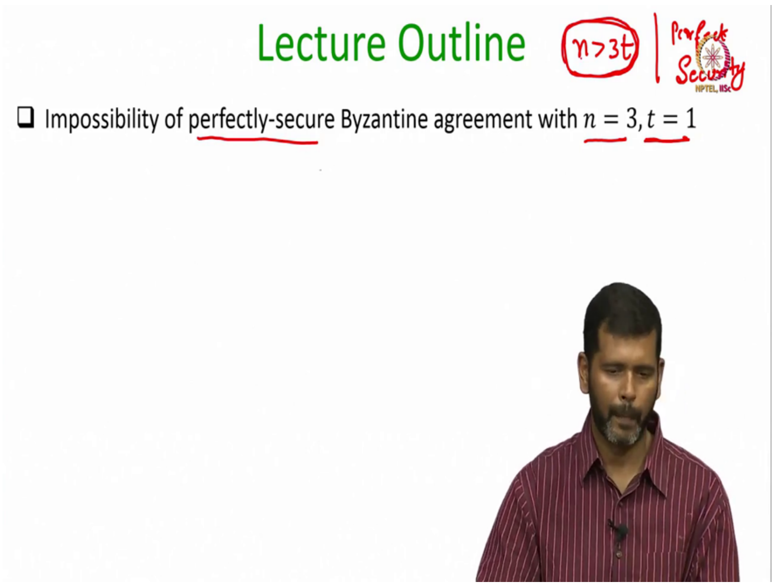
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 15
Lower Bound for Number of Parties for Byzantine Agreement: Part I

(Refer Slide Time: 00:27)



Lecture Outline

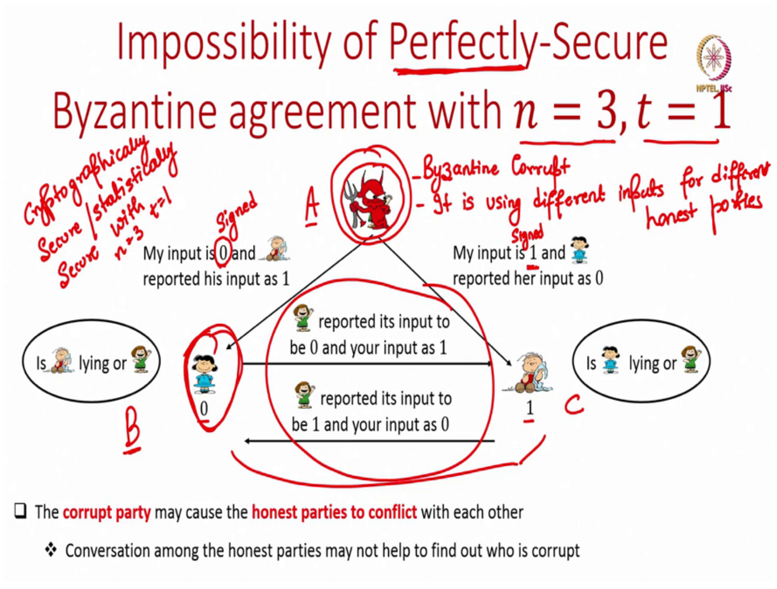
$\boxed{n > 3t}$ | Perfect Security

- Impossibility of perfectly-secure Byzantine agreement with $n = 3, t = 1$

Hello everyone. Welcome to this lecture. Till now we have designed byzantine agreement protocols assuming n greater than 3t ok. And our goal was perfect security. So, a natural question is, is this a necessary condition, if you want to get perfect security and the answer is yes. So, now, for the next few lectures we will focus on deriving this necessary condition, that why this condition n greater than 3t is necessary for perfectly secure byzantine agreement.

So, we will first show that it is impossible to design a perfectly secure byzantine agreement protocol with 3 parties and 1 corruption ok. And then in the follow up lecture we will generalize the proof and use this special case of n equal to 3 and t equal to 1 to show that this condition n greater than 3 t is necessary.

(Refer Slide Time: 01:33)



Impossibility of Perfectly-Secure Byzantine agreement with $n = 3, t = 1$

Cryptographically Secure / statistically Secure with n=3 t=1
A
- Byzantine Corrupt
- It is using different inputs for different honest parties
Signed
My input is 0 and reported his input as 1
Signed
My input is 1 and reported her input as 0
reported its input to be 0 and your input as 1
reported its input to be 1 and your input as 0
Is lying or
0
B
Is lying or
1
C

- The corrupt party may cause the honest parties to conflict with each other
  - Conversation among the honest parties may not help to find out who is corrupt

So, let us try to understand why it is impossible to design a perfectly secure byzantine agreement protocol with n parties where n is equal to 3 and one of them being corrupt. So, if one of the parties out of these three parties is corrupt and say the remaining two honest parties have different inputs. One of the honest parties has input 0 and the other honest party has the input 1. Then what this corrupt party can do is that it can always cause the honest parties to conflict with each other in terms of whatever they conclude.

Specifically, this corrupt party may behave towards this honest party as if its input is 0 and other party's input is 1. Whereas, to the other honest party it might pretend, it might behave as if its input is 1 and other party's input is 0. So, you see this party is byzantine corrupt and it is using different inputs for different honest parties towards this party. So, let us call these three parties as A, B, C. So, A is corrupt and A is reporting to B that its input is 1, yeah sorry its input is 0.

So, to B it will look as if A's input is 0 and her own input is 0. So, it will take 0 as the output, but towards C the same party A is now reporting that its input is 1. So, to C it will look like as if C and A together have inputs 1. So, C will output 1 and that will be a violation of the agreement property because B will be outputting 0 and C will be outputting 1. So, consistency or the agreement property is violated.

Now, you might be wondering that why cannot B and C talk with each other in the protocol and find out whether A is trying to confuse them and get into conflict. So, if B and C talk with each other then that is not going to help because B will report to C that A has reported its input to be 0 and C's input as 1 whereas, C will report to B that A has reported its input to be 1 and B’s input as 0.

Now, from the viewpoint of the party B, it is completely confused whether it is the party A who is lying or whether the party C who is lying and same is the scenario from the viewpoint of the party C. C is totally confused whether the party B is lying or whether the party A is lying. So, this communication among the two the honest parties, it is completely useless to find out who is lying whether A is causing this confusion among them.

And that is precisely is the intuition that why cannot we design a perfectly secure BA protocol with three parties and one of them being corrupt. Of course, we can design a cryptographically secure byzantine agreement protocol ok. So, we can design cryptographically secure protocol as well as a statistically secure protocol with n equal to 3 and t equal to 1. There this argument fails, there this argument breaks because we can have a protocol where every party associates its own signature with whatever message it is communicating to other.

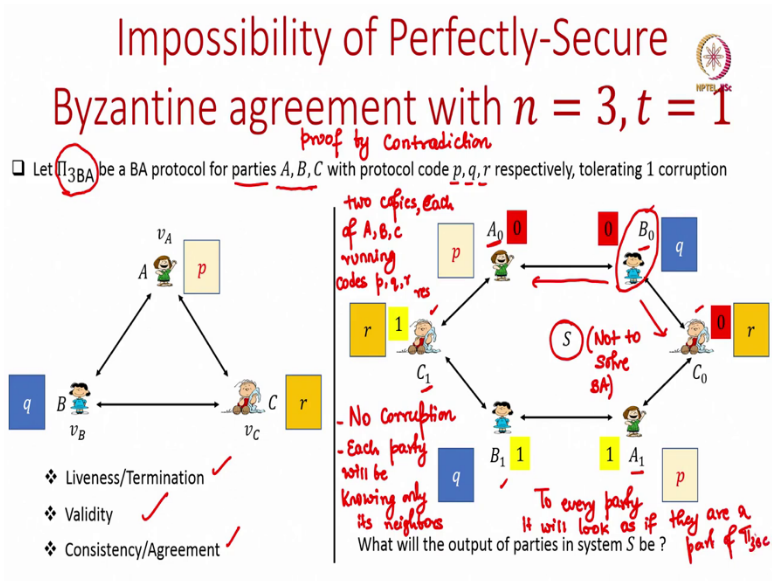
In that case this attack strategy by a fails because if it is forwarding if it is reporting its input as 0. So, it will be sending a signed version of 0 to this party and it will be sending a signed version of 1 to C. And now when B and C talk with each other they will be able to conclude that indeed A is the source of conflict. Because now B will be producing a signed 0 produced by A and C will be producing to be a signed 1 again signed by A.

And assuming signature forgery is difficult, B and C will conclude that indeed A is actually a corrupt party who has unnecessarily reported different inputs to B and C and then B and C can take some default decision. But this argument of using signatures completely breaks down because if you are assuming perfect security then in the perfect security regimes it is possible for an adversary to forge signatures.

In that case consider a scenario where A was honest and B is the source of confusion and B might unnecessarily report to C that A has produced a sign 0 for him. Even though A has produced a signed 1 for B ok. So, using signatures and other cryptographic primitives completely fails in the presence of computationally unbounded adversaries.

So, when we are talking about perfect security we cannot use any such gadget and that is why for perfect security this attack strategy will always work ok. But this intuition where A is the source of confusion and to one honest party it might report a different input and to another honest party it might report another input, this is very intuitive, this is informal we have to now formalize this argument ok.

(Refer Slide Time: 08:13)



Impossibility of Perfectly-Secure Byzantine agreement with $n = 3, t = 1$

proof by Contradiction

- Let Π_{3BA} be a BA protocol for parties A, B, C with protocol code p, q, r respectively, tolerating 1 corruption

- Liveness/Termination ✓
- Validity ✓
- Consistency/Agreement ✓

two copies, Each of A, B, C running codes p, q, r

- No Corruption
- Each party will be Knowing only its neighbors

S (Not to solve BA)

To every party it will look as if they are a part of Π_{3BA}

What will the output of parties in system S be ?

So, we now show formally that there exists no perfectly secure byzantine agreement protocol with three parties tolerating one corruption and the proof will be by contradiction. So, we will assume that suppose there is a three party byzantine agreement protocol I call it Π_{3BA} among three parties say A, B, C and whenever I say parties they need not be physical parties they need not be living entities they could be computers they could be processes, there are some entities who are actually running some protocol codes p, q and r respectively.

And this system has the property that if there are three such parties A, B, C who are running the codes p, q and r respectively and even if one of the parties get compromised and do not run the corresponding code still the termination validity and consistency properties are achieved, that is the guarantee given by this Π_{3BA} protocol. We do not want to go into the exact details of pi 3 BA it could be any round protocol it could be doing any set of actions and so on ok.

It is an abstract protocol. Now assuming this $\Pi_{3\text{BA}}$ protocol we design a new system S ok and this system S might look very non-intuitive at a first glance. So, let us first try to understand what exactly this system S is. So, this system S is not to solve BA. Its an arbitrary system ok. Where we assume that we have two copies each of A, B, C running codes p, q, r respectively.

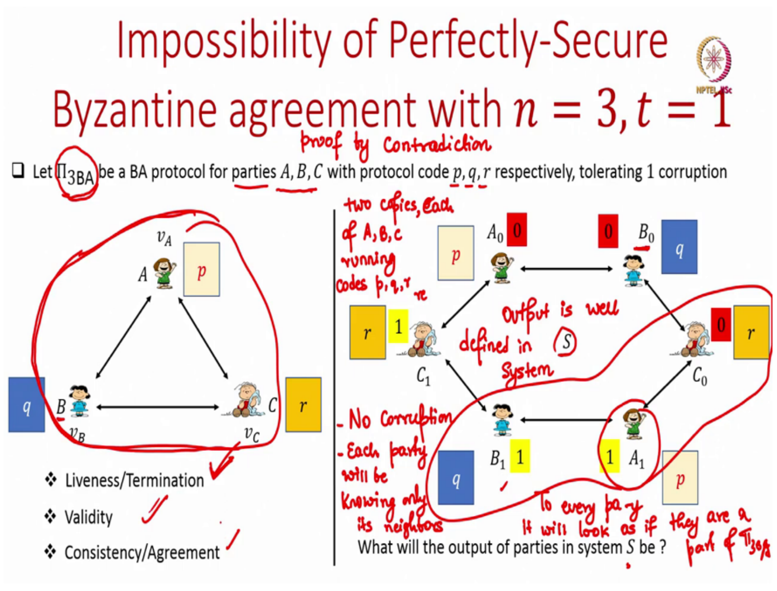
So, we have two copies of A, I am calling them as A0 and A1, one of the copies of A has the input 0 and other copy of A has the input 1. There are two copies of B, one of the copies has the input 0 and another copy has 1. And similarly, there are two copies of C ok. And each of the copies of A, B, C is running the code p, q, r as per the protocol $\Pi_{3\text{BA}}$ assigned. And in this system, there is no corruption.

And each party will be knowing only its neighbors ok. So, for instance A0. It will be knowing that it is talking with C1 and B0. It will not be knowing that it is C1 with input 1 and B0 with input 0, for A0 its as good as it is talking to a party C and to a party B. Similarly from the viewpoint of B0, it will be looking to B0 as if she is talking to the neighbors A and C that is all ok.

So, B naught will be not knowing that there is another copy of C, C 1 there is another copy of A, A 1 running in the system and so on. So, this actually models a scenario where every entity every party has only a partial knowledge about its network ok unlike the byzantine agreement protocols which we have discussed till now where all the parties will be knowing the full network configuration who is connected to whom and so on.

Also to every party in this system S, it will look as if they are a part of $\Pi_{3\text{BA}}$, why so? So for instance let us consider these parts B0, A0 and C0.

(Refer Slide Time: 14:07)



Impossibility of Perfectly-Secure Byzantine agreement with $n = 3, t = 1$

proof by Contradiction

- Let Π_{3BA} be a BA protocol for parties A, B, C with protocol code p, q, r respectively, tolerating 1 corruption

v_A
A
p
B
q
v_B
C
r
v_C
two copies, each of A, B, C running codes p, q, r
A_0
0
B_0
0
C_0
0
C_1
1
B_1
1
A_1
1
Output is well defined in S System
- No Corruption
- Each party will be knowing only its neighbors
To every party it will look as if they are a part of Π_{3BA}

- Liveness/Termination
- Validity
- Consistency/Agreement

What will the output of parties in system S be ?

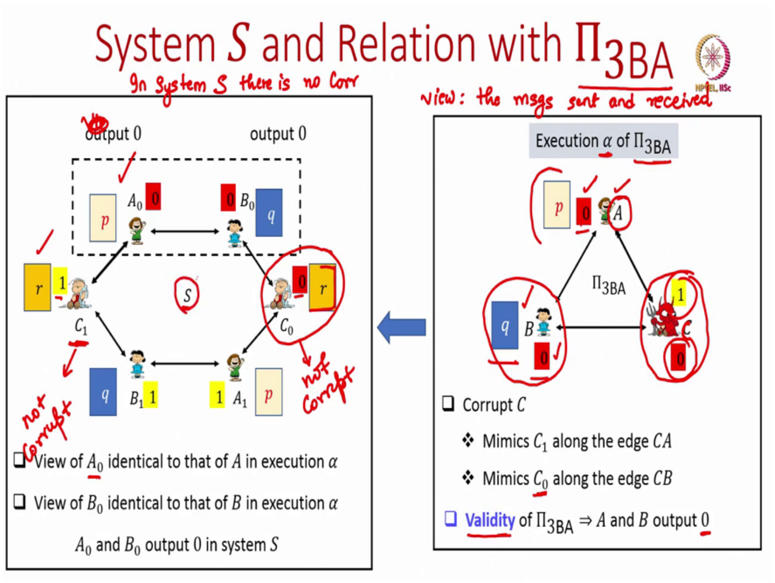
Then as per the protocol Π_{3BA}, B knows that it will be talking to an entity called A and to an entity called C who will be running the codes p and r respectively with some input bits and same is the scenario for B0 here. So, to this B0 it will look as if she is participating in an instance of Π_{3BA} with a copy of A or with an entity A with some input, but running the code p and an entity C who is running the code r.

So, from the viewpoint of B0, it will look like that A0, B0 and C0 compromise a system of 3 parties running the codes p, q, r respectively and participating in Π_{3BA},. In the same way, if I consider for instance the entity A1. Then from the viewpoint of A1, this triplet of parties B1 running the code q and party C0 running the code r, will look like as if there are 3 parties running the codes p, q and r respectively as per the protocol Π_{3BA}, and so on. I stress that the system S is not to solve the byzantine agreement.

But since everyone is running the codes p, q, r as per the Π_{3BA} protocol and since this Π_{3BA} protocol has the liveness and the termination guarantee; that means, after some specific number of rounds of communication A, B, C will have output. So, in the system S as well each of the six entities will get an output ok, as per the time specified in the protocol Π_{3BA}; that means, the output is well defined in system S.

Now, even though the output is well defined in system S, what we are going to show here is that there exists an adversary strategy in the protocol Π_{3BA} which will actually cause the parties in this system S to output contradictory values, which will be a contradiction to the fact that output of the system value S is well defined. So, basically what we actually are going to show is that output of the system is not well defined even though it should be well defined because of the liveness validity and consistency properties of Π_{3BA} ok.

(Refer Slide Time: 17:12)



System S and Relation with Π_{3BA}

In System S there is no Corr

view: the msgs sent and received

- View of A_0 identical to that of A in execution α
- View of B_0 identical to that of B in execution α

A_0 and B_0 output 0 in system S

Execution α of Π_{3BA}

- Corrupt C
  - Mimics C_1 along the edge CA
  - Mimics C_0 along the edge CB
- Validity of Π_{3BA} ⇒ A and B output 0

And I stress that in system S there is no corruption ok. Now what we are going to do is we are going to relate system S with the properties that can be concluded from invocations of the protocol Π_{3BA}. Now consider an execution alpha of Π_{3BA} where the party A and party B are honest and party C is corrupt, input of A is 0 input of B is 0. And now what this party C does is the following.

The party C, it mimics the party C1 as per the system S along this edge CA. When I say along the edge CA means to whenever it is communicating to the party A; that means, in system S, C1 is not corrupt, I stress C1 is not corrupt because in the system S, I assume that no party is corrupt. So, C1 with input 1 would have sent some messages by following the code r to A0.

And it will be continuously communicating to A0 as per the protocol code r, assuming its input was 1. So, when I say mimic C1, that is precisely what this corrupt C is doing during the execution alpha. So, for instance if the first message sent by C1 to A0 was hey my input is 1, then in execution alpha, C is basically talking to A and saying my input is 1.

Then suppose based on whatever message A0 would have sent to C1, then C 1 would have decided to send a set of new messages to A0, those will be the messages which C will be communicating to A in the execution alpha during the next round and so on ok. While in the execution alpha towards the party B, this corrupt C behaves as if he is going to mimic C0 as in this system S.

So, in this system S, this C0 was not corrupt, its input was 0 and it was running the protocol code r and based on its input being 0 and protocol code r it would have exchanged some messages with B0. Those are the messages which it is precisely exchanging with B in this execution alpha ok. That is what is the strategy of this corrupt C party ok.

Now, we know that this Π_{3BA} protocol has the validity guarantee. And validity guarantee demands that if honest parties during the execution have the same input then that should be the final output. So, who are the honest parties in this execution alpha? Its A and B. Whether they have the same input? yes. So, I do not care what exactly is the input of the corrupt party, to one honest party it is participating with its input being 1, to another honest party it is participating with input 0. The validity of Π_{3BA}; that means, the protocol code p and q in the execution alpha where the inputs of A and B would have been 0 and 0, should have ensured that A and B output 0, irrespective of what messages C is communicating with A and B. Now if we consider the view of the party A0 in this system S, it is identical to that of A. Now what does view mean here?

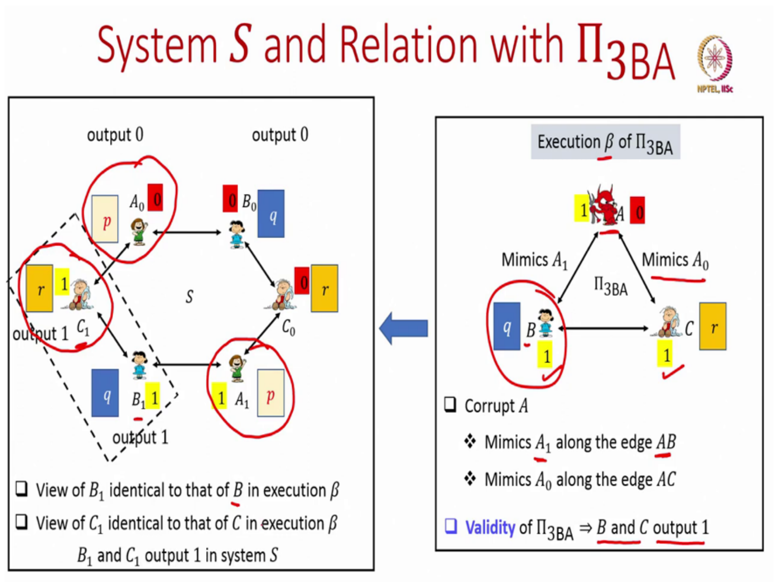
So, view means the messages sent and received by an entity. So, whatever A0 is seeing in the system S, that is identical to whatever A would have witnessed in the execution alpha. Because in the execution alpha, A would have talked with C1; that means, the messages it would have received from C will be identical to what C1 would have communicated to A0. And the messages that A0 would have received from B0 will be identical to what the entity A would have received from the entity B in the execution alpha.

So, that is why view wise the view of A0 will be identical as it is in alpha. And due to the same argument the view of the party B0 in the system S will be identical to the view of the entity B as per the execution alpha. Because in execution alpha the entity B would have talked with entity A whose input is 0 running the code p, that is precisely A0. And entity B would have talked with a corrupt C who is mimicking C0 ok.

With its input being 0 that is and running the code r. That is precisely is the party C0. So, to B0, it will look like as if it has participated in the execution alpha. Now based on the views A0 and B0 have in the system S, they should output whatever the entities A and B should have output in the execution alpha ok. So, based on the views of A and B, A and B would have output 0 in the execution alpha and since the view of A0 is same as that of A and since the view of B0 is same as that of B in the execution alpha, A0 and B0 would output 0 in the system S.

That is what we can conclude regarding the output of the entities A0 and B0 in the system S ok.

(Refer Slide Time: 24:24)



System S and Relation with Π_{3BA}

- View of B_1 identical to that of B in execution β
- View of C_1 identical to that of C in execution β

B_1 and C_1 output 1 in system S

- Corrupt A
  - Mimics A_1 along the edge AB
  - Mimics A_0 along the edge AC
- Validity of Π_{3BA} $\Rightarrow$ B and C output 1

Now let us consider another execution of the protocol pi 3 BA where now the entity A is corrupt and B and C are the honest entities and A mimics the entity A0 towards C whose input is 1 ok. So, basically this corrupt A is trying to mimic A1 along the edge A B ok. So, A1 let us see where it is A1. So, this is A1.

So, whatever messages A1 with input 1 and protocol code p would have communicated with entity B1 whose input would have been 1 and running the protocol code q, same messages the corrupt A will be communicating with the party B in the execution beta. Whereas, whatever messages A0 with input 0 would have communicated to C1 in the system S, those are the messages which A will be now communicating with the entity C in the execution beta ok.

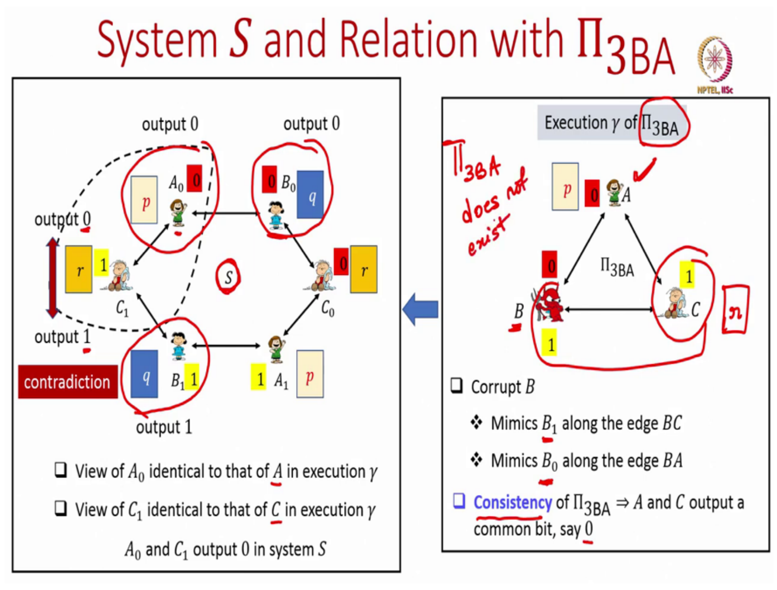
Now, again using the similar argument as we have done for the previous configuration, in this configuration B and C are the honest entities during the execution beta and they have the same inputs. So, it does not matter what the corrupt A communicates with B and C, the protocol code q and r should guarantee that based on the views that B and C have, they should output 1 in the system in the execution beta.

Now, let us compare what is the view of B1 and C1. The view of B1 will be identical to that of B as per the execution beta ok. Because B1 will be receiving the same set of messages from A1 as it would have received from the corrupt A in the execution beta. And it would have received the same messages from C1 as it would have received from the honest C during the execution beta.

So, its view will be identical to whatever is the view of the B. And in the same way if I consider the party C1 here in the system S, its view will be whatever is the view of the party C in the execution beta. Because C1 will be communicating to A0, that is mimicked by this corrupt A and C1 will be talking to B1, that will be anyhow are the messages which B going to send to C.

Now, based on the views that B1 and C1 have in the system S, which actually we have argued to be same as in the execution beta, B1 and C1 are going to output 1 because that is what B and C are doing in the execution of Π_{3BA}, namely the execution beta ok. So, now we have four conclusions; so, we have concluded that A0 should output 0, B0 should output 0 in the system S, C1 should output 1 and B1 should output 1 in the system S.

(Refer Slide Time: 28:12)



System S and Relation with Π_{3BA}

- View of A_0 identical to that of A in execution γ
- View of C_1 identical to that of C in execution γ

A_0 and C_1 output 0 in system S

Execution γ of Π_{3BA}

- Corrupt B
  - Mimics B_1 along the edge BC
  - Mimics B_0 along the edge BA
- Consistency of $\Pi_{3BA} \Rightarrow A$ and C output a common bit, say 0

Now, let us see another execution gamma of Π_{3BA} where B is corrupt ok. And A and C are honest and this corrupt B towards A, it mimics B1 ok. So, this corrupt B towards A it mimics B0. So, whatever the honest B in the system S would have exchanged whatever messages would it would have communicated to A0, the same messages this corrupt B exchanges with A in the execution gamma.

And whatever messages B1 would have communicated with to C1 in the system S, the same messages it sends to the honest C in the execution gamma ok. So, again now if you compare, if we consider the properties of Π_{3BA}, in this case in this execution who are the honest parties? A and C are the honest parties. Do they have the same input? No. They have different inputs.

So, we cannot trigger the validity property of Π_{3BA},, but the consistency property of pi 3 BA guarantees that A and C should have a common output bit say 0. It could be 1 as well, but that is determined by the protocol. So, what it means is that in execution gamma where A and C are honest with inputs 0 and 1 respectively and where B is behaving maliciously, towards one honest party it is mimicking whatever communication B0 would have done towards A0. And towards C it is mimicking the communication whatever B1 would have done towards C1, cannot cause A and C to output different bits right, even if the corrupt B does like that, A and C should output the same thing. Now let us see how it implies a contradictory behavior in the system S. If we see the parties A0 and C1 in the system S, their views will be now identical to that of A and C as per the execution gamma.

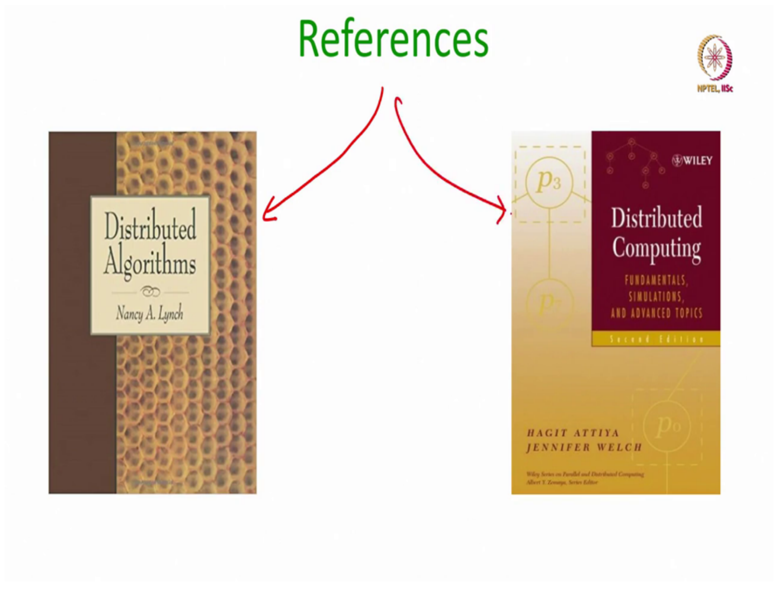
So, for instance let us consider A0, A0's input is 0, running the code B, that is what A is doing in the execution gamma. A0 is talking with C1, running the code r with input 1, that is what A is doing even in the execution gamma. Now the other neighbor of A0 is B0, who is running the code q with input 0 that is what the corrupt B is doing towards A in the execution gamma.

And what about C1? C1 is talking with B1, whose input is 1 running the code q. That is what the corrupt B is doing towards dishonest C. So, whatever A and C would have output in the execution gamma that is what C1 and B1 should do. So, what exactly A and C outputs in the execution gamma? They output 0. So, that is why C1 and B1 should also output 0 ok.

So, C1 and A0 should output 0. But now you see we get a contradiction because we have already concluded that C1 should output 1 earlier. And now we are concluding that C1 should output 0 in the system S and that is a contradiction. Because now it shows that the system S is not a well defined system. Because every one in the system S is honest (remember in the system S no party is corrupt they are exchanging messages based on the protocol code and whatever are their respective inputs). So, if there is no corruption then the output should be well defined because everyone is following code of Π_{3BA} only, but the well defined behavior is not there because we are getting a conclusion that C1 should output 0 after time t whatever is the time defined by the liveness property.

And at the same time we are concluding that C1 should output 1 after the time t. How can that be possible in a well defined system? That is coming only because there must be some problem with the properties of pi 3BA; that means, pi 3BA does not achieve all the claimed properties and that shows a contradiction that pi 3 BA does not exist. Because if pi 3 BA exist then this violation of the well defined output property for S should not occur at the first place ok.

(Refer Slide Time: 34:16)



References

Distributed Algorithms
Nancy A. Lynch

Distributed Computing
Fundamentals, Simulations, and Advanced Topics
Second Edition
Hagit Attiya
Jennifer Welch

So, that shows that pi 3BA does not exist ok. So, the reference used to derive at the impossibility result for n equal to 3 and t equal to 1 is explained in a detailed fashion in this textbook.

Thank you.