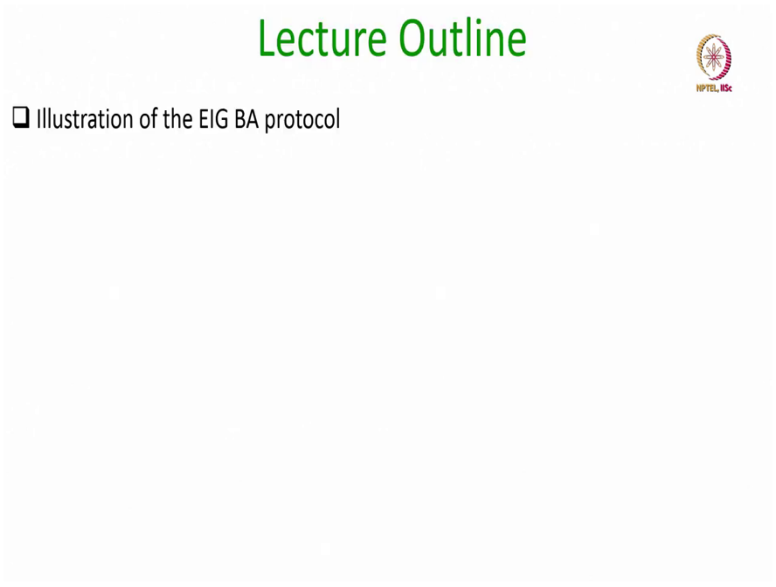
Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Information Technology, Bengaluru

Lecture - 04
EIG Protocol for Perfectly-Secure Byzantine Agreement: Illustration

Hello everyone, welcome to this lecture. In this lecture we will see an illustration of the EIG Byzantine agreement protocol.

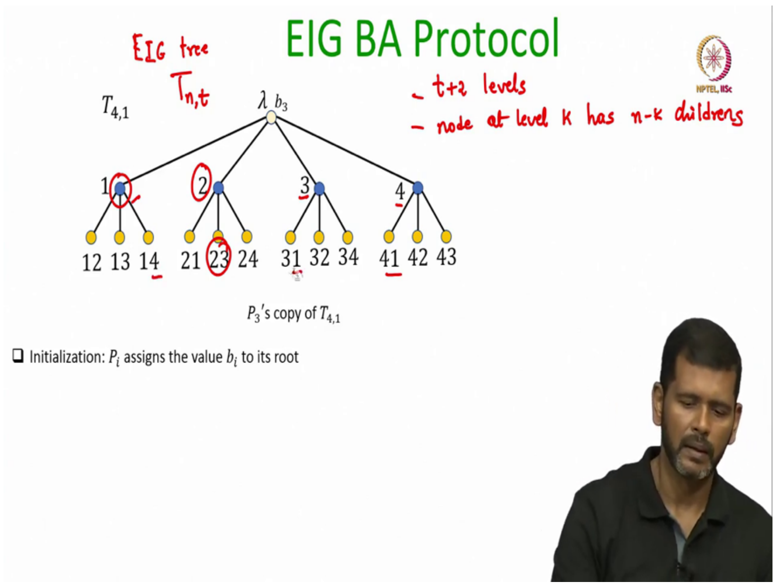
(Refer Slide Time: 00:30)



Lecture Outline

❑ Illustration of the EIG BA protocol

(Refer Slide Time: 00:31)



EIG BA Protocol

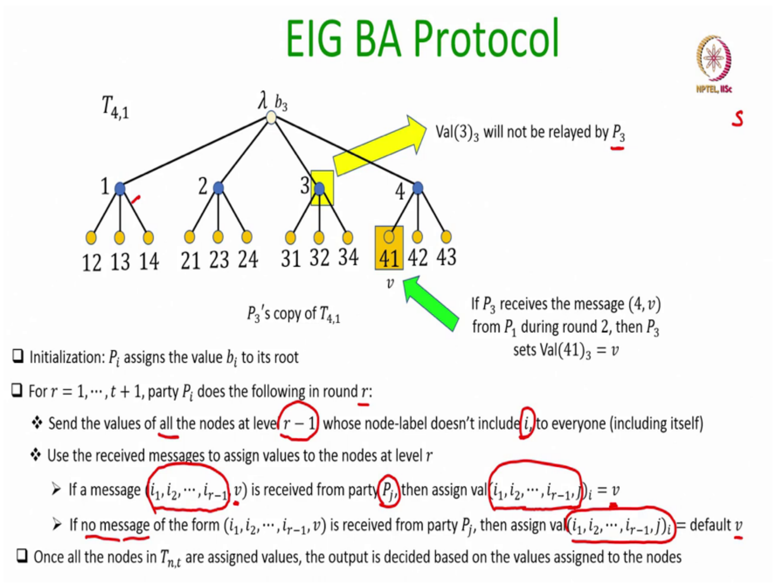
❑ Initialization: P_i assigns the value b_i to its root

So, let me quickly recall the EIG BA protocol. So, this is your EIG tree data structure denoted by $T_{n,t}$ and we have $t + 2$ levels. Node at level k has $n - k$ childrens and we have labels associated with each node consisting of strings, which has the identity of distinct parties.

Basically, the idea is whenever we go from root to any leaf node along that path there should be the identity of $t + 1$ distinct parties in any order. It need not be in the ascending order ok. So, for instance we have 3 followed by 1 here. We have 4 followed by 1 here or we have 1 followed by 4 here and so on.

So, in the protocol the parties exchange messages for $t + 1$ number of rounds. Basically what they do is at each round they exchange the values of their respective nodes at some specific layer with each other and based on whatever values they have received from other parties they decorate or assign values to the children's at the next layer and they do this process.

(Refer Slide Time: 02:27)



EIG BA Protocol

$T_{4,1}$
λ b_3
1
2
3
4
12 13 14
21 23 24
31 32 34
41 42 43
v
P_3's copy of $T_{4,1}$
S

Val$(3)_3$ will not be relayed by P_3

If P_3 receives the message $(4, v)$ from P_1 during round 2, then P_3 sets Val$(41)_3 = v$

- Initialization: P_i assigns the value b_i to its root
- For $r = 1, \cdots, t+1$, party P_i does the following in round r:
  - Send the values of all the nodes at level $r-1$ whose node-label doesn't include i, to everyone (including itself)
  - Use the received messages to assign values to the nodes at level r
    - If a message $(i_1, i_2, \cdots, i_{r-1}, v)$ is received from party P_j, then assign val$(i_1, i_2, \cdots, i_{r-1}, j)_i = v$
    - If no message of the form $(i_1, i_2, \cdots, i_{r-1}, v)$ is received from party P_j, then assign val$(i_1, i_2, \cdots, i_{r-1}, j)_i$ = default v
- Once all the nodes in $T_{n,t}$ are assigned values, the output is decided based on the values assigned to the nodes

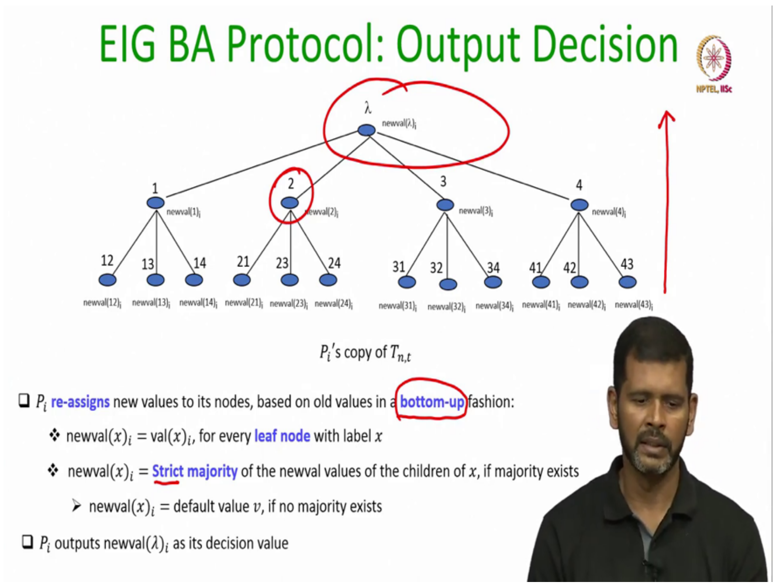
So, more specifically for r equal to 1 to $t + 1$, party P i does the following in round r; it sends the values of all the nodes at layer r - 1 except for those nodes whose tag includes i and those values are sent to everyone including the party itself ok. So, for instance the value of this node will not be communicated by P 3 during round 2.

Now, based on the messages received by every party, they decorate or assign values to the node at the next level. Namely if P j is reporting to P i that the value of the node with this tag was having value v, then P i decorates the node at the next layer whose tag is this with the value v.

However, if no message comes in that round number r; that means, Pj was corrupt because if Pj is not corrupt then it should have sent a message in that round. But, if P j is corrupt and does not send a message then Pi uses some default value right, it does not wait indefinitely for that expected message.

And then once all the nodes in Pi's EIGs tree is assigned values it stops interacting with other parties and then recompute the values of all the nodes in its local copy of the EIG tree and then whatever value it recomputes for its root node it takes that to be the output of the BA protocol.

(Refer Slide Time: 04:29)


λ
newval(λ)$_i$
1
newval(1)$_i$
2
newval(2)$_i$
3
newval(3)$_i$
4
newval(4)$_i$
12
13
14
21
23
24
31
32
34
41
42
43
newval(12)$_i$
newval(13)$_i$
newval(14)$_i$
newval(21)$_i$
newval(23)$_i$
newval(24)$_i$
newval(31)$_i$
newval(32)$_i$
newval(34)$_i$
newval(41)$_i$
newval(42)$_i$
newval(43)$_i$

EIG BA Protocol: Output Decision

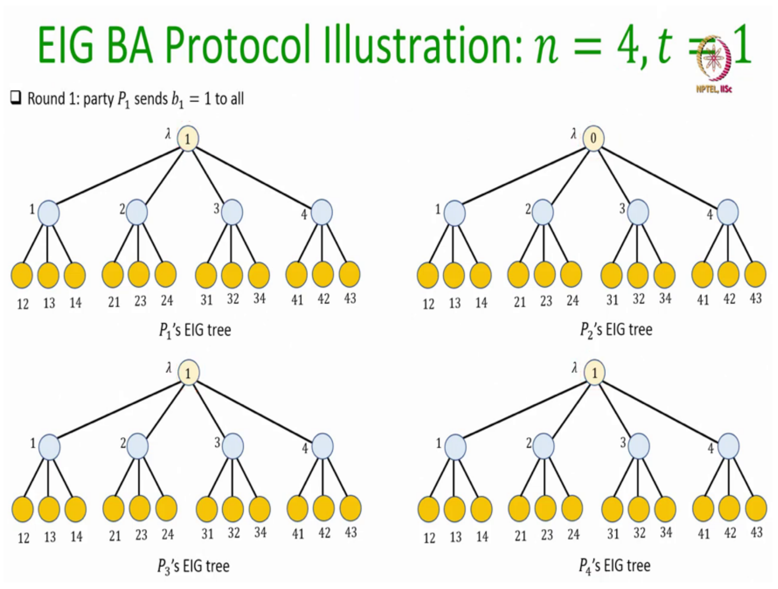
P_i's copy of $T_{n,t}$

- ❑ P_i **re-assigns** new values to its nodes, based on old values in a **bottom-up** fashion:
  - ❖ newval$(x)_i$ = val$(x)_i$, for every **leaf node** with label x
  - ❖ newval$(x)_i$ = **Strict majority** of the newval values of the children of x, if majority exists
    - ➢ newval$(x)_i$ = default value v, if no majority exists
- ❑ P_i outputs newval$(\lambda)_i$ as its decision value

So, for instance the reassignment process is done in a bottom of fashion. The reassignment of the leaf nodes remain the same, they are not assigned any new value they remain the same as it was earlier. And then as we go up if we take node x namely a node with label x, then the new value for that node will be taken to be the majority of the new value or the recomputed value of its children.

And remember those values would have been already recomputed, the recomputed values or the new values would have been already computed for the children of node labeled with x, because we are doing the reassignment in the bottom up fashion, However, if the strict majority does not exist then some default value is taken and once we finish this reassignment process from bottom to up the root will be now assigned a new value, which is taken to be the output of the BA protocol. So, that is the EIG BA protocol.

(Refer Slide Time: 05:53)


EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 1: party P_1 sends $b_1 = 1$ to all

Now, let us see an illustration to make it simpler. For the purpose of illustration I am taking n to be 4 and t to be 1. Why we are taking these specific values? Because remember I told earlier that in order that the protocol should achieve the properties we require n and t to satisfy the relationship that n should be strictly greater than 3t. So, if I take t to be 1, then the smallest value of n which satisfies the condition n greater than 3t is 4. Of course, we could take n equal to 5, t equal to 1 n to be 6, t to be 1 and so on.

But, let us take the smallest value of n satisfying the condition n greater than 3t. Of course you can take higher values of t say t to be 2, in which case you have to assign the value of n to be at least 7 and so on. And then you take any choice of inputs for the parties and then you can run the protocol and you will see that somehow all the properties of Byzantine agreement are satisfied by the protocol ok, but for our illustration we will take the case when n is 4 and t is equal to 1.

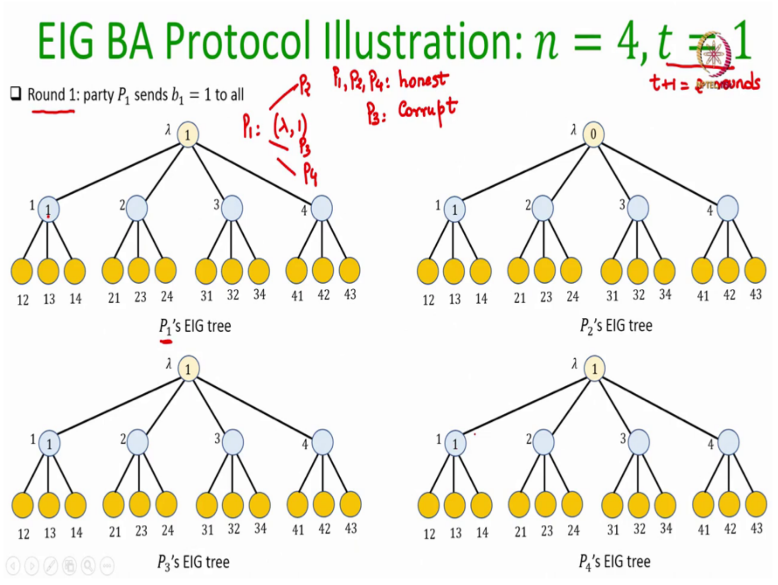
Now, I take the initial values of the parties to be 1011 and since one of the 4 parties could be corrupt it could be any of the 4 parties. For the purpose of illustration, I am taking the case when P3 is going to be corrupt during the protocol execution. Again I stress this will not be known, this will not be an information which is known beforehand to everyone. The parties will only know that at most one of the 4 parties is going to behave maliciously in a byzantine fashion during the protocol execution or it could be the case that no party deviates.

But, if at all any deviation or any corruption happens it could be only one corruption, but which party is going to be exactly corrupted will not be known beforehand ok. Also the respective inputs of the parties will not be known to each other. So, let us see the initialization process of the EIG tree ok. So, recall that each party maintains a local copy of its EIG tree. So, P1 have its own copy of the EIG tree and what it will do is it will assign the input that P1 has for the BA protocol to the root node.

So, P1's input is 1. So, that is why it is assigning the value 1 to its root node. P2's input is 0, that is why in P2's copy of the EIG tree the value 0 will be assigned to the root node by P 2. Now P3 is corrupt and we do not know what exactly is the input that P 3 would like to consider in the protocol. It completely depends upon P 3 because we have absolutely no control on the behavior of the corrupt parties.

So, imagine for the moment that P3 sticks to the input 1. So, it assigns the value 1 to the root node in its local copy of the EIG tree and since P4's input is 1, P4 assigns the value 1 to the root node in its local copy of the EIG tree ok that is the initialization process.

(Refer Slide Time: 09:41)



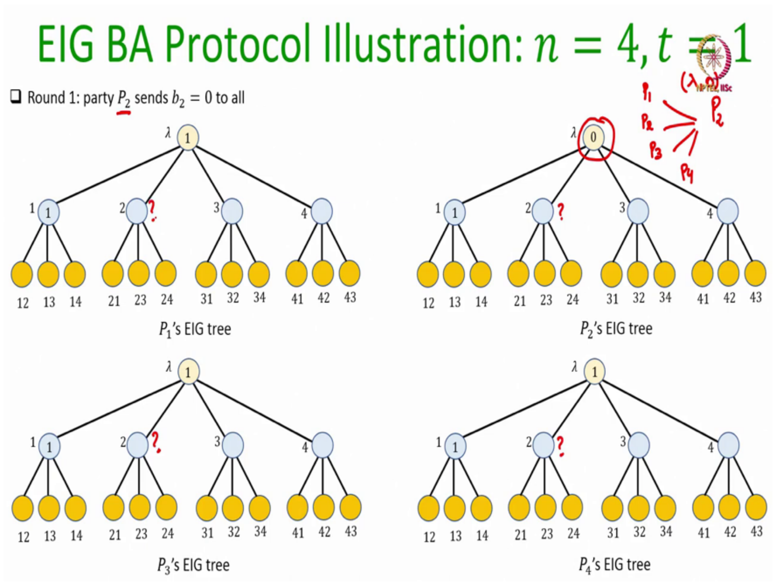
Now, since t is equal to 1, there will be total t + 1 = 2 rounds of message exchange. So, let us see what are the values which are communicated by the respective parties and when I say rounds and parties are exchanging messages they will be exchanging messages in parallel. So, it is not the case that when P1 is exchanging messages with others, P2 simply sits ideal, it is not the case. P2 also will be exchanging the messages as per its protocol instructions for the round 1 and so on.

So, in round 1 since P 1 is honest remember we are taking the case when P1, P2 and P4 are honest, they will follow the EIG protocol instructions and P 3 is corrupt and we have absolutely no control, how P 3 is going to behave during the EIG protocols execution. So, during round 1 what P 1 is supposed to do as per the EIG protocol? It is supposed to send the value of its root node to everyone else and since P 1 is honest it will do it as per the protocol instructions.

So, it will tell to everyone that my root node; that means, basically P1 will send the message that the node with label lambda has the value 1 and this it will send to P 2, this it will send to P 3, this it will send to P 4 simultaneously. Remember we have a channel between every pair of parties ok. Now, when P 1 is sending this information to everyone, as a result of that what is going to happen. So, you see at the time of initialization no value was assigned to the nodes with these labels.

The nodes with these label will be assigned values only when P 1 tells about the value of its root node during round 1 and now P 1 is reporting that the value of its root node is 1 ok. So, that is why at the end of round 1, everyone will assign the value 1 ok to the node with the label 1 in their respective copies of the EIG tree ok.

(Refer Slide Time: 12:34)

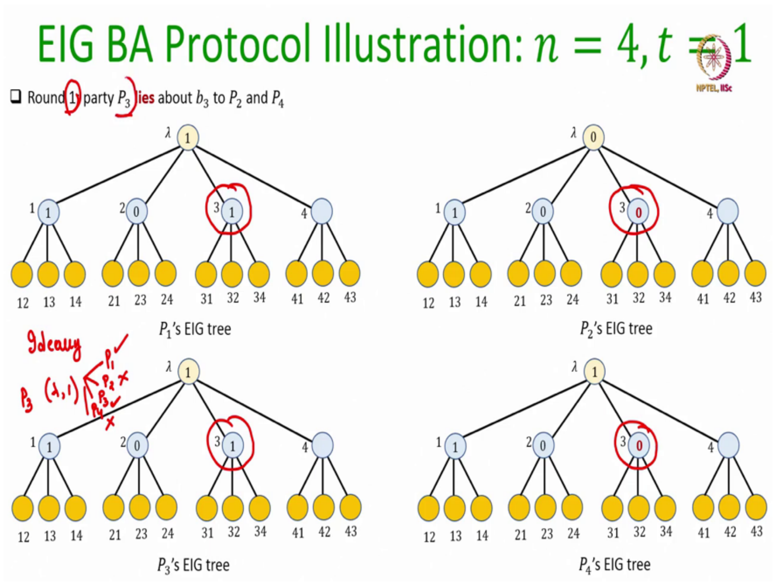


EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 1: party P_2 sends $b_2 = 0$ to all

Now, in the same round, round number 1, P2 will tell that its input is 0 and since P2 is honest it will do it identically for everyone. So, P2 will inform everyone that in my EIG tree the node with label lambda has been assigned a value 0 and this information it will give to 1, this will it will give to 2, it will give to 3 and it will give to 4 during round number 1 and due to this what will happen is the value of the node with label 2 will be assigned a value at the end of round 1; it will be now assigned a value 0.

(Refer Slide Time: 13:29)



EIG BA Protocol Illustration: $n = 4, t = 1$

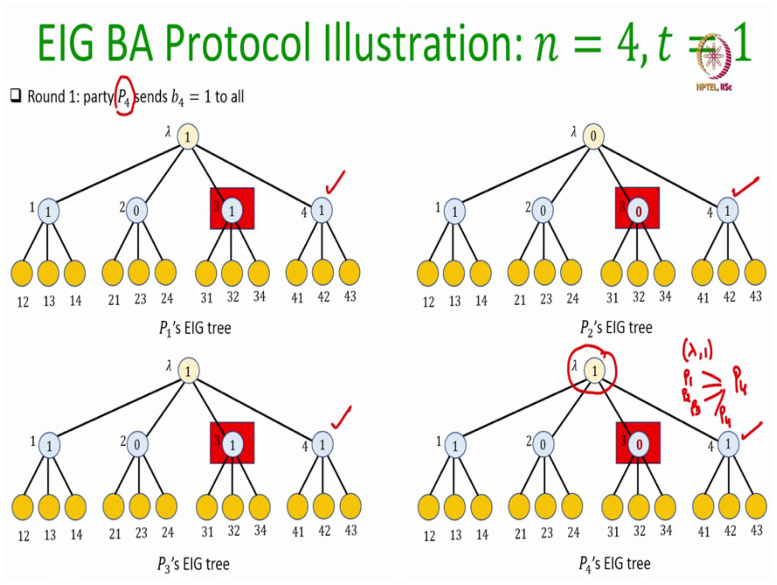
- Round 1: party P_3 lies about b_3 to P_2 and P_4

Now, recall we assume that P3 is corrupt we are still in round number 1, as per the EIG protocol P3 is supposed to tell everyone that the value of its root node has been the value of its root node is 1, that is what P3 is supposed to do. But, P3 now behaves maliciously, it lies to party 2 and 4 about its root value. Well P3 can lie to anyone, as I said the maliciously corrupt party can behave in any arbitrary fashion.

So, consider a strategy where adversary instructs P 3 that ok you lie about your root node to 2 and 4; that means, ideally P 3 is supposed to inform everyone that the value with root node lambda is assigned the value of the root node with the label lambda is 1 that is what it is supposed to tell everyone, but it is not doing so. It is doing so for 1 and 3; that means, to party number 1 and party number 3 it is telling correctly that it has assigned the value 1 to its root node, but to 2 and 4 it is telling a different version.

To 2 and 4 it is telling that 3 has assigned the value 0 to its root node. And due to this you see what is happening here. Since 1 and 3 have received correctly the value of P 3's root node, they have assigned the value 1 to the node with label 3. But if you see the P 2’s copy of the EIG tree and P 4’s copy of the EIG tree and the node with label 3 they are assigned a value 0. This is because P 3 have behaved maliciously ok.

(Refer Slide Time: 15:46)

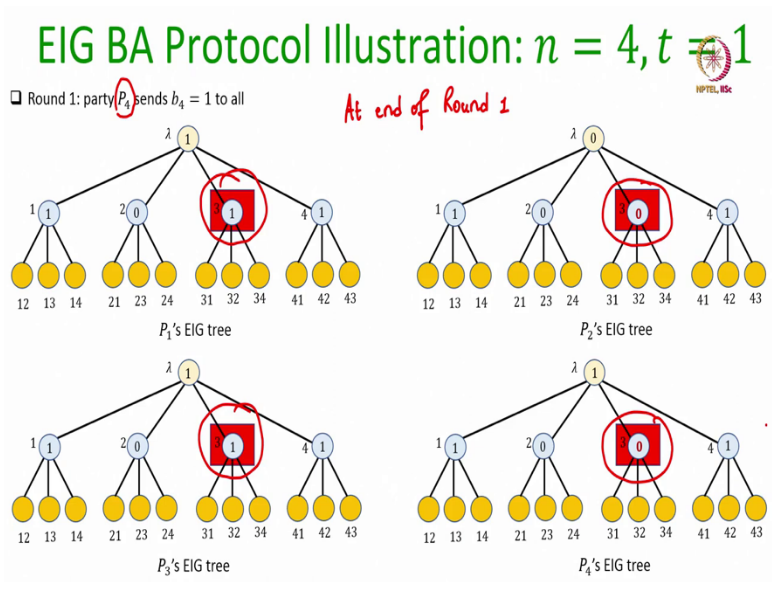


EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 1: party P_4 sends $b_4 = 1$ to all

And in the same round, P4 as per the protocol instruction is supposed to tell the value of its root node to everyone else. So, P1, P4 will send the message that the value of the root node is 1 to everyone. And since P 4 is honest it will send the identical message to everyone and as a result of this, everyone will assign the value 1 to the node with label 4 in their respective copies of the EIG tree.

(Refer Slide Time: 16:32)



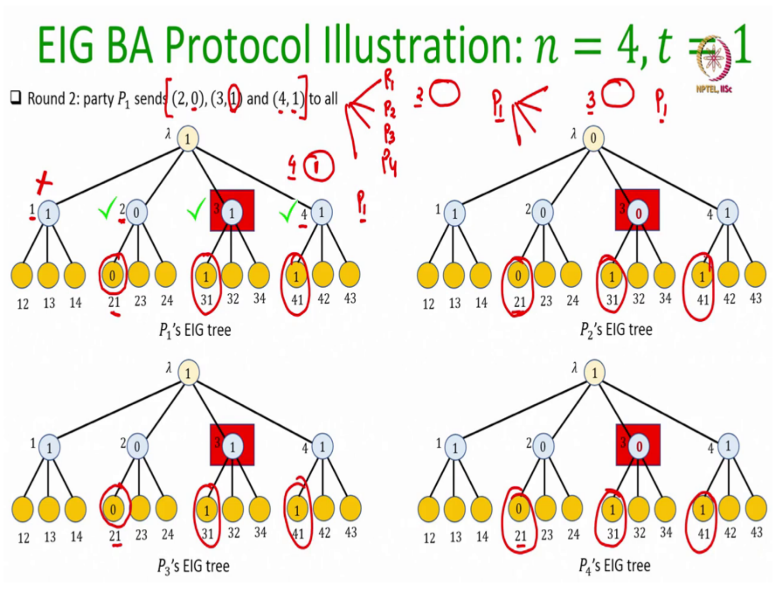
EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 1: party P_4 sends $b_4 = 1$ to all

So, that is the status after the end of round 1. At the end of round 1 this will be the view of respective copies of the EIG tree. At the end of round 1 this will be the view of P1's EIG tree. This will be the view of P2's EIG tree and remember no one will be knowing the view of other parties EIG tree ok.

Now, you can see that since P 3 has lied about is input value to 2 and 4, the value which I have highlighted the node which I have highlighted here ok, this node has been assigned different values in the respective copies of EIG tree and the parties will not be knowing this because they do not know who is corrupt and who has communicated the right information, who has communicated the wrong information fine.

(Refer Slide Time: 17:39)



EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 2: party P_1 sends $(2, 0), (3, 1)$ and $(4, 1)$ to all

Now, let us see what happens during round number 2. Now in round number 2 let us see what P 1 is going to do. So, P 1 is supposed to send the value of all the nodes at layer 1, except for this node. Why the label of this node will not be communicated, because it has a label which includes party number 1 whereas, the labels of these nodes do not include the party number 1 as one of the indexes right.

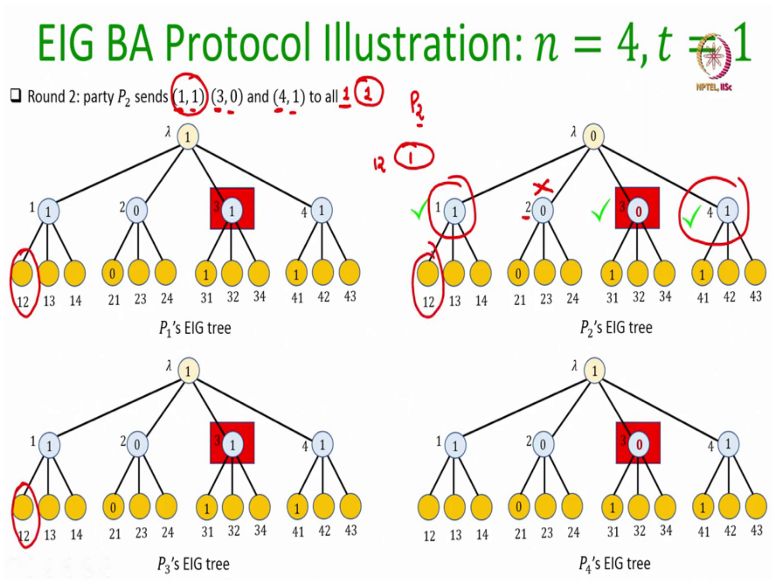
So, that is why party 1 will send this message to everyone. It will tell everyone that ok hey I have assigned the value 0 to the node with label 2. I have assigned the value 1 to the node with label 3 and I have assigned the value 1 to the node with label 4 and this it will send to everyone identically, ok.

Now, this message will be received by everyone ok. Now based on this message you see every party is going to decorate the node with label 21 in their respective copies of their EIG tree. Why 21, because the party number 1 is sending this information to everyone else regarding the node with label 2. So, 2 followed by 1 will be the node which get decorated with the value 0.

In the same way party number 1 is telling some information about the node which has the label 3. So, 3 followed by 1 that will be the node which gets decorated with the value 0 based on the information that P1 is communicating. So, in the same way P1 is telling everyone that in its copy of the EIG tree the node with label 4 has been assigned the value 1.

So, that is why 4 followed by 1 will be assigned the value 1 in every party’s EIG tree, namely this node. That is what P1 will do. In parallel, remember as I said and I say communication round everyone is communicating to every other party in the particular round parallelly.

(Refer Slide Time: 20:35)



EIG BA Protocol Illustration: $n = 4, t = 1$

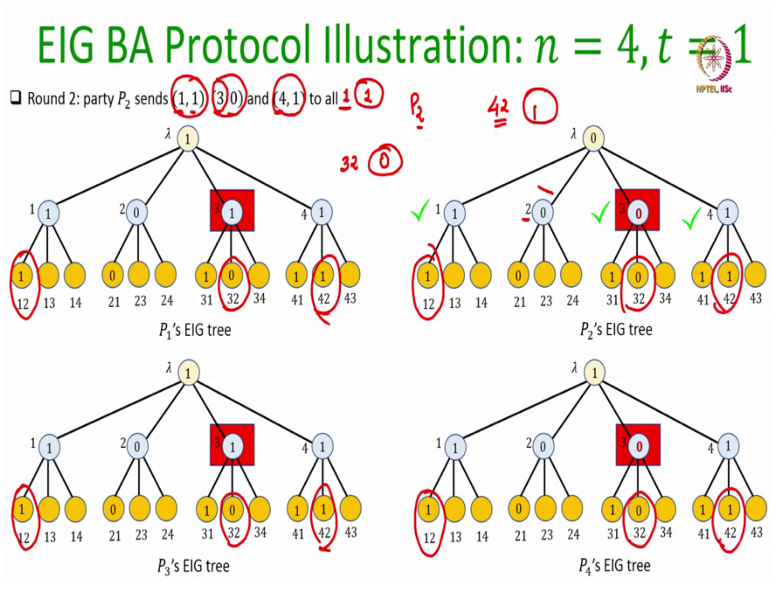
- Round 2: party P_2 sends $(1, 1)$ $(3, 0)$ and $(4, 1)$ to all

So, in round number 2: P 2 will send the value of this node, the value of this node and the value of this node to everyone else ok. Sorry the value of this node will not be communicated because its label include the identity of party 2 ok. So, 2 will tell that ok the node with label 1 has the value 1 in its EIG tree.

The node with label 3 has the value 0 in its EIG tree and the node with label 4 has the value 1 and its EIG tree ok. So, let us see based on this information how the parties populate or decorate the nodes at layer 2 ok. Let us analyze the first message. So 2 is telling to everyone that a node whose label was 1 has the value 1. Who is telling? 2 is telling and what was the label of the node that it is talking about? 1.

So, 1 followed by 2 that will be the node, which will be populated by everyone and what value it will be assigned? The value which P2 is informing regarding the node in the previous layer so the value that it is reporting is 1. So, that is why every party will assign the value 1 to the node whose label is 12, right.

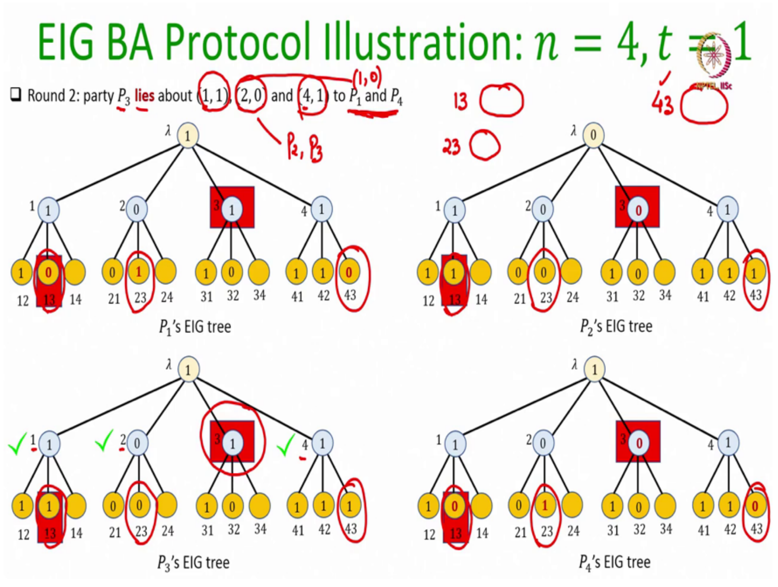
(Refer Slide Time: 22:46)


EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 2: party P_2 sends (1, 1), (3, 0) and (4, 1) to all

In the same way based on this message every party will populate the node with label 32, because 2 is informing about the node with label 3 and that value will be as the value that will be assigned will be 0. So, 32 right so this, this ok. In the same way based on this message the node 42 will be assigned the value 1 in all the EIG trees ok.

(Refer Slide Time: 23:37)


EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 2: party P_3 lies about (1, 1), (2, 0) and (4, 1) to P_1 and P_4

Now, let us see what P 3 does in round 2. Remember P 3 is corrupt. In round 1 it has lied to 2 and 3 and it honestly communicated to 1 and 4, but now it might change a strategy. In round number 2 it might decide to lie to different set of parties and that is quite possible that is allowed as per the Byzantine behavior. Because now P3's system is completely under the control of an adversary, its completely hacked and the hacker or the adversary can force P3 to communicate in whatever fashion it would like too ok.

So, first of all what are the value what are the nodes for which P 3 should send the values? Except for this node it should send the value with it should send the value of the node with label 1 with label 2 and label 4 right, but it has decided that ok it will send the values, but it will lie about the values of these nodes to 1 and 4; that means, to 2 and 3 it will send correctly, but to 1 and 4 it will send incorrectly.

Now, let us see what is the consequence of this. So, let us see what will be the consequence of the first message ok. So, who is sending? 3 is sending, about which node, the node with label 1. So, ideally everyone should decorate the node with label 1 3 with the value 1 if P 3 would have not lied to everyone else. But, P3 has decided that it will tell it correctly to 2 and 3, but it will tell it incorrectly to 1 and 4.

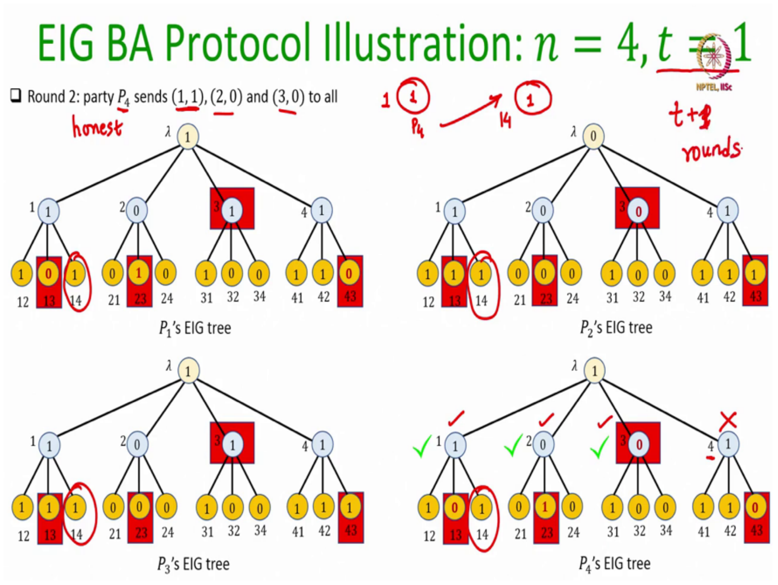
So, let us see what happens to this node. So, to 1 and 4 it is going to lie; that means, instead of sending 1 1 it is sending 1 0 whereas, ideally it should send 1 1. So, that is why you see in P1's EIG tree this value is the value that is assigned here is 0 and P 4 EIG tree that is the value that is that the value that is assigned here is 0. Whereas, in 3s copy and 2s copy the values are correct. In the same way based on the second message right, everyone should populate the node with label 23, but now P1 and P4 will have different values for the node with label 23 compared to the parties 2 and 3 ok.

You see, based on the last message everyone is supposed to populate the node with label 43, but again since P3 is telling different versions of the node with label 4, party number 1 and 4 will have different values for this node compared to party 2 and 3 ok. So, 1 and 4 will assign the values 0 will assign the value 0 whereas, 2 and 3 will assign the value 1 ok.

So, again now you can see if I see across all the EIG trees there are several nodes where different parties have assigned different values to a node with some specific label ok. So, for instance the node with label 3, it has different values across different EIG trees because during round 1, 3 lied and then if you see the node with label 1 3 in different EIG trees, again the values are different because again in round 2 party P 3 has lied ok.

In fact, all these nodes have different values. So, all the nodes ending with label 3 at layer 1 and all the nodes with labels ending with 3 at layer 2 they have different values, because party 3 has lied during round 1 and round 2.

(Refer Slide Time: 28:31)



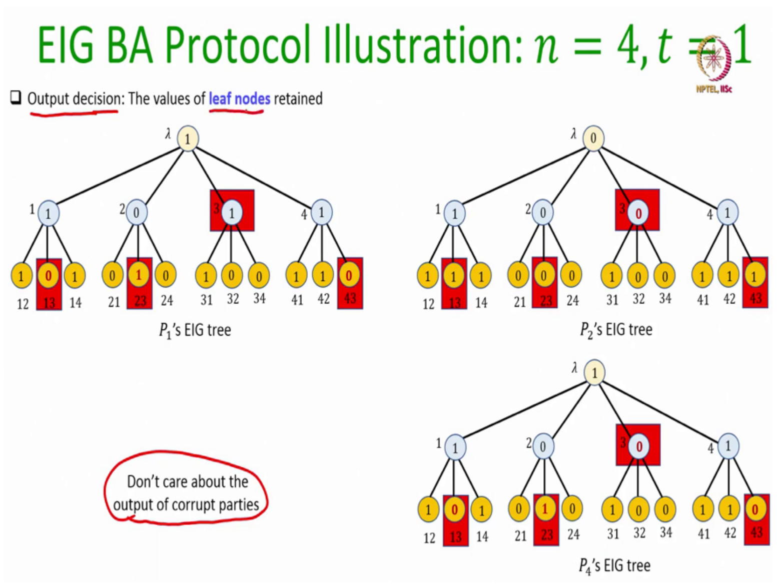
EIG BA Protocol Illustration: $n = 4, t = 1$

- Round 2: party P_4 sends $(1,1)$, $(2,0)$ and $(3,0)$ to all

Now, round 2 is still left for party number 4 ok, and party number 4 will send the value of this node, this node and this node. It would not send a value of this node because it has 4 included in the label. And since P 4 is honest, It will send the value of the nodes identically to everyone ok.

So, for instance let us see how this message will be used. So, P 4 is telling everyone that this is the status in its EIG tree due to which every party will take the node with label 1 4 and assign the value 1 ok, and in the same way you can process these 2 messages as well right. So, the protocol is done because remember we are in the setting where t is equal to 1 and the protocol requires only t + 1 rounds.

(Refer Slide Time: 29:54)



EIG BA Protocol Illustration: $n = 4, t = 1$

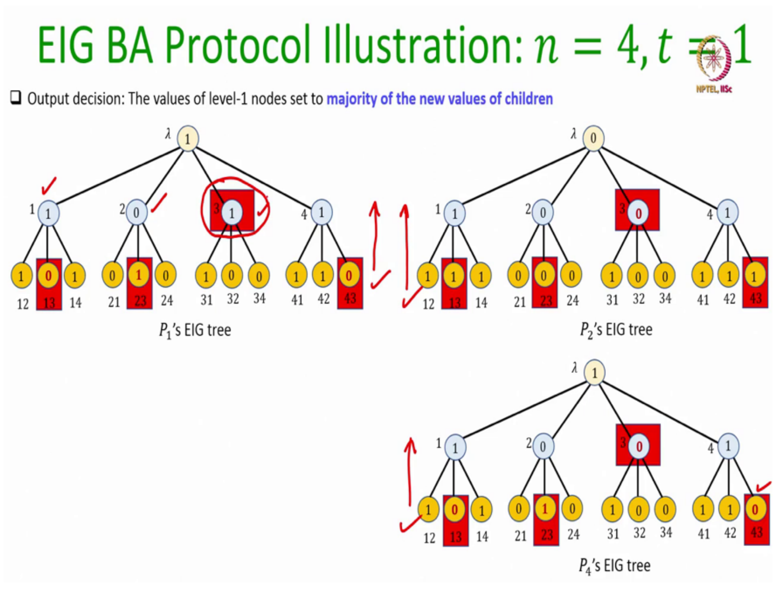
- Output decision: The values of leaf nodes retained

Don't care about the output of corrupt parties

So, now there will be no further communication ok. Now the parties will go to the output decision where they have to re-compute the values of each node and then based on the recomputed value of their respective root nodes they will take that value as the overall output for the BA protocol ok. So, we do not care about what is the output of the corrupt parties because, corrupt parties can do anything regarding their output. In fact, they may not even consider any output whatsoever and we have absolutely no control.

Remember the goals of the BA protocol is only to ensure that the honest participants the good participants they terminate they have a common output and so on. We do not say anything regarding the output or the behavior of the corrupt parties ok, output behavior of the corrupt parties because we have absolutely no control. So, let us see the re-computation process here. So, the values of the leaf nodes will be retained by every party as it is ok.

(Refer Slide Time: 30:59)

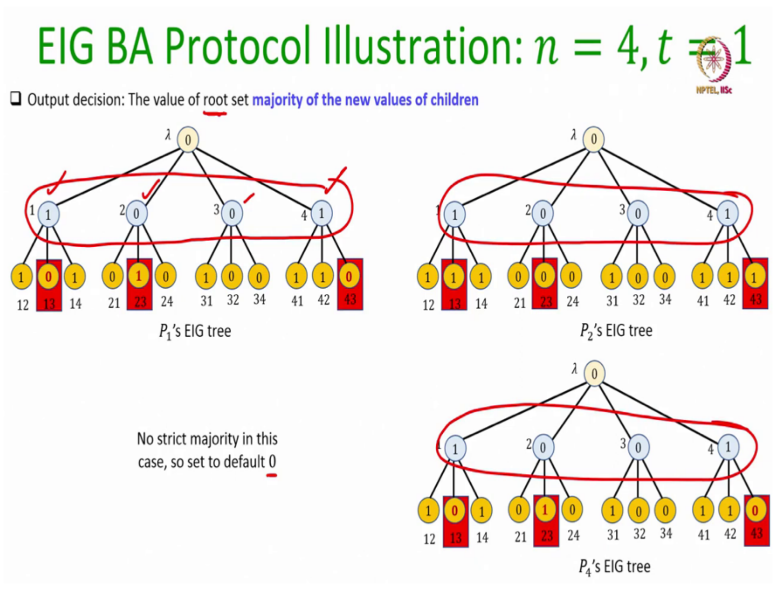

EIG BA Protocol Illustration: $n = 4, t = 1$

- Output decision: The values of level-1 nodes set to **majority of the new values of children**

So, that is done no change here ok. Now, the value of the next level, higher level will be decided based on the majority of the new values of the children. So, if I take this node its children has the new values 101, so its majority is 1. This node its children has the value 010 so, majority is 0 here the majority is now for this ok. So, for this particular node namely the node with label 3 you see the values which were computed by different EIG parties in the respected EIG trees were different ok.

In P 1's copy the value was 1, in P 2's copy the value was 0 and P 4's copy the value was 0. But, as soon as we do the reassignment process and take the majority of the new values of the children you see a magic happens. All the new values which are computed for the node ending with 3 turns out to be 0 only because the majority turns out to be 0; Even though the old values were different I stress, the old values were 100 respectively, but now the new value is 0 across all the EIG trees ok.

(Refer Slide Time: 32:48)



EIG BA Protocol Illustration: $n = 4, t = 1$

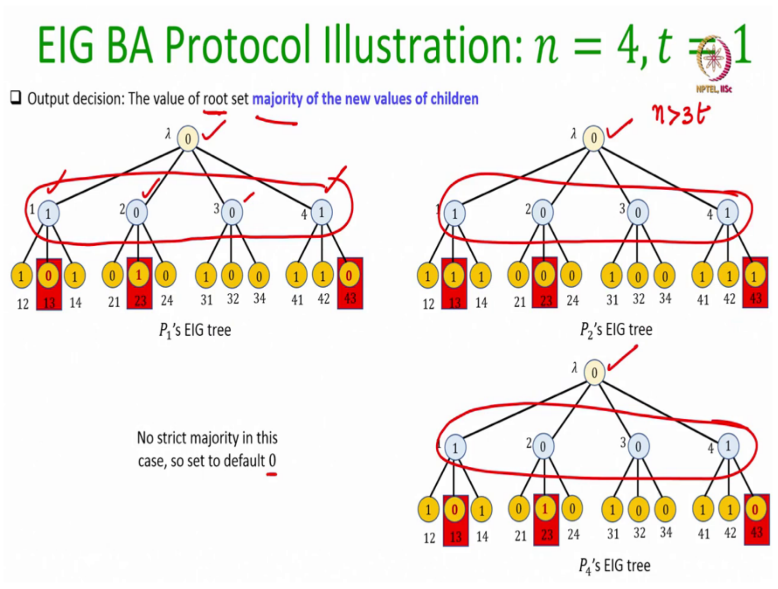
❑ Output decision: The value of root set **majority of the new values of children**

No strict majority in this case, so set to default 0

Now, we go to the root node and follow this majority rule. So, if we take P 1’s tree what are the new values for the children 1001 and there is no majority here. 0 is occurring twice, 1 is occurring twice. So, 1 and 1 occurring twice 0 and 0 occurring twice so no majority; So, we can take a default value 0 as the root and you see here closely what is happened is the new values of the root node in all the EIG tree turns out to be the same.

It was 1000; 1001 here also it is 1001 here also it is 1001. So, even though if we see the values the old values which were computed by the different EIG trees they were different ok recall if I go 2 slides back ok.

(Refer Slide Time: 33:42)



EIG BA Protocol Illustration: $n = 4, t = 1$

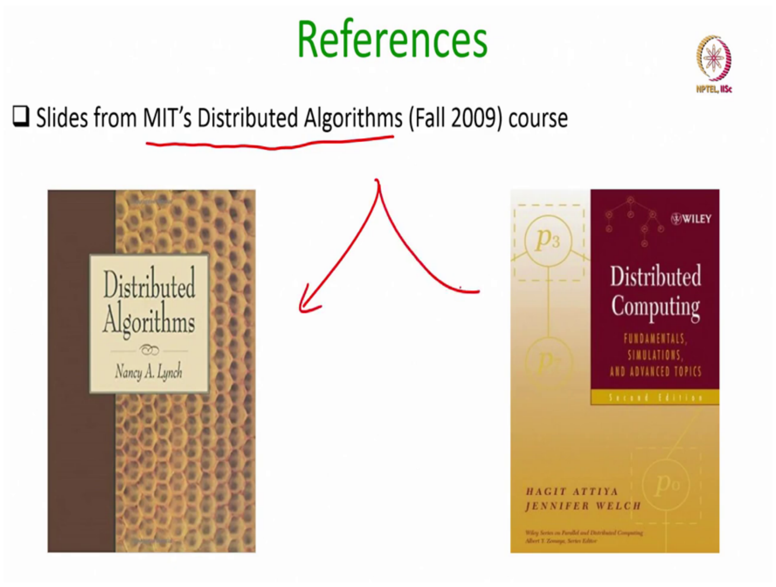
- Output decision: The value of root set **majority of the new values of children**

No strict majority in this case, so set to default 0

The old values of this particular child of the root was different across different EIG trees, but now because of the reassignment process what has happened is that the new values of all the children of the root turned out to be the same. Well you might be wondering that this is happening this might be happening just for this example, but later on we are going to prove that that is going to happen in general for any execution of the EIG protocol provided your n is greater than 3 t and you are using this majority rule ok.

So, every party will now output the value 0 ok, which ensures that the consistency properties satisfied. So, that is an illustration of the EIG BA protocol exponential information gathering BA protocol. I hope through the illustration the protocol is clear of course, we have to formally prove the security properties, which we will do in the subsequent lectures.

(Refer Slide Time: 34:42)



References

❑ Slides from MIT's Distributed Algorithms (Fall 2009) course

Distributed Algorithms
Nancy A. Lynch
WILEY
Distributed Computing
FUNDAMENTALS, SIMULATIONS, AND ADVANCED TOPICS
Second Edition
HAGIT ATTIYA
JENNIFER WELCH
NPTEL, IISc

So, the references for today's lecture are the same as for the earlier lecture, namely I have used heavily the slides from MITs Distributed Algorithm course and of course, if you want the details they are available in these two textbooks.

Thank you.