Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 13, Issue 1, January 2022: 903-916

# Game Theory based Consensus Learning in Unidirectional Rings

**Radha Rani a, Mayank Kumar Singh b, and Dharmendra PrasadMahato c**

[a]Ph.D. Research scholar, Department of Computer Science and Engineering, National Institute of Technology, Hamirpur, India 177005, radharani8288@gmail.com

[b]M.Tech. Research scholar, Department of Computer Science and Engineering, National Institute of Technology, Hamirpur, India 177005

[c]Assistant Professor, Department of Computer Science and Engineering, National Institute of Technology, Hamirpur, India 177005

Article History: Do not touch during review process(xxxx)

**Abstract:**This paper discusses the game theory based consensus learning in unidirectional rings. Distributed systemfindstheambiguousprocessorsorrationalagentsinthesamewayasgametheoryfindsambiguous players. The rational agents are those processors which deviate from the protocol and instead they follow their own path in order to fulfil their utility. In a bidirectional rings, the rational agents can be detected easily by each neighbourhood agents. But, in unidirectional case, the rational agents cannot be easily detected. This limits the honest agent to detect incorrect IDs sent by adversarial agents. So, our main focus is on the asynchronous unidirectional ring. We study the shortcomings of the protocol presentedin(Abraham*etal.*2013)and(Afek*etal.*2014).Inthispaper,wepresentanimprovedversion oftheprotocolbasedonGameTheorywhichismoreresilienttothepreviousone,byapplyinginvasion of adversarial processors and by proving it to be resilient in presence ofadversaries.

**Keywords:** Consensus problem; Leader Election; Game Theory; Unidirectional Rings.

## 1. Introduction

Obtaining consensus in distributed computing on some data value is a fundamental problem to achieve overall reliability. In distributed system, the all components cannot be fault free. Therefore, all the components should agree on some data or state to guarantee the reliability of the system. Consensus can be understood as an agreement between the components of the distributed system on a data value or component. When the consensus is achieved on a component, then this mechanism is known as consensus mechanism based on leader election.

Consensus in distributed system is to elect a leader in an anonymous networks. Message passing in this network does not have a fixed time to send or receive the messages. There is no guarantee that the message will be received on time or reach to its destination. Therefore, some components may become rational components in the system and they start their own paths.

In this paper, we incorporate game theory approach to maximize its Expected Utility. In this case, the utility will be to choose itself as a leader or to form a coalition of agents to elect a leader among

themselves. The implementation of this algorithm is done on unidirectional ring networks.

Therestofthispaperisorganizedasfollows. SectionIIbrieflydescribesthebackgroundinformation. Section III presents the fundamentals of game theory and game models that are commonly used in blockchain. Section IV discusses applications of game theory for security issues in blockchain. Section Vpresentsapplicationsofgametheoryfortheminingmanagementinblockchain.SectionVIdiscusses applications of game theory atop blockchain platforms. Section VII outlines challenges and future research directions. Section VIII summarizes and concludes thepaper.

## 2.BackgroundInformation

### 2.1 Consensus

Designing and Implementation of a fault tolerant distributed system is considered as one of the complex endeavours in the last few years. The verification of these models is the next challenge that we face after implementation. Nowadays, several new techniques have been developed to identify and simplifythesetasks.Therearetwotechniqueswhicharewidelyusedforthispurpose.

TheyareAtomic Broadcasting and Consensus. In distributed system, simple definition of consensus is agreement. It is a task to get all the processors in a network to agree on some specific protocol or condition based on the votes of each processors. It allows each processor to reach to a common conclusion and result, where each processor may or may not have similar inputs. Many problems that arise in distributed system can be solved through such as leaderelection.

### 2.2 Unidirectional RingNetwork

Inaunidirectionalringnetwork as shown in Fig.1,thedataflowsinonlyonedirectioneitherclockwiseoranti-clockwise and such a network is called a half-duplex network. In a bidirectional ring network, each agent can communicate with its predecessor and successor as well. This allows each agent to detect faulty agents in its neighborhood by continuously matching the information sent by its neighbor agents in previous rounds. However, this is not the case in unidirectional ring networks. The message is passed in only single direction and that direction is decided beforehand. This limits the honest agent to detect incorrect IDssentbyadversarialagents.So,ourmainfocusisonasynchronousunidirectionalringnetwork.

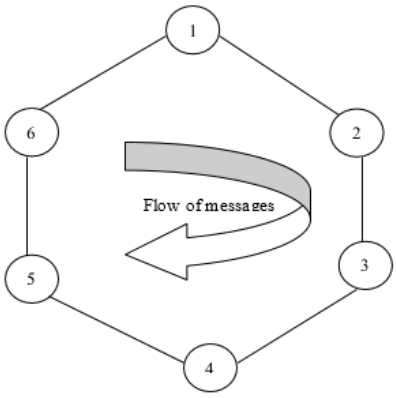


**Fig. 1.** Unidirectional Ring Network

### 2.3 Game Theory in DistributedSystem

Game theory approach has been a better option for solving the complicated problems in distributed system. In game theory concept, there are no good players or bad players, but rational players make moves in selfish manner by assigning payoffs (or utilities) to outcomes. With this type of play with rational players, game theory-predictions consists of several solution concepts considering theoutcome of a game with rational players. In selfish type of play in game theory, the players make bestresponsesto their beliefs. But, in the team, all the players are unknown what the others players' strategies are.

Letusassumeamulti-partygamemodelwheretherare$n$numberofparticipantsandnoneofthem arehonest.Theseparticipantsorplayersarereferredaseitherrationaloradversarial.Therationalplayers arethosewhoactintheirselfishinteresttomaximizetheirutility.Adversarialplayersarethosewhoact arbitrarily (Lysyanskaya *et al.*2006).

### 2.4 Assumptions

In our protocol, we assume that each processor knows about their IDs as well as the IDs of the processors present in the network. This assumption is necessary because getting to know the information about each and every processor during the execution of the protocol is trivial in electing a leader. Hence, we created a wake-up phase which runs before the actual protocol and facilitates each processor with the information of other processors. Once each processor knows about every other processor, our main protocol starts. Although, we should not deny the fact that knowing the IDs of other processors may have many other advantages over not knowing about it atall.

Thefirstadvantageisthateachprocessorgetstoknowaboutthetotalnumberofprocessorspresent inthering.Thisinformationshouldnecessarilybeavailableorelsetheyprotocolwouldneverstop.The processors may not know when to stop sending messages to their neighbouring processors if thereisnotstoppingconditionsuchaswhenthelastprocessorreceivesthefinalmessage,nonew messagesshouldbesentuntiltheinitiatorstartsendingagain.Thesecondadvantageofthisisthat atthetimeofsendingtheirIDstoitsneighbours,processorswillreceiveIDsfromotherprocessors aswell.Hence,theyusethisinformationtoverifythattheIDssentbyotherprocessorsisthesame id that they sent during wake-upphase.

We assume that the agents are rational. If we assume that each processor prefers to have a leader rather than having no leader at all and that they are indifferent of who becomes the leader then any leader election algorithm can be applied. But if there exist a single rational agent among the honestagentswhodeviatesfromtheprotocoltogetthemaximumbenefit,thentheseprotocolsfail to perform well. If we view the scenario as a formal intuition where all the processors arehonest, thenweimplicitlyassumethatinstandarddistributedsettingsallprocessorsfollowthedesignated algorithm.

But what happens when agents prefer to have different leaders? One of the cases maybe when someagentwantshimselftobeelectedasaleaderasitwouldlowerthecostofroutingwithother agents. Here, the standard settings of distributed protocol (which assumes that each agent has distinct id, and the leader is elected based on thehighest or the lowest id present in the network) fails to work. Thus, our major focus being on obtaining a fair Nash Equilibrium, where each processor has an equal

probability of becoming a leader. In simple words, the probability that someonewillbeelectedasaleadershouldbe1.Butbeforedigginginintothesolutions,weneed to understand what Nash equilibrium means in an asynchronoussetting.

### 2.5 Advantages

Indistributedsystem,leaderelectionisanideaofgivingsomespecialpowerstooneoftheprocessors in a network which include assigning work, scaling bottlenecks, handling responsibilities of requests in a system. It is a powerful tool for reducing operations, reducing coordination, improving efficiencyand throughput, simplifying architecture, etc. Leader election provides a centralized way of management where a single processor makes decisions suitable for the rest of the processors and the entire system itself. Advantages of leader election are asfollows:

- A leader provides consistency to the clients as it has the power to control the changes that can be made to the state of thesystem.
- Designing a software for one specific processor is much easier than designing different protocols for each individual processor. The leader does not care about the conditions of system whether working or not at the time of decisionmaking.
- One of the major advantages of leader election is to reduce partial failures and keeping logs and metrics for failure and recovery. If there is one centralized system that takes all the decisions, it makes the system easier for humans to thinkabout.
- Efficiency of a centralized system is better since it can simply circulate the changes need to be made rather than building consensus about the changes need to bemade.
- Asingleleadercanimproveperformanceorreducecostbyprovidingasingleconsistentcacheof data which can be used everytime.

### 2.6 Disadvantages

Disadvantages of leader election are as follows:

- Paradigms like partial deployment, incremental deployment, A-B testing, automatic rollback is hard to apply in leader electionsystem.
- Adversarial attacks of processors may result in disobeying leader’s decision and following their own protocols to maximize their optimal expectedutility.
- Asingleleaderispronetosinglepointfailure.Theentiresystemwillsufferfromunavailabilityif the leader itself is failed or fix a faultyprocessor.
- Iftheleaderitselfisnotfollowingthedesigner’sprotocolanddoingwrongworkandifthereisno othernodetocheckforit,thenitcanmakedecisionsfortheprocessorswhichwillbeinfavorof the leader and the entire system will have a high blastradius.

### 2.7 Motivation

In this paper, our motive is to solve the consensus problem based on game theory approach. Here, leader election is used because of:

- **Data Consistency:** In a bank server system, multiple servers exist in a cloud, but only a single server should be responsible for a singlecustomer.
- **Data Replication:** If a group of servers decide to replicate files among themselves, they need to electaoneserverthatcontainstheprimarycopyofthefilethatwillcommunicatewithotherclient machines.
- **Error-Free Transition:** In comparison to government elections, sometimes leadercommunicate withclientsthathavedifferentkindsofinformationwhichmayormaynotbeunderstoodbyother servernodes.So,leadertransformstheinformationsothatitisunderstoodbyallthenodes.

This paper does not focus on how leader election can be applied using game theory approach, butit focuses on how leader election in a unidirectional ring can be made fault tolerant so that it is efficient andthatnoothernodedeviatesfromtheprotocol.Thepaperproposessometheoremsandproofstouse game theory as a tool to define how leader is elected and that why other nodes get to the conclusion of electing that particular node as a leader. Also, it states why no other node (not leader) deviates from its decision of choosing a leader rather than beingone.

### 3. Relatedwork

Thereisonestudywhereitisproventhatarandomprotocolcanachieveresiliencetocoalitionoflinear size. This means that if the coalitions are linearly dependent to *n* then the protocol is resilient. This wasfoundbyMichealSakswhichisgivenin*(SaksandMichael1989).*Inthelateryears,Boppana optimal resilience. i.e., resilience to size $(\frac{1}{2} - \epsilon)$. It is important to note that our approach can be and Narayanan *(Boppana et al. 2000)* provided a brief proof of this protocol that has approximate wellsuitedtodefinetheunderlyingprincipleof*(Abrahametal.2013)*whichshowsthatitiseasierto design a leader election protocol when working under synchronous or completely connected network. Forasynchronoussettings,theseprotocolsfailtoperformwellevenifthereisasingleprocessorfailure since we cannot identify whether the processor is really at fault or just responding really slowly. They alsostatedthatinbidirectionalringnetwork,eachprocessorcansendmessagestoitspredecessoraswell as its successor since the ring is bidirectional. This means that if there is a single fault in the network, the processors can confirm it by sending the message in the opposite direction from where the fault has initiallyoccurred.Theprotocolthatworksforunidirectionalnetworkinsynchronoussettingswillwork sameasinbidirectionalnetworkinsynchronoussettings.Theonlythingneedstochangeisthedirection of the message transfer at the time of initiation. If there is no agreement among the processors on the orientationofmessagepassingthentheinitiatorprocessorcanchoosethedirectionarbitrarilyandrestof theprotocolwillbeexecutedasdesigned;alltheotherprocessorswillcontinuetoforwardtheirmessages in the same direction. If any of the processor registers its buffer with ‘null’, then the protocol will fail and no leader will be chosen. Since this approach reduces the adversarial attacks on the protocol, itwill not be fair in terms of getting an elected leader. If the processor keeps on deviating from the

protocol, no leader will be chosen at all. That is why, we assume in our model that each processor follows a solution-drivenapproach.

### 3.1 Secret sharing and MultipartyComputation

Secret Sharing method: A method to divide a secret into parts, called shares that are distributed among a group of agents such that the secret can only be reconstructed when an authorized group of agents combine their shares. Combinations of shares of unauthorized agents do not reveal the secret. Multipartycomputationmethod:Insecuremulti-partycomputation(MPC),asetofparties,eachhaving asecretvalue,wanttocomputeacommonfunctionovertheirinputs,withoutrevealinganyinformation abouttheirinputsotherthanwhatisrevealedbytheoutputofthefunction.In(*Abrahametal.2006*)they mathematicallyprovedthatifthereisacoalitionCofsizekandallthemembersofthecoalitiondeviates fromtheprotocolthennomembercandobetterinelectingaleaderamongthem.Fewotherthingsother thanthiswasthattheadversarialagentscanattackandgetaleaderelectediftheypreferasolutiondriven approach, which is exactly what we assumed in our model. In solution driven approach the processors tend to elect a fair leader rather than having no leader at all. Finally, Abraham *et al.* in (*Abraham et al. 2006*) discussed the mediator technique and simulated games with mediator by games without mediator.Theirmaincontributionliesontheimprovementoftheresultsby(*Andrychowiczetal.2014*) wheretheyusedrationalitytostickoutfromthetraditionalmethodof‘good-guy’‘bad-guy’modelwhich isastandarddistributedcomputingcommunitymodel. *(Andrychowiczetal.2014)*providedamodelthat hadaresultwith1-resilientequilibrium.*(Abrahametal.2013)*improvedthisresultto*(n-1)*resilience. Thismeansthatifanymultipartycomputationiscarriedoutbyatrustedmediator,thenitcanalsobe done without a mediator with a high resilience output. Their result gives a deeper understanding of cheap talk where a model can be built to securely simulate resilient equilibrium without the use of a trusted third party mediator. The motivation to do so is because there is a possibility that we may not find a trusted third party all the time. In *(Izmalkov et al. 2011)* and *(Lepinski et al. 2004)* theoretically described two ways to actually get an equilibrium in a game which is converted from using trusted mediatortonotrustedmediatoratall.Theyreliedontwoverystrongfundamentalsknownasballotbox andenvelope.Thesefundamentalsareundevelopedtobeimplementedinamodelyet.*(Vidaetal.2013)* proposedamodeltosimulateNashequilibriumwiththehelpofatrustedmediator.Heusedthestrategy called ‘punishing the player’ which threatens the players who deviate from the protocol by notelecting anyprocessorasaleaderiftheycometoknowsomeoneisnotfollowingtheprotocol.Later*(Hellerand Yuval2010)*extendedBenPorth’sresulttomodelthisapproachinacoalition.Theyshowthatifstandard assumptionsaremadeincryptography,anequilibriumofagamecanbesimulatedwithamediatorsuch that players use punishmentstrategy.

### 3.2 BARModel

BAR Model (Byzantine, Altruistic, Rational): Byzantine nodes may deviate arbitrarily from the suggestedprotocolforanyreason.Theymaybebroken(e.g.,misconfigured,compromised,malfunctioning, ormis-programmed)ormayjustbeoptimizingforanunknownutilityfunctionthatdiffersfromtheutility functionusedbyrationalnodes—forinstance,ascribingvaluetoharminflictedonthesystemoritsusers.

Altruisticnodesfollowtheexactprotocol. Theyrepresent“seednodes”inrealsystem.Rationalnodes areself-interestedandseektomaximizetheirbenefitaccordingtoaknownutilityfunction.Theydeviate fromtheirsuggestedprotocolonlyifindoingso,givesthembetterutilityinparticipatinginthesystem. *(Aiyeretal.2005)*discussedBARmodelandprotocolsthatcantolerateByzantinebehaviourswhenthe protocolisnotfollowed,misconfiguredorbroken. Rationalbehaviourofanagentwhenselfishprocessor deviates from the designer’s specification to increase their expectedutility outcome. Amidst this, they reduced the complexity of a general three-level architecture model under BAR to get optimal building services. They described the first ever backup service to condone Byzantine processors and the number of unbounded rationalusers.

*(Chandra et al. 1996)* introduced a fault detection algorithm which is based on the study of an actual RAIN system. RAIN stands for Reliable Array of Independent Nodes which is implemented to recovering faults using distributed checkpoint. Other than this they designed a fault tolerantdatabase and distributed web services. For a system to be fault tolerant, the system must monitor each node independentlyandkeeptrackonfailuresandneighbouringprocessorsmusttakeactionsiftheirsuccessor orpredecessorareatfault.Theyspecificallymakeuseoftwoapproachestohelpdetectfailureandtake necessary actions forrecovery.

The first approach is group membership that provides coalition of processors which work in unison to detect their neighbour’s status *(Lysyanskaya et al. 2006)*. The second approach is that this paper is based on i.e., leader election. Leader election protocol brings an inconsistency in global state. Tasks like consensus, load balancing, resource allocation, etc can be solved once a leader has been elected (*Abraham et al. 2013*). But even these approaches have some problems related to them.Both of them require some kind of agreement during the presence of failure. One should keep in mind the impossible result that (*Fang and Chunsheng 2011*) which showed that it is impossible to get consensus in an asynchronous network if there exist a single silent failure. (*Chandra et al. 1996*) discussed that we can remove weakly specified group members that are prone to system failure sooner than other healthyagents.

### 3.3 SimilarAlgorithms

(*Bakshi et al. 2011*) presented a leader election algorithm in asynchronous bidirectional ring based on probabilistic approach. The final election of leader is done through random identity selection, to detect identity clashes they used hop counters and round modulo 2. The validation and verification are donebybuildingthemodel,final-state,sothattheprobabilityofelectedleadershouldbe1.Theirwork isanextendedstudyoftheleaderelectionalgorithmin(*Changetal.1979*).Theyrequiredtohaveunique identities for each of the processor and that it should be ordered. Their algorithm elects a leader with the largest identity available in the network. This may give rise to adversarial attack of the processors. Processors can deliberately choose a highest number and confirms it to be itsID.

(*Bauk and Sung Uoon 2005*) designed a leader election algorithm for unidirectional ring where the size of the ring was known and it was similar to the leader election model of (*Abraham et al. 2006*). (*Fischer et al. 2006*) presented a self-stabilizing leader election algorithm for unidirectional ring where the size of the ring was unknown which, at the end, returns the same leader to each agent.

Aftercarefulconsiderationofthisproblem,weconcludethatitishelpfultouseamediatorinelecting

aleaderinnetwork.Buttakingintoaccounts,athird-partymediatorwillincreasethechancesofdata leak and reduced security. Basically, giving authority to an agent, who is not participating in the leader election, to elect leader among various agents is risky. Hence, there has been studies on whether a problem that can be solved with a third-party mediator can be converted to be solved using cheap-talk, withouttheconcernofamediator.Themethodofusingamediatorhasgainedcurrencyintherecentyears sinceitcanbeusedtosolvemanydistributedcomputingproblems.Forexample,byzantineagreementis achievableusingamediatorevenifhalfoftheagentsarecorrupted.Eachagentcantellthemediatortheir ID's and their preference to who should be the leader, and the mediator chooses the majority. We focus on this type of problem, i.e., to study the result of the protocol in (*Abraham et al. 2013*), and formulate an enhanced version ofit.

## 4. ProblemStatement

Inthispaper,wefocusontheproblemofconsensusbyleaderelectioninasynchronoussettings.Wesay thatadistributedsystemisasynchronousifthereisnotimelimittoexecuteasteplikemessagedelivery, clockdrift,etc.Ithassimplesyntax;theportabilityoftheapplicationsbasedonthismodeliseasierthan assumingtimeconstraints;andpractallywecansaythatunexpectedloadoveraspecificprocessorisa specialityofsysteminasynchrony.Theprotocolbreaksifevenasinglenodefailstofollowtheprotocol in unidirectional or bidirectional ring. However, there are some chances of recovery when protocol is appliedinacompletelyconnectednetwork.Thefaultyagentscanbereducedbyapplyingthealgorithm designed in the failed areas which can detect failures consistently in distributed system. The processors should be able to differentiate between a fault-free population and faultyone.

## 5. Acceptance-Based-a-Lead $^{uni}$ A Robust FileProtocol

In this section, we will be discussing about the enhanced approach of leader election which is an extendedversion of A-Lead$^{uni}$by (*Abraham et al. 2013*). This protocol is expected to be resilient to $O(\sqrt{n})$ Inthissection,wewillbediscussingabouttheenhancedapproachofleaderelectionwhichisanextended dishonest processors. We extend the A-Lead$^{uni}$protocol and add an "acceptance" stage which keepsall the processors in synchrony. We apply "dashing of data" technique where the adversarial agents waitto receive all the messages from other agents before sending its own message. In this way, the adversarial processorsminimizethenumberofmessagesbeingtraversedalongthering.Themajordrawbackthatwe observed in A-Lead$^{uni}$is that during the execution of the protocol, all the processors shouldnecessarily be in synchrony or else the honest processors can detect the fault and abort the protocol. We know that in A-Lead$^{uni}$, theprocessors keep sending the secret messages till all the processors has received the IDs of every other processor in the ring. The processors do this by keeping the number of times they havesentandreceivedIDsfromtheirrespectivepredecessorprocessor.Thisissimulatedusing"circular trip" starting and ending from and to the originator. In each circular trip, every processor receives a secret message from its predecessor and transfer the secret message (also known as the ID of previous processor) that it received in the precedingtrip.

To understanditsimply, letusassume$Transmit_i$,tobethetotalnumberofsecretmessages there are no adversaries in the ring then we have, | $Transmit_i$ - $Transmit_j \leq 1$ |. The equation already sent by a processor $i$ (be it honest or adversarial processor). If we consider a case where confirms that all the processors are in synchrony. Since, we are focused on solving the leaderelection problem in asynchronous ring network, our attack, invasion of adversarial processor, makes use of the disadvantage of asynchronous settings in distributed system and de-synchronize non-adversarial processors. The crucial remark that can be made here that enabled the attack to occur is the difference between | $Transmit_i$ - $Transmit_j$ | is considerably substantial as compared to longest genuine portion, $H_1$. This is the major reason how an adversarial processor $A_i$comes to know about all the secret messagesbeforechoosingitsownmessage. The"acceptance"stageworksinthefollowingmanner:

Every processor needs to choose a uniformly distributed "acceptance value", $a_i \epsilon X$ where $X = n^2$. Thedecisionforhowmanynumberofcirculartripsarerequiredwillbebasedonthetotalnumberof processors participate for the leader election present in the ring.

Each processor will be given the opportunity to validate the acceptance value of other processors. Only one processor will be allowed to become the acceptor in a particular circular trip. That is, in the$i^{th}$trip,$i^{th}$processorwillbecometheacceptorandsendsitsacceptancevalue,$a_i$.Alltherestof the processors will receive and send this acceptance value immediately to their neighbouring processor with no delay in the ring.

No other processor can save this acceptance value or send its own value in the ring unless its his turn to do so. When all the processors have transferred the value, the value will be checked by the trip's validator that whether it is the same acceptance value that he sent and committed to in the beginning. The acceptance value is not same as the secret message that every processors send. We distinguished both the values for different purposes. The secret message is used tocalculatethefinalresult,thatis,whichprocessorshouldbeelectedasaleaderandacceptancevalue isusedtovalidatewhetherornotisthereanyprocessortryingtomanipulatetheincomingmessages. Inotherwords,weuseacceptancevaluetokeepalltheprocessorsinsynchrony. Thesecretmessages are the random values that each processor, $i$ selects to send. The final result will be calculate $\sum_{i=1}^{n} m_i (mod\ n)$.

The vital distinguished design behind the enhanced protocol is to force all the processors to be more synchronized than before. Since, there are some drawbacks in synchronization settings like small informationcanmovefaster,sothemethodusedtoprovepreviousresiliencefails.Therefore,inorderto copeupwiththisproblem,thenewprotocolmakesuseofrandomfunctionthatgeneratesrandomvalues for each processor. Hence, adversarial processors receive lots of information before getting biased to calculate the output. Therefore, we prove that there is a need to send an enormous amount of messages by an adversarial processor before knowing about the secret messages of otherprocessors.

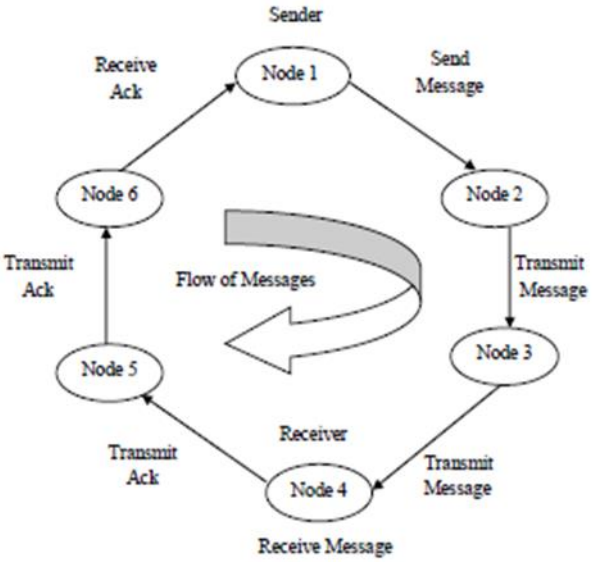


**Fig. 2.** Secret Sharing Protocol

If we look back on the A-Lead$^{uni}$protocol, it was based on Shamir's secret sharing sub-protocol. Our protocol is based on the similar paradigm which deals with sharing secret messages along the ring and wait to receive an acknowledgement from the receiver.

**Algorithm 1** Acceptance-based-A-LEAD$^{uni}$

*Phase 1 – For originator processor 1*

1: Initially select an acceptance value $a_{1,}$and a secret message $m_i$, $t = m_i$, where $t$ is the trip number.

2: Send the secret message in the ring and wait for the secret message $m_{1,nt+1}(modn)$

3: **if** it is the first trip, i.e. $t = 1$,

**then** Send its acceptance value in the ring and wait to receive it at the end of the trip to validate the synchrony of processors.

4: **if**itisnotthefirsttrip

**then** forwardimmediatelytheincomingacceptancevaluetothesuccessive processor. Finally, forward the remaining secret messages $m_{1,nt+1}(modn)$

*Phase 2 – For standard processor i*

5: Choose a secret message $m_i$, $i = m_i$and an acceptance value $a_{i,t}=a_i$

6: Since it is a standard processor, it will receive the secret message $m_{1,nt+1}(modn)$ and then it will send the message stored in the buffer$m_{1,nt+1}(modn)$

7: **if** The value of trip number, $t$ is equal to $i$ i.e., the $i^{th}$processor

**then** send theacceptance value $a_{i,t}$and wait to receive it at the end of the trip to validate the synchrony of processors.

8: **if** They are not equal

---

**then** forward immediately the incoming acceptance value to the successiveprocessor.

At first, the originator processor 1 chooses a random secret message m1 and an acceptance value a1. Using Shamir's secret sharing protocol send the secret message and acceptance value $\{m1\}$ and $\{a_i\}$ to every processor in the ring. Foraprocessor$i$itreceives$m_{i,1},\ldots,m_{i,n},a_{i,1},\ldots,a_{i,n}$.Thesevaluesare value with the one that it sent at the beginning of the trip. If $a_{i,j} \neq a_i$the processor aborts the protocol receivedalternativelyandnotduringthesametrip. Eachprocessorthenverifiestheincomingacceptance by giving output $=NULL$.

Inourmodel,weassumethattheprocessorsareconsecutivelyplacedinanincreasingorderoftheir IDs. However, this arrangement can be changed according to the need. Our protocol works even when theprocessorsarelocatedrandomlyinthering.WeprovethatSync-A-Lead$^{uni}$isareliableprotocol byshowingthatadversariescannottransferlargeamountofinformationamongthemselveswhichisthe crucial step for them to know the secret messages of every processor and bias the outcome. We apply a random function R instead of the summation function to calculate the final result/output. To stop the adversaries from misusing the acceptance stage for their own benefit, we apply the random function to both secret message and acceptance value stage so that even if the processors calculate the partial sum of the secret messages during the execution of the protocol, they will not be able to bias the result of random function at the end that determines a fair leader for the asynchronousring.

## 6. Results

GameTheoryandDistributedSystemproblemsgohandinhand. Bothstriveindeterminingthebest possible outcome for a player/processor in a game/network.We studied the drawbacks of A-Lead$^{uni}$ protocol and proposed an enhanced version of it resilient to $O\sqrt{n}$adversaries and design a new protocol after some modifications. Our protocol performedwell even in the presence of invasion of adversarial processors. Although there were few assumptions in out model, they helped us in overcoming the drawbacks of A-Lead$^{uni}$ and design a new protocol after some modifications.

We can say that our protocol used game theoretic approach as a tool to define how leaders are elected and also how each processor comes down to electing only a single leader among various candidates. Lastly, we can say that with the help of our model, we managed to keep all the processors in synchrony and that they follow a solution based approach where they will choose to have any leader rather than having no leader at all.

## 7. Conclusion and FutureScope

Our protocol performed well even in the presence of invasion of adversarial processors. Althoughthere were few assumptions in our model, they helped us in overcoming the drawbacks of A-Lead$^{uni}$and design a new protocol after some modifications. We can say that our protocol used game theoretic

approachasatooltodefinehowleadersareelectedandalsohoweachprocessorcomesdowntoelecting only a single leader among various candidates. Lastly, we can say that with the help of our model, we managed to keep all the processors in synchrony and that they follow a solution based approach where they will choose to have any leader rather than having no leader atall.

Therearemanyassumptionsonwhichourmodelisbasedupon.Inthissection,wediscussaboutthe variousassumptions and behavior of processors based on these assumptions that helped us designing the improved version of the protocol in *(Abraham et al. 2006*). Below are some of the questions that arises when we make thoseassumptions:

- In *(Abraham et al. 2006),* processors do not know the size or IDs of other processors in the ring. So they are transferred during the wake-up phase and then the processors decide the direction for message passing. We assumed that we don't need a wake-up phase because our invasion of adversaries do not affect the wake-up phase nor it tries to deviate from the designated protocol. Hence, we assume that processors already know about the size and IDs of each processor. Our invasion protocol can be extended to execute honestly, the wake-up phase of the protocol in *(Abraham et al. 2006).*However, we cannot say that adversaries will never abuse this phase. Hence, the question still remains open fordiscussion.

- For rational agents, we defined that the output of the given protocol will be $[n] \cup [NULL]$. But the problem still revolves around the fact when IDs are unknown. Simply put, it is still difficult to find a resilient FLE protocol when the IDs are unknown. To understand this, let's consider an example. Let the IDs are from huge sample space $\sum$. Now, the expected utility of a processor, i will be, $EU: \sum \cup [NULL] \rightarrow [0,1]$.This function will be considered rational when $EU_i[NULL]=0$. Consider the protocol in (*Abraham et al. 2006*) to be P having the $EU_P[\sum]$.Although, an adversarial agent may or may not obey the protocol and abuse to have $EU_A[\sum] = \frac{k}{n}$.This means that in a unidirectional ring, for *k* adversaries with $k > 1$, there does not exist a k resilient protocol.

- In order to find a solution to this problem, what we can do is define an $\epsilon$-k-unbiased protocol for all the IDs. Let the deviation from every dishonest processor be $\forall x \epsilon \sum : Pr(output = x) \leq \epsilon + \frac{1}{n}$. Hypothetically, we proved that A-Lead$^{uni}$ and Acceptance based-A-Lead$^{uni}$ are $\epsilon$ - *k – unbiased* for *k* and $\epsilon$.

- Amongvariouschallengesthatwefacedtodesignaninvasionofadversarialprocessors,wetackle a particular problem which is an adversarial coalition can bluff the honest set into believing that there exist an Originator processor in every set. Considering the protocol discussed in *(Abraham et al. 2006)* and *(Abraham et al. 2013),* the processors tend to know the IDs of every other processor during the wake-up phase by exchanging and selecting the lowest ID among them to be the Originator. No honest set can have two Originator processors as each processor receives a setofIDswhichisusedtoidentifyifalltheprocessorsarehonestandthatnoIDisrepeated more thanonce.However,anadversarialcoalitioncanabusethisphasebyhidingthehighestIDamong honestprocessors$h \in H_i$and change it to 0 before forwarding it to the next set $H_j$, $j \neq i$.The adversary can use this highest ID to set it for itself.

## References


1. Abraham,Ittaiand Dolev, DannyandGonen,RicaandHalpern,JoeDistributedcomputingmeets game theory: robust mechanisms for rational secret sharing and multiparty computation, Proceedings of thetwenty-fifthannualACMsymposiumonPrinciplesofdistributedcomputing,53–62,2006.
2. Abraham, Ittai and Dolev, Danny and Halpern, Joseph Y.Distributed protocols for leader election: A game-theoretic perspectiveInternational Symposium on Distributed Computing 61–75.
3. Afek, Yehuda and Ginzberg, Yehonatan and Landau Feibish, Shir and Sulamy, Moshe, Distributed computing building blocks for rational agents Proceedings of the 2014 ACM symposium on Principles of distributed computing406–4152014.
4. Aiyer,AmitanandSandAlvisi,LorenzoandClement,AllenandDahlin,MikeandMartin,Jean-Philippe and Porth, CarlBAR fault tolerance for cooperative servicesProceedings of the twentieth ACM symposium on Operating systemsprinciples45–582005.
5. Ajtai,MiklósandLinial,NathanTheinfluenceoflargecoalitions,Combinatorica,132129–1451993Springer.
6. Andrychowicz,MarcinandDziembowski,StefanandMalinowski,DanielandMazurek,LukaszSecure multipartycomputationsonbitcoin2014IEEESymposiumonSecurityandPrivacy443–4582014 IEEE.
7. Bakhshi, Rena and Endrullis, Joerg and Fokkink, Wan and Pang, JunFast leader election in anonymous rings with bounded expected delayInformation Processing Letters11117864–8702011Elsevier.
8. Bar-Joseph,ZivandKeidar,IditandLynch,NancyEarly-deliverydynamicatomicbroadcastInternational Symposium on DistributedComputing1–162002Springer.
9. Bauk, SungUoon Revisiting the election problem in asynchronous distributed systemsInternational Workshop on Advanced Parallel Processing Technologies141–1502005Springer.
10. Boppana, Ravi B and Narayanan, Babu OPerfect-information leader election with optimal resilienceSIAM Journal on Computing 29 4 1304–13202000SIAM.
11. Chandra, Tushar Deepak and Toueg, SamUnreliable failure detectors for reliable distributed systemsJournal of the ACM (JACM) 432225–2671996ACM New York, NY, USA.
12. Chang, Ernest and Roberts, RosemaryAn improved algorithm for decentralized extrema-finding in circularconfigurationsofprocessesCommunicationsoftheACM225281–2831979ACMNewYork, NY,USA.
13. Clementi,AndreaandGualà,LucianoandProietti,GuidoandScornavacca,GiacomoRationalfair consensus in the gossip model 2017 IEEE International Parallel and Distributed Processing Symposium(IPDPS)163–1712017IEEE.
14. Coccoli, Andrea and Bondavalli, Andrea and Simoncini, LucaConsensus in asynchronous distributed systemsProc. of the 5th Int. Conf. on Integrated Design and Process Technology (IDPT'00)2000Citeseer.
15. Fang,ChunshengNovelFrameworksforMiningHeterogeneousandDynamicNetworks2011University ofCincinnati.
16. Fischer, Michael and Jiang, HongSelf-stabilizing leader election in networks of finite-state anonymous agentsInternational Conference On Principles Of Distributed Systems 395–4092006Springer.
17. Gordon,SDovandKatz,JonathanRationalsecretsharing,revisitedInternationalConferenceonSecurity and Cryptography forNetworks229–2412006Springer.
18. Heller, YuvalMinority-proof cheap-talk protocol, Games and Economic Behavior2010.
19. Hsieh,Hui-ChingandChiang,Mao-LunAnewsolutionfortheByzantineagreementproblemJournal of Parallel and DistributedComputing71101261–12772011Elsevier.
20. Izmalkov, Sergei and Lepinski, Matt and Micali, SilvioPerfect implementationGames and Economic Behavior711121–1402011Elsevier.
21. Lamport,LeslieandShostak,RobertandPease,MarshallTheByzantinegeneralsproblemConcurrency: the Works of LeslieLamport203–2262019.
22. Lepinski, Matt and Micali, Silvio and Peikert, Chris and Shelat, AbhiCompletely fair SFE and coalition-safe cheap talkProceedings of the twenty-third annual ACM symposium on Principles of distributedcomputing1–102004.
23. Lysyanskaya, Anna and Triandopoulos, NikosRationality and adversarial behavior in multi-party computationAnnual International Cryptology Conference180–1972006Springer.
24. Moses, Yoram and Dolev, Danny and Halpern, Joseph YCheating husbands and other stories: a case study of knowledge, action, and communicationDistributed computing13167–1761986Springer.
25. Pease, Marshall and Shostak, Robert and Lamport, LeslieReaching agreement in the presence of faultsJournal of the ACM (JACM)272228–2341980ACM New York, NY, USA.
26. Saks,MichaelArobustnoncryptographicprotocolforcollectivecoinflippingSIAMJournalonDiscrete Mathematics22240–

2441989SIAM.
27. Vida,PéterandForges,FrançoiseImplementationofcommunicationequilibriabycorrelatedcheaptalk: The two-player caseTheoretical Economics8195–1232013Wiley OnlineLibrary.
28. Zuckerman, DavidRandomness-optimal sampling, extractors, and constructive leader electionProceedings of the twenty-eighth annual ACM symposium on Theory of computing286–2951996.