulating a broadcast channel over a point to point network in the presence of faults. The problem, introduced by Lamport *et al.* [LSP82], is popularly known as "Byzantine Generals problem"(BGP). 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 $u = v$. Traditionally, the notion of faults in the network is captured via a fictitious entity called *adversary* that can choose a subset of players as "pawns". An adversary that controls up to any t of the n players is denoted by t-adversary.

The problem was first studied on an all mighty t-adversary which can corrupt upto any t of the n nodes in *B*yzantine fashion. These Byzantine nodes have unlimited computational power and can behave arbitrarily, even colluding to bring the system down. In a completely connected synchronous network with no additional setup, classical results of [LSP82, PSL80] show that reliable broadcast among n parties in the presence of up to t Byzantine nodes 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. 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 BGP. After the semenial work of Pease *et al.* [LSP82, PSL80], studies were initiated under various settings like asynchronous networks [FLP85], partially synchronous networks [DDS87], incomplete networks [Dol81], hypernetworks [FM00], non-threshold adversaries [FM98], mixed-adversaries [AFM99], mobile adversaries [Gar94], and probabilistic correctness [Rab83] to name a few. An important variant of BGP is the *authenticated* model proposed by Pease *et al.* [PSL80], which as the title of this paper suggests, is our main focus. In this model, hereafter referred as *authenticated Byzantine General* (ABG), players are supplemented with "magical" powers (say a Public Key Infrastructure (PKI) and digital signatures) using which 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 [DS83] 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 [Bor95, Bor96b, ST87, Bor96a, KK09, GLR95, SW04].

1.1 Motivation

A large part of the literature considers the adversary to have same amount of control over all the corrupt players. However, many a times this may not be true. For example, some nodes may act in Byzantine manner while few others may atmost fail stop. In such a case modeling all the faults uniformly as Byzantine will be gross misrepresent. An elegant way to capture this is via a mixed adversary where by the adversary has varied control over different corrupt players i.e. it controls some players actively, another fraction as fail-stop. Note that the mixed adversary model not only generalizes the adversary models where only "monotype" corruption is considered but also facilitates a deeper understanding of the relationship between computability/complexity of the task at hand and the adversary's power. With respect to BGP, mixed adversary model has been considered in [GP92, AFM99] to name a few. Motivated from this, we initiate the study of ABG in a setting where a upto a fraction of the nodes are Byzantine and another fraction of nodes "leak" their internal state to the adversary. We model this via a $(t_b$,$t_p)$*−adversary*, whereby, the adversary controls upto any t_b players actively and up to another t_p players passively. We strive to answer the following: *what is the necessary and sufficient condition(s) for simulating a broadcast channel over a completely connected point-to-point synchronous network tolerating a* $(t_b$,$t_p)$*-adversary ?* Note that this is same as simulating a broadcast channel for the entire gamut of adversaries between $t_b = t$, $t_p = 0$ (ABG) and $t_b = t$, $t_p = n - t$ (BGP).

1.2 Our Contributions and Results

The contributions of this paper are manyfold:

1. *Better definition:* As a first contribution of this work, we argue that the problem of ABG under the influence of a $(t_b$,$t_p)$-adversary requires a slight modification in the standard definition of ABG available in the extant literature. Our argument stems from the observation that a protocol that satisfies the extant definition of ABG but does not meet our definition, *fails* to simulate a broadcast channel, as originally intended. Therefore, the definition of ABG available in the extant literature is not straightaway suitable in our setting. None the less, we essentially use the same principles to define a suitably adapted and faithful definition in our setting.

2. *Complete characterization:* We give the necessary and sufficient condition(s) for designing ABG protocols tolerating a (t_b,t_p)-adversary. We prove that over a completely connected synchronous network of n nodes with $t_p > 0$, $n > 2t_b + min(t_b, t_p)$ is necessary and sufficient for existence of ABG protocols tolerating a (t_b,t_p)-adversary. For $t_p = 0$, the bound is $n > t_b$ as given by Pease *et al.* [PSL80].

3. *Unification:* For the problem of reliable broadcast in the authenticated model (ABG), it is assumed that the adversary can forge signatures of only those players which are under its control. In contrast, the unauthenticated model (BGP), assumes no authentication tools. This can also be seen as all the players using *insecure signature schemes* and therefore the adversary can forge their signatures. With ABG tolerating a (t_b,t_p)-adversary, we initiate the study of the entire gamut of broadcasts in between, viz., an adversary that can forge signatures of up to any t_p nodes apart from controlling up to t_b nodes actively. Thus, BGP and ABG are merely two extreme points of this entire gamut. Our work gives a *characterization for the entire gamut.* Therefore, our work *unifies* the extant literature on BGP and ABG.

4. *Fault tolerance of signature schemes:* In the age of modern cryptography, it is reasonable to assume availability of Public Key Infrastructure (PKI) and digital signatures over any communication network. All known PKI and digital signature schemes are usually based on the conjectured hardness of some problems like integer factorization [RSA78], discrete logarithms [Gam85], permutations [Sha94, Sha85], lattice based problems [GPV08, Reg04] to name a few. Further, the proofs of the hardness of these problem seem to be beyond the reach of contemporary mathematics. Thus, it may well be the case that some of these schemes are eventually proven to be insecure.

   An elegant way to deal with this scenario is to consider the approach adopted by *robust combiners* [MPW07, HKN$^+$05, MP06]. Informally, a (k, n)-robust combiner is a construction that takes in n candidate protocols for a given functionality and combines them into one scheme such that even if up to any k of the n protocols are rendered incorrect during actual execution, the combined scheme is guaranteed to correctly implement the desired functionality. Note that different sets of up to k candidate protocols may fail for different executions/inputs.

   In context of ABG, different players may use different signature schemes. Since most known signature schemes in literature are based upon unproven assumptions, some of these schemes may eventually turn out to be insecure. Analogous to a (k, n)-robust combiner, one will prefer to design ABG protocols that work correctly even if up to a fraction of the signature schemes are rendered insecure. We capture this by assuming that the adversary can forge signatures of up to another t_p players. Thus, t_p can also be seen as a *robustness parameter of authentication* for ABG protocols.

1.3 Organization of the Paper

In section 2 we formally introduce our model. In section 2.1 we formally define the problem statement. Section 3 gives the complete characterization of ABG tolerating a (t_b, t_p)-adversary over completely connected synchronous networks, followed by the conclusion in section 4.

2 Our Model

We consider a set of n players denoted by $\mathbb{P}$, computationally unbounded and fully connected. Communication over the network is assumed to be synchronous. That is, the protocol is executed in a sequence of *rounds* where in each round, a player can perform some local computation, send new messages to all the players, receive messages sent to him by players in the same round, (and if necessary perform some more local computation), in that order. During the execution, the adversary may take control of up to any t_b+t_p players. The adversary can make t_b players to behave in any arbitrary fashion and read the internal states of upto another t_p players. W.l.o.g we assume that the adversary always uses his full power, therefore $t_b \cap t_p = \emptyset$. We further assume that the communication channel between any two players is perfectly reliable and authenticated i.e. the adversary cannot modify messages sent between non-malicious parties. We also assume existence of a signature/authentication scheme[1] where the sender signs the message to be sent. This is modeled by all parties 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 player's signature and the receiver can uniquely identify the sender of the message from 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 their messages with their private key.

2.1 Problem Definition

Consider a ABG protocol wherein a player, say P_i, is passively controlled by (t_b,t_p)-adversary. By virtue of passive corruption, adversary can always forge messages on behalf of P_i (adversary can read and thereafter use the private key used by P_i for authenticating its messages). In such a scenario, at the end of the ABG protocol, is P_i required to output value same decided upon by honest players ? At a first glance the answer may be NO. The rationale being: P_i has lost his private key to the adversary, therefore, in a way P_i is helping the adversary. Thus, any correct ABG protocol need not ensure passively corrupt players (such as P_i) to output a value same as honest players. However, in the sequel, we present a series of arguments to demonstrate that *any valid ABG protocol tolerating (t_b,t_p)-adversary is required to ensure that all passively corrupt players output same value as honest players.*

As a prelude, we remark that in literature, a player considered to be *faulty* if and only if that player deviates from the designated protocol. Consequently, a player can be non-faulty in two ways – first the adversary is absent and (therefore) player follows the protocol and second the adversary is present passively and (therefore) player follows the protocol. For the rest of this paper, we refer to the former kind of non-faulty player as *honest* and the latter as *passively corrupt.* Our arguments in support of passively corrupt players are:

1. *Simulation of broadcast channel:* As highlighted in previous sections, aim of any (valid)BGP/ABG protocol is to simulate a broadcast channel over a point to point (unreliable)network. This should also hold for ABG under the influence of (t_b,t_p)-adversary. Now consider a physical broadcast channel, say $\mathcal{C}$, among a set of n players under the influence of (t_b,t_p)-adversary. Adversary can corrupt upto t_b players actively and another upto t_p players passively. Via $\mathcal{C}$, the General sends his input value $v \in \{0, 1\}$ to all the n players. By property of $\mathcal{C}$, all the n players are guaranteed to receive value v. All honest and passively corrupt players will output v. Adversary can make all the actively corrupt players to output a value of his choice (which may be different from v). It is evident from the above example that for any physical broadcast channel, passively corrupt

[1]In line with literature [PSL80], we assume the authentication scheme to be secure against a computationally unbounded adversary

[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 [LLR02].

players will always decide upon a same as honest players. Thus, any ABG protocol aiming to *truly simulate* a broadcast channel in the presence of (t_b,t_p)-adversary, *has to ensure* that all the non-faulty (honest and passively corrupt) players output *same* value.

2. *Authentication is a means, not the end:* The objective of any (valid)BGP protocol is to simulate a broadcast channel from a designated sender to a set of receivers. In order to facilitate this process, authentication is used as a tool in protocols for BGP. Clearly, authentication is a means and broadcast is the end. In such a scenario even if the tools fails to do its job (in the case of passively corrupt players), why should the objective be altered ? In order to fulfill the original objective, all non-faulty(honest and passively corrupt) players must output same value.

3. *Ideal world/Real World:* A popular paradigm used define the security of any task is the Ideal world/Real World simulation technique [Can01]. We now show that in the ideal world for ABG in the presence of a (t_b,t_p)-adversary, all non-faulty players always decide on same the value. It then follows that the corresponding ABG protocol in the real world has to ensure that all non-faulty players also decide on the same value.

   Informally, consider a set of n players connected to a Trusted Third Party(TTP). (t_b,t_p)-adversary follows its strategy. W.l.o.g let P_i be a passively corrupt player and P_j be a honest player. The General sends a value to TTP. TTP forwards this value to all the n players. All non-faulty players output the value received from TTP. Thus, in the ideal world, P_i and P_j output same value.

4. *Motivation from real life:* In order to authenticate important documents, use of physical signatures is a common practice in day-to-day life. Consider a person who forges signature of some other person(s) for an undue benefit/advantage. It is well known that in such scenarios the law penalizes the person committing the forgery and not the victim(s) of the forgery. Analogously, for ABG under the influence of (t_b,t_p)-adversary, passively corrupt players should not be penalized for the adversary being able to forge messages on their behalf. Thus, all passively corrupt players should be part of agreement like honest players.

Based on the above discussion we now define ABG under the influence of a (t_b,t_p)-adversary. Throughout the rest of this paper we refer to it as 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 the 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. A *passively corrupt* player is one who follows the designated protocol diligently, but the adversary has complete access to his internal state. An *honest* player is one who follows the designated protocol, and over whom the adversary has absolutely no control. For the purpose of this paper, we refer to both *honest* and *passively corrupt* players together as non-faulty. Rest of the paper focuses on the complete characterization of ABG_{mix} over completely connected synchronous graphs.

2.2 Definitions and Notations

We now formally define some of the terms used in this paper:

Definition 2 (Adversary Structure) *An adversary structure $\mathcal{Z}$ for the player set $\mathbb{P}$ is a monotone set of subsets of $\mathbb{P}$, i.e. $\mathcal{Z} \subseteq 2^{\mathbb{P}}$, where all subsets of Z are in $\mathcal{Z}$ if $Z \in \mathcal{Z}$.*

Definition 3 (Adversary Basis) *For an adversary structure $\mathcal{Z}$, $\bar{\mathcal{Z}}$ denotes the basis of the structure, i.e. the set of the maximal sets in $\mathcal{Z}$: $\bar{\mathcal{Z}} = \{Z \in \bar{\mathcal{Z}} : \nexists Z' \in \bar{\mathcal{Z}} : Z \subset Z'\}$*

An important aspect while studying the problem of boradcast and its variants is giving lower bounds for fault tolerance. The main objective of this paper is to give lower bounds for ABG_{mix}. We now formalize certain concepts/terms used in our proofs. 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:

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

The messages sent by player a in any round i of some execution, say Ω, depends on the internal state of a just before sending the message. The internal state of a player consists of all the data the player possesses including the code the player is executing. For the case of ABG, the internal state of a player a consists of 4 parameters: input value with which a starts, secret key used by a for authentication, code being executed by a, and messages sent and received by a up to round $i-1$ of Ω. Since the outgoing messages in any round are a function of incoming messages, we can rewrite the Equation 1 as:

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

To show that the views of 2 different players a, b in 2 different executions(of some ABG_{mix} protocol) 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 same 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:

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

3 Complete Characterization

In this section we give the necessary and sufficient condition(s) for existance of ABG_{mix} protocol tolerating a (t_b,t_p)-adversary over any completely connected synchronous network. We first show that

[3]To account for the fact that players a, b may even run different codes say θ and θ', we require that the message generated for a given player, say C, by θ on input $\mathcal{I}$ should be same as message generated for C by θ' on input $\mathcal{I}$

[4][FLM85] 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}, players having same secret key in both the executions is must.

there does not exists any ABG_{mix} over a completely connected synchronous network $\mathcal{N}$(Figure 1) of three nodes $\mathbb{P} = \{A, B, C\}$ tolerating an adversary basis $\bar{\mathcal{A}} = \{((C),(A)),((A),(\emptyset)),((B,A))\}$. For the rest of this work $((x_1, \ldots, x_i),(y_1, \ldots, y_j))$ represents a single element of adversary basis such that adversary can corrupt $x_1, \ldots, x_i$ actively and simultaneously control $y_1, \ldots, y_j$ passively.

Lemma 1 *There does not exists any protocol solving ABG_{mix} over a completely connected synchronous network $\mathcal{N}$ of three nodes $\mathbb{P} = \{A, B, C\}$ tolerating an adversary basis $\bar{\mathcal{A}} = \{((C),(A)),((A),(\emptyset)),((B,A))\}$.*

Proof: Proof by contradiction. We assume there exists a protocol Π that solves ABG_{mix} over a completely connected synchronous network of three nodes $\mathbb{P} = \{A, B, C\}$ tolerating an adversary basis $\bar{\mathcal{A}} = \{((C),(A)),((A),(\emptyset)),((B,A))\}$. The proof proceeds to demonstrate that there exists an input assignment for which adversary(characterized by $\bar{\mathcal{A}}$) can ensure that non-faulty players in an execution of Π do not always have a consistent output. This contradicts our assumption of Π.

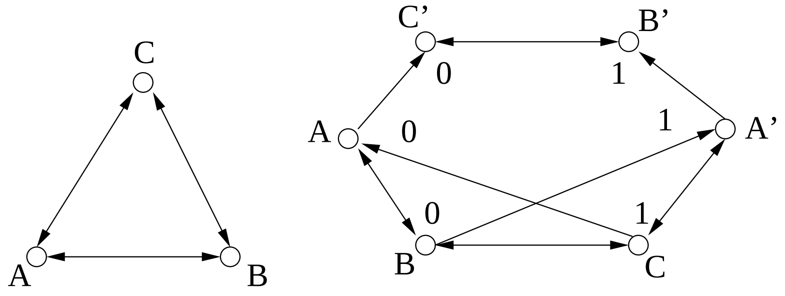

Figure 1: Network $\mathcal{N}$ and System S

To show that $\bar{\mathcal{A}}$ can ensure that non-faulty players do not have a consistent output, we use the technique for giving impossibility proofs, developed by Fischer *et al.* [FLM85]. Using Π we create a protocol π'[Definition 4] in such a way that if Π exists then so does π'(Lemma 2). Using two copies of π' we construct a system S (Figure 1). We then show that S must exhibit a contradictory behavior. This implies impossibility of the assumed protocol Π.

Formally, S is a synchronous system with a well defined output distribution for any particular input assignment. Here we do not know what system S is supposed to do. Therefore, the definition of ABG[Definition 1] does not tell us anything directly about what the players' output should be. All we know is that S is a synchronous system and has a well defined behavior. Further, no player in S knows the complete system. Each player in aware of only his immediate neighbors.

Let α_1, α_2 and α_3 be three distinct scenarios in execution of Π over $\mathcal{N}$. In α_1, A is the General starting with input 0. $\bar{\mathcal{A}}$ corrupts C actively and controls A passively. In α_2, A is the General. $\bar{\mathcal{A}}$ corrupts A and makes him to interact with B as if A started with input 0, and, interact with C as if A started with input 1. In α_3, A is the General starting with input 1. $\bar{\mathcal{A}}$ corrupts B actively and controls A passively. Further, let α be an execution of L where each player starts with input value as shown in Figure 1. All the players in α are honest and follow the designated protocol correctly.

We claim that no matter for how many rounds Π executes, for any round i, whatever *view* (Equation 2) A, B get in α, $\bar{\mathcal{A}}$ can ensure that A, B respectively get same view in α_1 i.e. $view^{\alpha}_{A,i} \sim view^{\alpha_1}_{A,i}$. This implies that the player A cannot ever differentiate between α_1 and α (dubbed $\alpha_1 \overset{A}{\sim} \alpha$). Similarly, player B cannot ever differentiate between α_1 and α ($\alpha_1 \overset{B}{\sim} \alpha$). From the definition of ABG_{mix}

[Definition 1], in α_1, both A, B should decide on value 0. Since $\alpha_1 \overset{A}{\sim} \alpha$ and $\alpha_1 \overset{B}{\sim} \alpha$, both A, B in α will also decide on value 0 (we are able to make claims regarding the outputs of A and B in α as their views are same as those in α_1. Thus by analyzing their outputs in α_1, we can determine there outputs in α). Similarly, $\bar{\mathcal{A}}$ can ensure that $\alpha_3 \overset{A'}{\sim} \alpha$ and $\alpha_3 \overset{C}{\sim} \alpha$. Both A, C in α_3 should decide on value 1. Then so will both A', C in α. Similarly, we claim that $\bar{\mathcal{A}}$ can ensure that $\alpha_2 \overset{B}{\sim} \alpha$ and $\alpha_2 \overset{C}{\sim} \alpha$. As per the definition of ABG_{mix}, B, C in α_2 should agree on same value, then so should B, C in α. But B,C have already decided upon values 0 and 1 respectively in α. This implies S must exhibit contradictory behavior. ■

To complete the above proof, we need to show that $\bar{\mathcal{A}}$ can always ensure that – A, B get same view in α and α_1, B, C get same view in α and α_2 and A, C get same view in α and α_3. We prove the same in Lemma 3, 4, 5 respectively. We now define the protocol π'[Definition 4] and show that if Π exists then so does π'(Lemma 2).

Definition 4 **(π')** *For 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')* [5] *in π'. Similarly any statement of the kind "c sends message m to a" in Π is replaced by "c multicasts message m to all instances of a" in π'. Rest all statements in π' are same as those in Π.*

Lemma 2 *If Π exists then π' exists.*

Proof: Implied from Definition 4. ■

As a prelude to Lemma 3, 4, 5, we introduce the notion of *execution trees.* In essence, we wish to show that view of certain player(s) remains same in two distinct executions of some (valid) ABG_{mix} protocol. For example, in the proof of Lemma 1 we wish to prove that the adversary $\bar{\mathcal{A}}$ can always ensure that A(similarly B) gets same view in α and α_1. From equation 3, it suffices to show that $\bar{\mathcal{A}}$ can always ensure that $A(B)$ gets same messages in α_1 and α. Thus, all we need to show is that whatever messages A receives from B, C in α, $\bar{\mathcal{A}}$ can always ensure that A gets the same messages from B, C in α_1 too. Similarly, whatever messages B receives from A, C in α, $\bar{\mathcal{A}}$ can always ensure that B gets the same in α_1 too. Our technique is as follows – note that what node A receives in round i of α(or α_1) depends on what nodes B and C send to it in round i of α(or α_1). So we need to argue that these messages sent in round i of α and α_1 respectively are same or *can be* made same by adversary. Now the messages B,C send in round i of α and α_1 depend on what they them self receive in previous round $i-1$. This in turn depends on what A, C(or A, B) send to B(or C) in round $i-2$ of α and α_1 respectively. Thus we need to argue that adversary can ensure that whatever messages A, C(or A, B) send to B(or C) in round $i-2$ of α is same as whatever messages A, C(or A, B) send to B(or C) in round $i-2$ of α_1. Note that this continues in a recursive manner until recursion stops at round 1. The entire recursion can be visualized as trees T^A_α and $T^A_{\alpha_1}$ rooted at A for executions α and α_1 respectively as shown in Figure 2.

We now formally describe execution tree T^x_α. 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. *We note the following* – to show that a player, say x, receives same messages in two different executions, say α and α_1, it suffices to

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

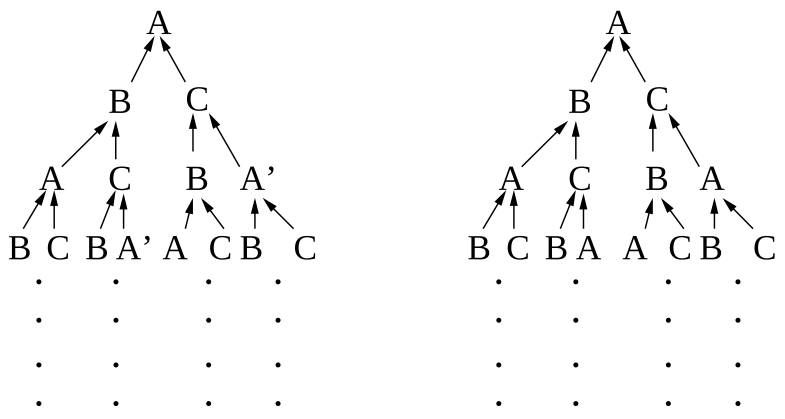

Figure 2: T^A_α and $T^A_{\alpha_1}$

show that execution trees T^x_α and $T^x_{\alpha_1}$ are similar. We prove this similarity using induction on height of T^x_α and $T^x_{\alpha_1}$.

We now show that the adversary $\bar{\mathcal{A}}$ can ensure that A, B get same view in α and α_1. As a prelude, we specify the behavior of the adversary for scenario α_1:

1. *Send outgoing messages of round i:* Based on the messages received during round $i-1$, $\bar{\mathcal{A}}$ decides on the messages to be sent in round i. For round 1, $\bar{\mathcal{A}}$ sends to B what an honest C would have sent to B in execution α_2. For $i \geq 2$, $\bar{\mathcal{A}}$ authenticates $msg^{\alpha_1}_{i-1}(B, C)_B$ using C's key and sends it to A. For $msg^{\alpha_1}_{i-1}(A, C)_A$, $\bar{\mathcal{A}}$ 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. If so, $\bar{\mathcal{A}}$ authenticates and sends same message to B as C would have sent to B in round i of execution α_2. Formally, $\bar{\mathcal{A}}$ constructs $msg^{\alpha_1}_{i-1}(A, C)_A$, ($\bar{\mathcal{A}}$ can construct $msg^{\alpha_1}_{i-1}(A, C)_A$, since it passively controls A and has messages received by A in previous rounds) such that $msg^{\alpha_1}_{i-1}(A, C)_A \sim msg^{\alpha_2}_{i-1}(A, C)_A$, authenticates it using C's key and sends it to B. If the message has been authenticated by B even once, $\bar{\mathcal{A}}$ simply authenticates $msg^{\alpha_1}_{i-1}(A, C)_A$ using C's key and sends it to B.

2. *Receive incoming messages of round i:* $\bar{\mathcal{A}}$ obtains messages $msg^{\alpha_1}_i(A, C)_A$ and $msg^{\alpha_1}_i(B, C)_B$ via C. (These are round i messages sent by A and B respectively to C). Similarly via A, $\bar{\mathcal{A}}$ obtains messages $msg^{\alpha_1}_i(B, A)_B$ and $msg^{\alpha_1}_i(C, A)_C$. (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 up to round $i-1$).

Lemma 3 *$\bar{\mathcal{A}}$ can ensure $view^\alpha_A \sim view^{\alpha_1}_A$ and $view^\alpha_B \sim view^{\alpha_1}_B$*

Proof: From equation 3 it is sufficient to show that for any round i, whatever messages $A(B)$ gets in α, $\bar{\mathcal{A}}$ can ensure that $A(B)$ get same messages in α_1 i.e. $msg^\alpha_i(x, A)_x \sim msg^{\alpha_1}_i(x, A)_x$ and $msg^\alpha_i(x, B)_x \sim msg^{\alpha_1}_i(x, B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$.

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

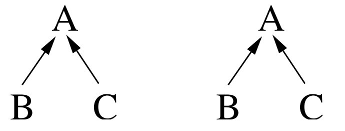

Figure 3: T_α^A and $T_{\alpha_1}^A$ at the end of round 1.

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

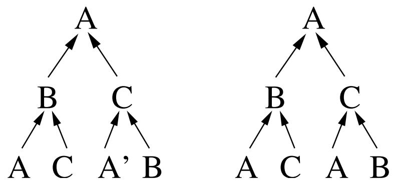

Figure 4: T_α^A and $T_{\alpha_1}^A$ at the end of round 2.

We now claim that the similarity holds in round 2 as well i.e. $msg_2^\alpha(x,A)_x \sim msg_2^{\alpha_1}(x,A)_x$. Consider trees T_α^A and $T_{\alpha_1}^A$ at the end of round 2 as shown in Figure 4. Node A 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 C is Byzantine in α_1, $\mathcal{A}$ can construct message $msg_1^{\alpha_1}(A,C)_A$ such that $msg_1^{\alpha_1}(A,C)_A \sim msg_1^\alpha(A',C)_A$. Thus C can simulate to receive messages in α_1 same as those in α at the end of round 1. Now B receives same messages in α and α_1 and has same input value, secret key and executes same code, thus $msg_2^\alpha(B,A)_B \sim msg_2^{\alpha_1}(B,A)_B$. 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.

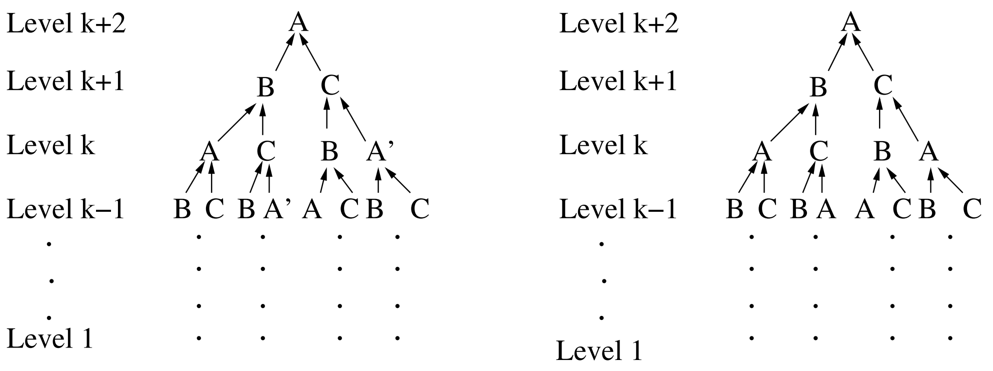

Figure 5: T_α^A and $T_{\alpha_1}^A$ at the end of $k+1$ rounds.

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

Lemma 4 *$\bar{\mathcal{A}}$ can ensure $view_B^{\alpha} \sim view_B^{\alpha_2}$ and $view_C^{\alpha} \sim view_C^{\alpha_2}$*

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

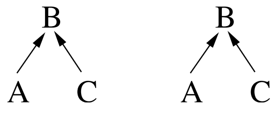

Figure 6: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 1.

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

We now claim that the similarity holds in round 2 as well i.e. $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_2}(x,B)_x$. Consider trees T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 2 as shown in Figure 7. B as well as C start with same input value, secret key and execute same code in both α and α_2 respectively, thus $msg_1^{\alpha}(B,A)_B \sim msg_1^{\alpha_2}(B,A)_B$, $msg_1^{\alpha}(C,A)_C \sim msg_1^{\alpha_2}(C,A)_C$ and $msg_1^{\alpha}(B,C)_B \sim msg_1^{\alpha_2}(B,C)_B$. $\mathcal{A}$ can ensure that

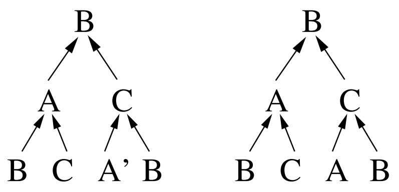

Figure 7: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of round 2.

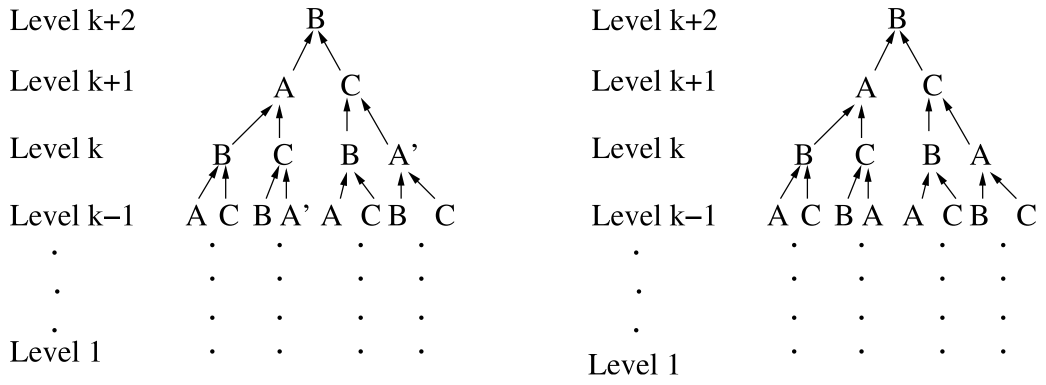

Figure 8: T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of $k+1$ rounds.

$msg_1^{\alpha}(A',C)_{A'} \sim msg_1^{\alpha_2}(A,C)_A$. At the end of round 1, A receives same messages in α and α_2 and has same input value, secret key and executes same code, thus $msg_2^{\alpha}(A,B)_A \sim msg_2^{\alpha_2}(A,B)_A$. Similarly, since C also receives same messages in α and α_2 and has same input value, secret key and executes same code, thus $msg_2^{\alpha}(C,B)_C \sim msg_2^{\alpha_2}(C,B)_C$. Thus, $msg_2^{\alpha}(x,B)_x \sim msg_2^{\alpha_2}(x,B)_x$, $\forall x \in \mathbb{P}$.

Let the similarity be true till some round k i.e. $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$, $\forall i|1 \leq i \leq k$, $\forall x \in \mathbb{P}$. We now show that $\mathcal{A}$ can ensure that the similarity holds for round $k+1$ also. Consider T_{α}^{B} and $T_{\alpha_2}^{B}$ at the end of $k+1$ rounds as shown in Figure 8. For proving the induction step, we need to show that B at level $k+2$ receives same messages in both trees. Consider edges between level k and $k+1$. From induction hypothesis any node B up to level k receives same messages in T_{α}^{B} and $T_{\alpha_2}^{B}$. Since B starts with same input value, secret key and executes same code in both α and α_2 respectively, thus will send same messages in round k i.e. $msg_k^{\alpha}(B,A)_B \sim msg_k^{\alpha_2}(B,A)_B$ and $msg_k^{\alpha}(B,C)_B \sim msg_k^{\alpha_2}(B,C)_B$. Similarly one can argue that $msg_k^{\alpha}(C,A)_C \sim msg_k^{\alpha_2}(C,A)_C$. This is because from the induction hypothesis step on heights of T_{α}^{C} and $T_{\alpha_2}^{C}$, one gets $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i|1 \leq i \leq k$, $\forall x \in \mathbb{P}$. Now consider A' at level k in in T_{α}^{B} and corresponding A at level k in in $T_{\alpha_2}^{B}$. For time being assume A' up to level k in T_{α}^{B} receives same messages as corresponding A in $T_{\alpha_2}^{B}$. Since A is corrupt in α_2, $\mathcal{A}$ can always ensure that in round k of α_2, A sends to C what A' sends to C in round k of α. Thus A at level $k+1$ receives same messages in both T_{α}^{B} and $T_{\alpha_2}^{B}$. Since A starts with same input value, secret key and executes same code in both α and α_2 respectively, one gets $msg_{k+1}^{\alpha}(A,B)_A \sim msg_{k+1}^{\alpha_2}(A,B)_A$. Similarly, one gets $msg_{k+1}^{\alpha}(C,B)_C \sim msg_{k+1}^{\alpha_2}(C,B)_C$. Thus induction holds for round $k+1$ too. The proof is based on a assumption that A' at level k in T_{α}^{B} receives same messages as corresponding A in $T_{\alpha_2}^{B}$. Note that A' in T_{α}^{B} and A in $T_{\alpha_2}^{B}$ receives messages from B and C. Using induction and arguments similar to those given above one can show that such an assumption indeed holds true. Thus $msg_i^{\alpha}(x,B)_x \sim msg_i^{\alpha_2}(x,B)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ holds true. Argument for $msg_i^{\alpha}(x,C)_x \sim msg_i^{\alpha_2}(x,C)_x$, $\forall i > 0$, $\forall x \in \mathbb{P}$ follows similarly. Combining these with Equation 3, we get $view_B^{\alpha} \sim view_B^{\alpha_2}$ and

$view_C^{\alpha} \sim view_C^{\alpha_2}$ ■

Lemma 5 *$\bar{A}$ can ensure $view_C^{\alpha} \sim view_C^{\alpha_3}$ and $view_{A'}^{\alpha} \sim view_A^{\alpha_3}$.*

Proof: Owing to the symmetry of System S, the proof is very similar to the proof of Lemma 3. Details omitted. ■

We now present the main theorem of this work.

Theorem 6 (Main Theorem) *There does not exists any protocol solving ABG_{mix} over a completely connected synchronous network $\mathcal{N}'$ 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. Let us assume that there exists a protocol η solving ABG_{mix} over a completely connected synchronous network $\mathcal{N}'$ of n nodes tolerating (t_b,t_p)-adversary if $n \leq 2t_b + min(t_b, t_p)$, for $t_p > 0$. We show that using η one can construct a protocol Π solving ABG_{mix} over a completely connected synchronous network $\mathcal{N}$ of three nodes $\mathbb{P} = \{A, B, C\}$ tolerating an adversary basis $\bar{\mathcal{A}} = \{((C), (A)), ((A), (\emptyset)), ((B, A))\}$. From Lemma 1 we know that there does not exists any such Π. This contradicts our assumption of η.

We partition the n players into three mutually disjoint nonempty sets I_A, I_B and I_C such that $|I_A| \leq min(t_b, t_p)$, $|I_B| \leq t_b$ and $|I_C| \leq t_b$. Since $n \leq 2t_b + min(t_b, t_p)$, such a partitioning is always possible. The edges in $\mathcal{N}'$ can now be considered as the bundle of edges between the groups I_A, I_B and I_C. Each of the three players A, B and C in Π simulate players in I_A, I_B and I_C respectively. Each player i in Π keeps track of the states of all the players in I_i. Player i assigns its input value to every member of I_i, and simulates the steps of all the players in I_i as well as the messages sent and received between pairs of players in I_i. Messages from players in I_i to players in I_j are simulated by sending same messages from player i to player j. If any player in I_i terminates then so does player i. If any player in I_i decides on value v, then so does player i.

We now prove that if η solves ABG_{mix} over $\mathcal{N}'$ tolerating (t_b,t_p)-adversary when $n \leq 2t_b + min(t_b, t_p)$, then Π also solves ABG_{mix} over $\mathcal{N}$ tolerating adversary basis $\mathbb{A} = \{((C), (A)), ((A), (\emptyset)), ((B, A))\}$. Let Ψ and Ψ' be executions of η and Π respectively. W.l.o.g we let honest, passively corrupt and malicious players in Ψ to exactly simulated by honest, passively corrupt and malicious players respectively in Ψ'. As per our assumption Ψ solves ABG_{mix}, thus satisfies Definition 1. We now show that same holds for Ψ' if it holds for Ψ. W.l.o.g in Ψ, let the general be from set I_i, then in Ψ' player i acts as the general. Note that in Ψ if I_i is controlled actively or passively by the adversary, then so is i is Ψ'.

Let j,k ($j \neq k$) be two non-faulty players in Ψ'. j and k simulate at least one player each in Ψ. Since j and k are non-faulty, so are all players in I_j, I_k. For Ψ, all players in I_j, I_k must terminate, then so should j and k. In Ψ, all non-faulty players including all the players in I_j and I_k should decide on same value, say u, then in Ψ', j, k will also decide on u. In Ψ, if the general is non-faulty and starts with input value v, then in Ψ' too, the general will be non-faulty and starts with input value v. In such a case in ψ, all non-faulty players including all the players in I_j and I_k should have $u = v$. Then in Ψ', j, k will also have $u = v$. Clearly, ψ' also satisfies definition 1. Thus Π solves ABG_{mix} over $\mathcal{N}$ tolerating adversary basis $\mathbb{A} = \{((C), (A)), ((A), (\emptyset)), ((B, A))\}$. ■

Theorem 7 *It is possible to design protocol solving ABG_{mix} iff $n > 2t_b + min(t_b, t_p)$, for $t_p > 0$.*

Proof: Necessity follows from Theorem 6. For sufficiency, we present protocol solving ABG_{mix} for $n > 2t_b$+min(t_b, t_p), $t_p > 0$. We present the protocols separately for $t_b > t_p$ and $t_b \leq t_p$.

`Case of` $t_b > t_p$: $n > 2t_b + \min(t_b, t_p)$ reduces to $n > 2t_b + t_p$. For this we present a protocol and prove its correctness in section 3.1.

`Case of` $t_b \leq t_p$: $n > 2t_b + \min(t_b, t_p)$ reduces to $n > 3t_b$. Here any known BGP protocol works (one such simple protocol is *EIGByz* protocol [Lyn96, page 120]). This is because for unauthenticated setting $t_p = n - t_b$. This completes the sufficiency proof. ■

Remark: As evident from the results of Pease *et al.* [PSL80], for t_p=0, ABG_{mix} is possible iff $n > t_b$. It is interesting to note that for $t_p = 1$, one gets $n > 2t_b + 1$. Thus, with respect to fault tolerance of ABG_{mix}, $t_p = 0$ or $t_p \neq 0$ makes a huge difference.

3.1 Protocol for $n > 2t_b + t_p$

The proposed protocol is obtained by a sequence of transformations on EIG tree [BNDDS87]. A detailed discussion on the construction of EIG tree is available in [BNDDS87] [Lyn96, page 108]. Our protocol *EIGPrune* is given in Figure 9. Despite our protocol being exponential in number of messages, we present the same for its ease of understanding.

***EIGPrune* Algorithm**

General $\mathcal{G}$ send his value to every player. Every player assumes this value from the $\mathcal{G}$ as his input value and and exchanges messages with others as per *EIGStop* protocol [Lyn96, page 103] for $t_b + t_p + 1$ rounds.

At the end of t_b+t_p+1 rounds of *EIGStop* protocol, player p_i invokes **Prune**(EIG) [Definition 5]. Player p_i applies the following decision rule – take majority of the values at the first level (i.e. all the nodes with labels l such that $l \in \mathbb{P}$) of its *EIG* tree. If a majority exists player, p_i decides on that value; else, p_i decides on *default value*, v_0.

Figure 9: *EIGPrune* algorithm

Definition 5 (Prune(EIG)) *This method that takes an EIG tree as an input and deletes subtrees say* $subtree_j{}^i$ *(*$subtree_j{}^i$ *refers to a subtree in i's EIG tree such that the subtree is rooted at node whose's label is j) of i's EIG tree as given in the sequel. For each subtree* $subtree_j{}^i$*, where label* $j \in \mathbb{P}$*, a set* W_j *is constructed which contains all distinct values that ever appears in* $subtree_j{}^i$*. If* $|W_j| > 1$*,* $subtree_j{}^i$ *is deleted and modified EIG tree is returned.*

We prove the correctness of *EIGPrune* via Lemma 8 – 11.

Lemma 8 *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 9 *After t_b+t_p+1 rounds, if a $subtree_j{}^i$ has more than one value then $\forall$ k, $subtree_j{}^k$ also has more than one value, there by ensuring that all $\forall$ k, $subtree_j{}^k$ are deleted (i, j, k are not necessarily distinct), where i, k are non-faulty.*

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

Lemma 10 *$subtree_j{}^i$ 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 9 as, if subtrees had different values; then as per the protocol they would have broadcasted the values in their *EIG* tree in the next round and thus the subtrees would have more than one different value resulting in their deletion during **Prune**(*EIG*) step. ∎

Lemma 11 *For $n > 2t_b + t_p$,* EIGPrune *algorithm solves ABG_{mix}.*

Proof: $n-(t_b+t_p)$ represents the number of honest players and according to $n > 2t_b+t_p$, $n-(t_b+t_p) > t_b$. Thus honest majority is guaranteed which vacuously implies non-faulty majority. The decision rule ensures that in case the General is non-faulty and starts with v, all non-faulty players decide on v. Further if the General is faulty, all non-faulty should agree on same value. Let i and j be any two non-faulty players. Since, decisions only occur at the end, and by previous lemma we see that $\forall i, subtree_j{}^i$ can have only one value which consistent throughout all $subtree_j^i, \forall i \in \mathbb{P}$. This implies they have the same set of values. The decision rule then simply implies that i and j make the same decision. ∎

4 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

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

[BNDDS87] 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.

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

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

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

[Can01] Ran Canetti. A unified framework for analyzing security of protocols. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(16), 2001.

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

[Dol81] Danny Dolev. The Byzantine Generals Strike Again. Technical report, Stanford University, Stanford, CA, USA, 1981.

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

[FLM85] 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.

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

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

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

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

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

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

[GP92] 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.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 197–206, New York, NY, USA, 2008. ACM.

[HKN$^+$05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, pages 96–113, 2005.

[KK09] Jonathan Katz and Chiu-Yuen Koo. On Expected Constant-round Protocols for Byzantine Agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, 2009.

[LLR02] 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.

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

[Lyn96] Nancy A. Lynch. Distributed algorithms. *Distributed Computing*, 1996.

[MP06] Remo Meier and Bartosz Przydatek. On robust combiners for private information retrieval and other primitives. In *CRYPTO*, pages 555–569, 2006.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In *TCC*, pages 404–418, 2007.

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

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

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21:120–126, February 1978.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Sha94] Adi Shamir. Efficient signature schemes based on birational permutations. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 1–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

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

[SW04] 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.