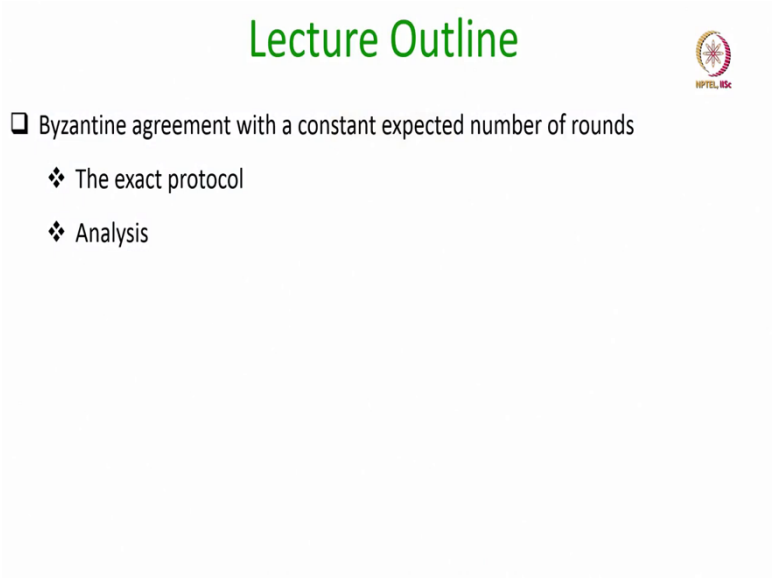
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

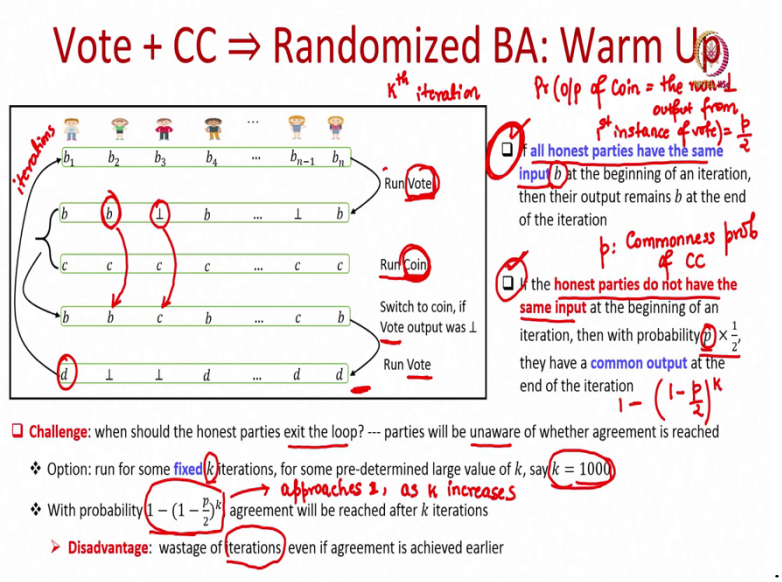
Lecture - 14
Randomized Protocol for Byzantine Agreement: Part III

(Refer Slide Time: 00:25)



Hello everyone, welcome to this lecture. So, in this lecture we will do the exact analysis for the constant expected round Byzantine Agreement protocol.

(Refer Slide Time: 00:36)



So, recall that this was the framework, where we combine the vote and the common coin primitive, to get a byzantine agreement protocol. So, the framework basically involves several iterations. In every iteration the parties do the following: they first run an instance of the vote protocol, to check whether all the honest parties have the same input. And depending upon that, the parties output either a bit or ⊥. Independent of whatever output they obtain during the instance of the vote, they run the coin protocol, and now they again run an instance of the vote protocol. But before doing that, they would like to decide their inputs for this second instance of the vote protocol by comparing the output from the previous instance of the vote protocol. So, if the output from the previous instance of the vote protocol for a party was ⊥, then it changes it input to the output of the coin primitive. Whereas, if the output from the previous instance of the vote was a bit, then the party sticks to that output during the next invocation of the vote. And now, whatever parties obtained from the second instance of the vote, that is treated as the input for the next iteration. And then again the parties run the same sequence of actions ok.

So, in our earlier lecture we have proved 2 properties regarding this framework, the first property was that if all the honest parties have the same input at the beginning of any iteration, say, the input was b, then their output remains b even at the end of that iteration. And this simply comes from the property of the vote protocol. So, say for the Kth iteration, all the honest parties have the same input at the beginning. Then the first instance of the vote, will give the output b for all honest parties and they will never consider what the coin outputs

for them. And again they participate in the second instance of the vote with the same input and obtain the same output. So, basically this property says that, if we reach an iteration where all the honest parties start that iteration with the same input, then agreement is achieved.

And it will never get disturbed; it will keep on outputting that same common bit. And the second property that we proved was that if in any iteration the honest parties have a mixed bag of inputs at the beginning, then with probability at least p over 2 they will have a common output at the end of the iteration and here p is the commonness probability of your coin primitive; that means, with probability at least p all the honest parties have a random and a common bit during the instance of the coin primitive. Again how we proved it? Well, if all the honest parties have a mixed bag of inputs, then we do not get any guarantee whatsoever regarding their outputs at the end of the first invocation of vote, some honest parties may output a bit b; while others may output the value bot. And now, before they start the second instance of the vote protocol, some honest parties might be switching their outputs from bot to the output of coin, while others might be sticking to whatever output they have obtained from the first instance of vote. So, what is the probability that the parties who are switching their inputs from bot to a non-bot value, that non-bot value which is actually the output of coin turns out to be the output of vote? That probability is p over 2 because the instance of coin is invoked only after the vote gets over. And what is the probability that the output of coin is equal to the non-bot output from 1st instance of vote in an iteration, its p over 2, because with probability p the output of coin will be random and common for all honest parties, and given that it is random and common for all honest parties; what is the probability that output is actually the same output b which is obtained from the first invocation of vote, its 1 over 2. So, overall probability turns out to be p over 2 ok.

So, the basic idea behind this framework was that, remember for the ba we need to satisfy the validity and consistency. Validity means, if the parties start the protocol with the same input, then this first property guarantees that they have agreement on that input throughout the protocol; that never gets changed during any of the iterations. Whereas, to get the consistency of the agreement property, for the case when the honest party start with a mixed bag of input, we basically depend or take the help of this coin primitive to guarantee, that at the end of every iteration, the parties reach agreement. And once they reach agreement at the end of the iteration from next iteration onward, this first property will be triggered and agreement will

be maintained. That was basically the idea, that is basically the idea behind the framework. But now the challenge behind this framework is that, when should the honest parties exit the loop? Ok. What is the condition?

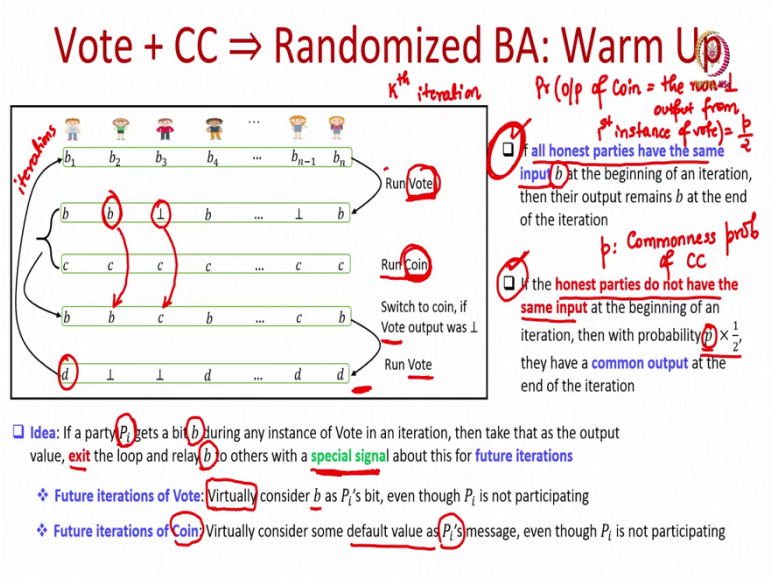
Because parties will be unaware, whether agreement is reached or not. So, for instance if I am a party, and if at the end of an iteration I obtain a non-bot value, I am not sure whether other honest parties also obtained a non-bot output or not. They could output bot. Because I do not know, who are the honest parties in the system. And if I leave the protocol, right, if I simply leave the protocol and do not participate in the subsequent invocation subsequent iterations, then the other honest parties who might have obtained bot and who goes to the next iteration, they might be waiting forever for the vote protocol or the coin protocol to get over ok. Because my participation is critical for the subsequent invocations of the vote and the coin. So, one option to deal with this challenge is that irrespective of when exactly agreement is achieved, run this framework for k number of iterations, where k is some parameter and we can set k to be a large value say 1000.

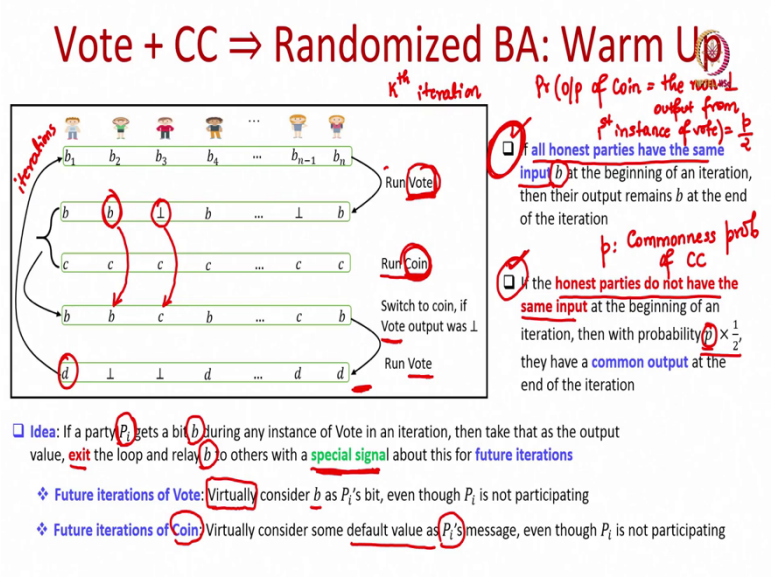
And with probability $(1 - p/2)^k$, agreement will be reached after k iterations. This is because, if anyhow parties have reached an iteration where they have the same input to begin with, agreement is reached. And what is the probability that in none of the k iterations, the parties have the common output at the end of the iteration, its $1 - p/2$. And since we are doing it for k iteration, it is $(1 - p/2)^k$.

So that means, the probability that at least in one of the k iterations the agreement is achieved is this much. And this turns out to be a very small quantity if we keep on increasing the value of k. So, this quantity approaches 0 sorry, it approaches 1 sorry. This quantity will be almost 1, if I set the k to be large. So, it approaches 1 as k increases; that means, the probability that agreement is not achieved, as I keep on increasing the number of iterations will eventually approach to 0.

But this approach has a huge disadvantage because, it results in a huge wastage of iterations. Because it could be possible that the party started the protocol with the same input. In that case, even though the agreement would have been achieved at the end of the first iteration itself, the parties would be unnecessarily running the protocol for 1000 iterations. So, now, the challenge is how exactly we decide, when should the honest parties exit the loop.

(Refer Slide Time: 11:16)

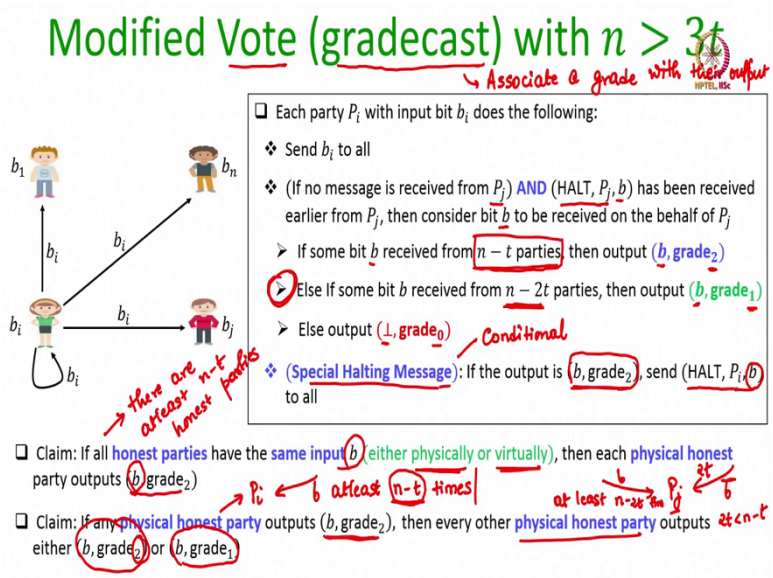




So, we will not use this idea of running this framework for some fixed number of iterations, irrespective of when the agreement is achieved. Rather we will do the following ok. If any party Pi obtains a bit during any instance of vote in an iteration, so remember, in each iteration there are 2 invocations of vote; 1 at the beginning and 1 at the end. So, if a bit is obtained in any of these 2 iterations, then it simply takes it to be the overall output for the BA and it decides to exit the loop. But with a special halting signal to everyone, ok, that I have obtained a bit from this instance of vote, and I am not going to participate in the future iterations, but you can imagine that I am participating virtually with my bit being b for future iterations of vote. This will be a signal for every other party that ok, Pi is participating

virtually; that means, there will not be any message coming from Pi. But on the behalf of Pi everyone would take the bit b as the message coming from Pi in the future invocations of vote. Of course, if Pi is honest, it will do it only when it indeed gets a bit as the output during the instance of a vote, but if Pi is a corrupt party, it may unnecessarily send an halting instruction, but that is fine. We need to ensure that the participation from the honest parties is continued, either virtually or physically in all the instance of vote, till all the honest parties terminate the protocol right. And if this halting signal has been sent by any party Pi, then that is also an indication, that even in the future invocations of the coin primitive, Pi will not be sending any message. So, take some default message as the message on the behalf of Pi, that is the idea here. Now let us see how we can materialize this idea and how exactly we use it to determine the termination condition.

(Refer Slide Time: 13:54)



So, we will first do a modification to our instantiation of the vote protocol, which we had seen in the earlier lecture, where we will incorporate this idea of sending the special halting message. So, this modified vote protocol in the literature is often called as the gradecast primitive because, here every party will associate a grade, they will associate a grade with their output and that grade will be basically a measure of confidence level which they have regarding the output that they are going to obtain ok.

Let us see how the grades are assigned. So, the communication pattern remains the same, it is still a 1 round protocol. So, every party will send its bit to everyone, including itself;

however, it could be possible that there is some party Pj from whom no message is coming. He is supposed to send its input, but no input is coming from that party Pj, but on behalf of that party Pj we have already received some halting instruction with a bit P in some previous invocation, previous instance of the vote protocol, ok. If that is the case, then we assume that party Pj is sending the bit b, even during this instance of the modified vote or the gradecast protocol ok. So that means, if Pj would have been an honest party and it has already terminated it has already gone out of that loop where we are designing the ba protocol, then in all the future invocations of the vote protocol where Pjs participation is required physically, we can imagine that Pj is still present there, but virtually with its message being b. Because it would have sent a halting instruction during the instance of vote, where it would have exited the loop.

Now the decision rule. So, every party Pi will decide its output as follows, it will check whether any bit b is received at least n minus t times, either physically or virtually, when I say physically; that means, indeed that party has sent a message b, with or without halting instruction. And possibly including from other parties who have halted in previous invocations of the gradecast. So, if there are total n minus t copies of the bit b, then output that bit with the highest grade, full confidence. Whereas, if some bit is received physically or virtually from at least n minus 2 t parties, then output that bit, but with a lower grade. So, you see the confidence level is now low, confidence regarding whether other parties will be outputting b or not.

If I output b with grade 2; that means, I am giving a very high confidence that every other honest party will also output either this bit b or the value bot, but no other bit. But if I am outputting the bit b with grade 1, then I am not sure what is going to be the case. But if there is no bit b which is received n minus t times or n minus 2 t times either physically or virtually, then I output the value bot and assign the lowest grade ok.

Apart from this there is a special halting message which is conditional and this message is sent by Pi only when it is outputting a bit b with the highest grade. If it is outputting a bit b with the highest grade, then it sends a conditional message, it sends an instruction that ok I am halting, my identity is Pi, with the bit P and as a result of this for all the future instances of the gradecast or the modified vote protocol in the ba protocol, you can imagine that I am

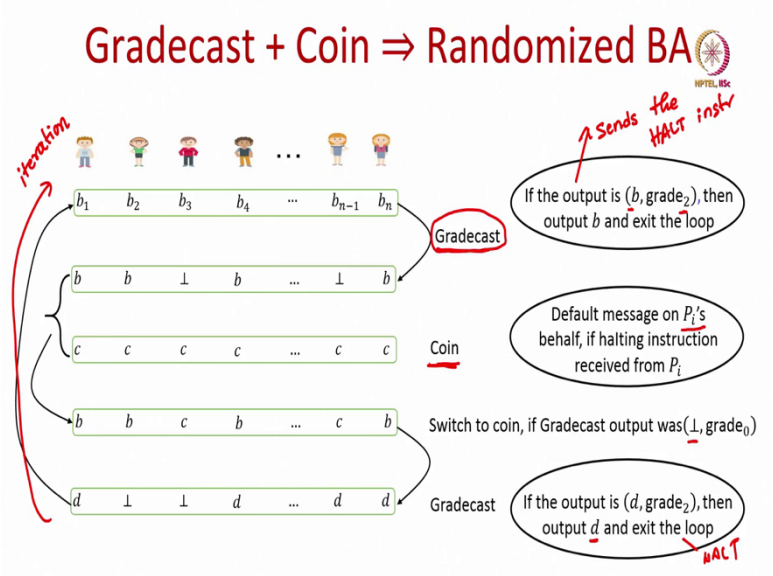
setting, I am sending the bit b, my participation will not be there physically ok. So, I stress that this is a special halting message this will not be sent every time by Pi. It will be sent only when Pi is going to obtain an output with the highest grade.

Now the claims regarding this gradecast protocol. If all the honest parties have the same input bit either physically or virtually, then each party who is physically still present in the system, who is still running that sequence of iterations, will be outputting the bit b with grade 2; the highest grade. And this again simply comes from the fact that if all the honest parties have the same bit, then since there are at least n minus t honest parties and since we are assuming they have the same bit irrespective of whether they are present physically or virtually, when I say virtually bit b; that means, they have sent the halting instruction with b as the message. So that means, at this step every party would have received n minus t copies of the bit b and hence they will output the bit b with the highest grade.

And the second claim is that, if any physical party outputs any bit b with the highest grade, then every other physical honest party will output either the same bit with the highest grade, or the same bit with grade 1. It cannot be the bit b prime with grade 2 or grade 1 ok or it cannot be the value bot. And this simply comes from the fact that if any physical party say Pi, it has output b with grade 2; that means, it has received b at least n minus t times from n minus t different parties, among these n minus t different parties n minus 2 t are honest. So, if I consider another physical party Pj, it is guaranteed to receive b at least n minus 2 t times. Because it could be possible that among these n minus t parties who have sent b to Pi, t are corrupt and they decide not to send b to Pj. They may decide either not to send anything or to send b prime. But there are still n minus 2t honest parties in this collection, who will send the same bit b as their input to the party Pj. So that means, this else condition is definitely going to be true for Pj and it cannot be possible that Pj receives b prime from another set of n minus t parties. Because there could be at most 2 t parties from whom it could get b prime, namely the corrupt parties who would have sent Pi the bit b might decide to send b prime to Pj, along with t parties who have sent some value to Pi outside the set of this n minus t parties. So, overall 2t copies of b prime could come to Pj, but 2t is strictly less than n minus t, so that means, this first if statement is not going to be true for Pj. So, second statement is definitely

going to be true for Pj. So, it is bound to get at least the output b with great one, but it could be possible that Pj receives n minus t copies of b in which case it would output b with grade 2 ok. So, these are the two claims right; regarding the gradecast. If all the honest parties have the same input then every honest party who is present, who is still running the code physically will output b with highest grade. And if someone outputs a bit with highest grade, then everyone else will output the same bit either with grade 2 or grade 1.

(Refer Slide Time: 23:46)



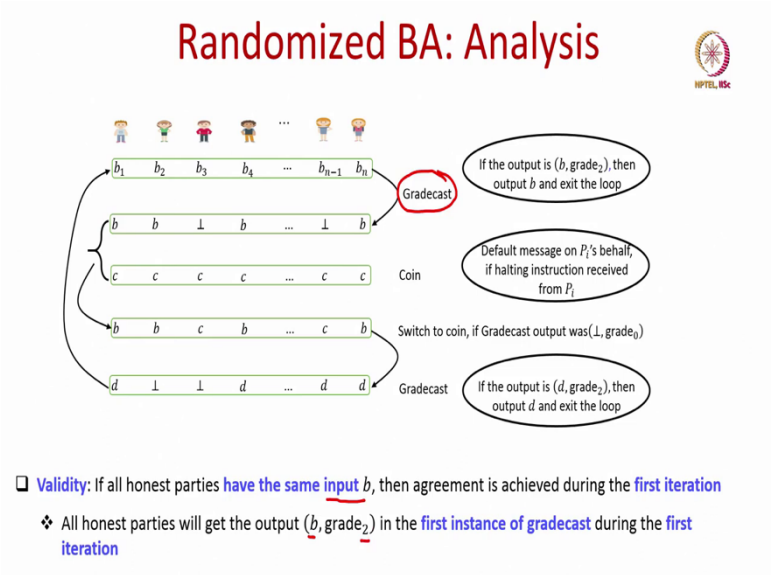
Now, let us see how what happens if we use this gradecast protocol in the framework instead of the vote protocol ok. So, gradecast is still the vote protocol except that we now ask the party to terminate the protocol and exit the loop with a special halting instruction; for all the future instances of vote and coin in the framework.

So, in every iteration the parties do the following, they first run an invocation of gradecast to check whether all the honest parties have the same input or not. And during this instance of gradecast, if the output for any party is a bit with the highest grade, then that party decides to exit the loop. And it sends the halting instruction, as part of the gradecast. Now, irrespective of what output the parties obtained from the invocation of gradecast, the parties run an instance of coin. And here, if some message is expected from Pi, but Pi has already sent the halting instruction, then during the instance of coin, we can imagine that Pi is sending some default message. And then, the parties decide how to participate in the second instance of gradecast, they check, if any party has obtained the output bot with grade 0 during the first

invocation of gradecast. If that is the case, then they switch their input to whatever output they have obtained from the coin primitive. Otherwise, they stick to the output that they have obtained from the gradecast. And then they run the gradecast primitive. And again here there is a possibility that, an output for an honest party is a bit with highest grade. If that is the case, then it outputs that bit as the output for the byzantine agreement and exits the loop with the halting instruction. And the parties go to the next iteration.

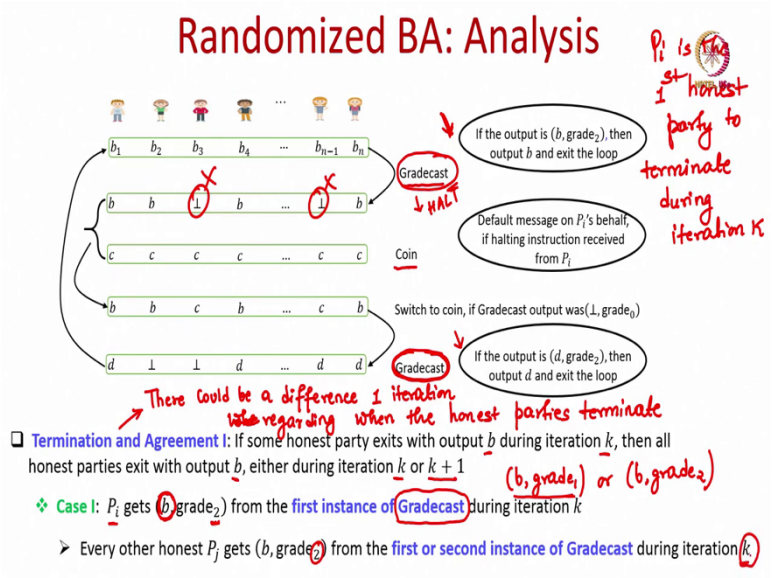
Whoever has not halted, whoever has not exited the loop, they go to the next iteration. But the parties who have halted, who have decided to exit the loop, they simply exit with whatever output they have obtained, with the highest grade from the corresponding instance of gradecast.

(Refer Slide Time: 26:31)



Now, let us see the analysis of this protocol. So, the validity property is very simple to prove, we can very easily prove that if all the honest parties have the same input bit b, then the agreement is achieved during the first iteration itself. This is because, from the first invocation of gradecast, everyone will obtain the output b with the highest grade and they will send the halting instruction and then exit the loop that is all ok.

(Refer Slide Time: 27:12)



Now, arguing the consistency is slightly tricky. So, we will prove a sequence of statements. So, let us prove the first property, first helping lemma regarding the termination and agreement.

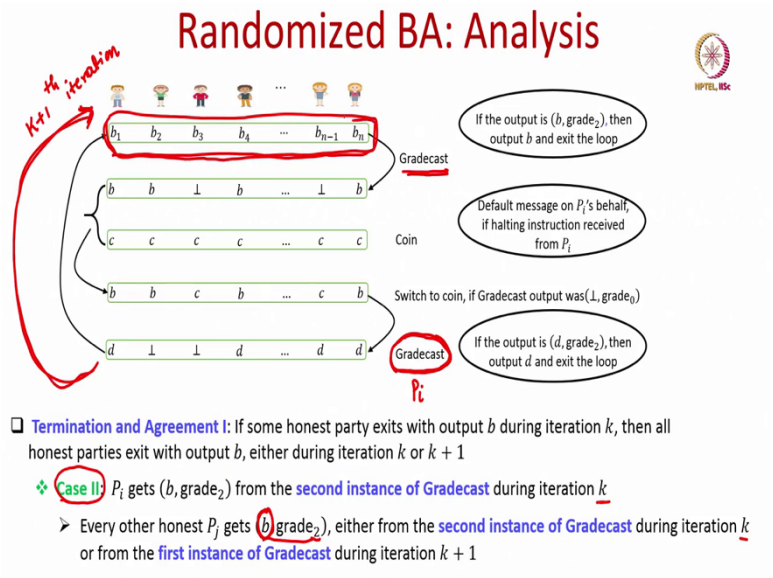
So, the first property is that, if some honest party exits the loop with an output b during the iteration k, then all honest parties will also exit the loop with the same output, either during the same iteration or definitely by the end of the next iteration. So, before we go into the proof of this property, this property basically states that, there could be a difference of 1 iteration, regarding when the honest parties terminate. It is not always necessary that is the case. But it could be possible that, one honest party terminates in say the iteration number 20, and other honest parties terminate in iteration number 21, that could be possible. And whenever an honest party is terminating, it will not be knowing, it will not be sure that everyone else also will be terminating along with him in the same iteration. What it will know? Definitely by the end of next iteration everyone else will also terminate ok.

So, let us prove this property ok. So, when exactly a party terminates, when it obtains the highest grade output during an instance of gradecast. So, suppose there is a party Pi. So, suppose Pi is the first honest party, to terminate during iteration k. Now there are 2 possibilities, where it would have terminated during iteration k, either at the end of the first invocation of gradecast or at the end of the second invocation of gradecast.

If it has terminated at the end of the first invocation of the gradecast; that means, it has obtained the output b with the highest grade, during that instance of gradecast and remember, the property of gradecast guarantees that, if Pi has obtained b with the grade 2 as the output, then every honest party would have obtained b either with grade 1 or b with grade 2, during the same invocation of gradecast.

Well, if everyone else also has obtained this output b with the highest grade 2; that means, everyone else also would have terminated in that iteration itself. But it could be possible that Pi has obtained b with grade 2, but others have obtained b with grade 1 from this first invocation of gradecast. Now when they go to the second invocation of the gradecast in the same iteration, the inputs of all the parties including Pi will be b only. Because Pi would have sent a halting message as part of this gradecast; that means, on behalf of Pi everyone will consider the input b during the second invocation of gradecast, and anyhow all other parties who are still participating physically during the second invocation of gradecast has the input b; they will never switch their input to the output of the coin. Because, they will not obtain bot from the first invocation of gradecast. And that will ensure that from the second invocation of gradecast, every party who is still running the code physically by actually sending the message, will obtain the output b with highest grade. So, irrespective of when exactly it happens, it will be either during the first or the second invocation of gradecast, but during the same iteration, when everyone else will output the value b with grade 2. And that will imply that they also exit the loop during the same iteration ok.

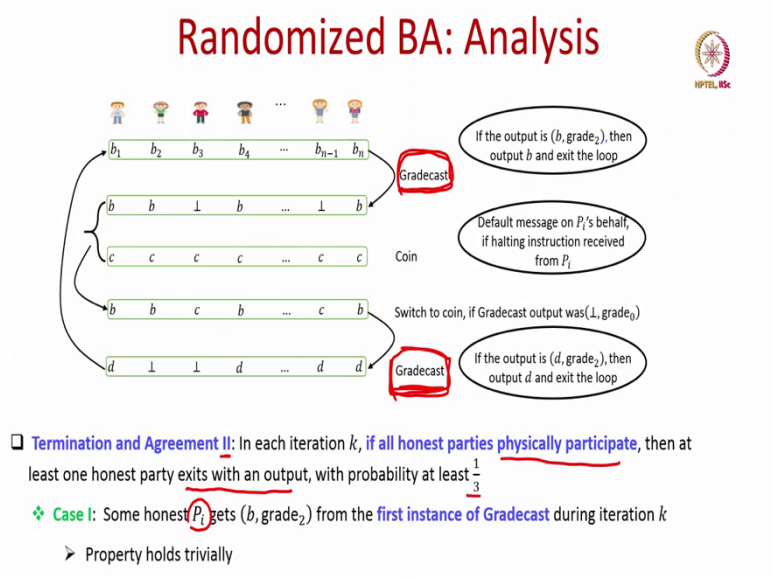
(Refer Slide Time: 32:56)



Now, there is a second possibility, that the party Pi has actually exited the loop after the second invocation of gradecast, during iteration k, right. In this case, what we can conclude is that every other honest party Pj who decides to go to the k plus 1 th iteration, would also obtain the output b with grade 2 right. Well, it could be possible that there are some parties Pi who do not decide to go to the next iteration. Because they also would have obtained the output b, with highest grade from the iteration k only. So, they also would have exited the protocol with the output b during the iteration k itself. But if at all any honest party Pj decides to go to the next iteration, then during the first invocation of gradecast, in the k plus 1 th iteration, they are going to obtain the output b. Because all the honest parties who go to the next iteration, namely the k plus 1 th iteration, they will have their input b; and anyhow the honest parties who have exited the loop during the iteration k would have sent a halting instruction, as part of this last invocation of gradecast in the iteration k. So, for those parties, also the input will be considered as b. So, overall there will be n minus t honest parties, participating physically or virtually with input b during the first invocation of gradecast during iteration number k plus 1, and that will guarantee that whoever are continuing to the k plus 1 th iteration, they terminate at the k plus 1 th iteration ok. So, it is this case number 2, which actually results in a lag of 1 possible iteration between 2 honest parties regarding the time when they are outputting ok.

So, if 1 if Pi decides to terminate because of this second invocation of gradecast, during the iteration k then it is not necessary that every honest party Pj also decides to terminate because of this second invocation of gradecast, during the iteration k. It may be possible that some of them has to continue to the k plus 1 th iteration, but in the k plus 1 th iteration this first invocation of gradecast will cause them to terminate ok right.

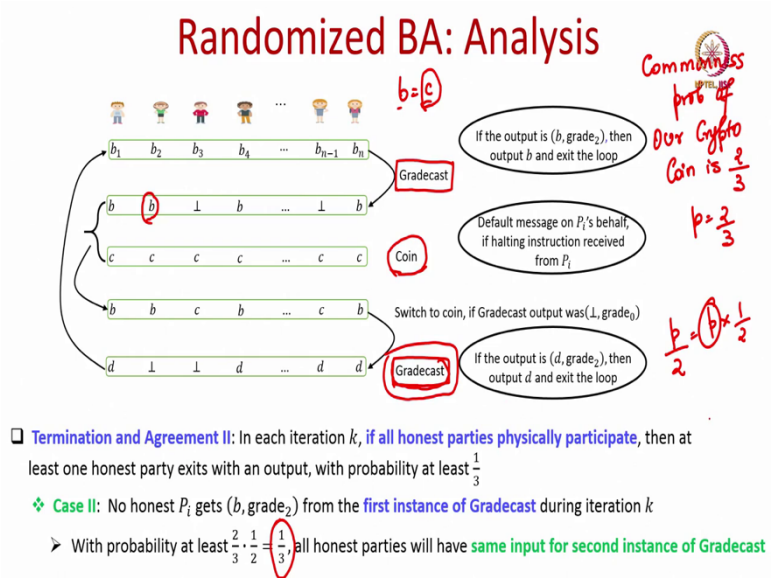
(Refer Slide Time: 35:43)



Now, let us prove the second helping lemma, regarding the termination and agreement. So, this property states that in each iteration, if all the honest parties physically participate; that means, they have not yet terminated, then at least one honest party terminates and exit the loop, with an output either 0 or 1, with probability at least 1 over 3 ok. So, again there are 2 possibilities here.

So, suppose Pi is an honest party, who obtains the highest grade output during the first invocation of the gradecast, during the iteration number k. Then in this case, with probability 1 everyone will terminate, at the end of the same iteration ok. This is because, since Pi has obtained b with grade 2 from the first invocation, it will be guaranteed that either from the same invocation of gradecast everyone else terminate with highest grade output b. Or definitely from the second invocation of the gradecast during the same iteration k everyone else would have terminated. So, there is no probability associated, it is a 100 percent guaranteed event if that is the case.

(Refer Slide Time: 37:22)

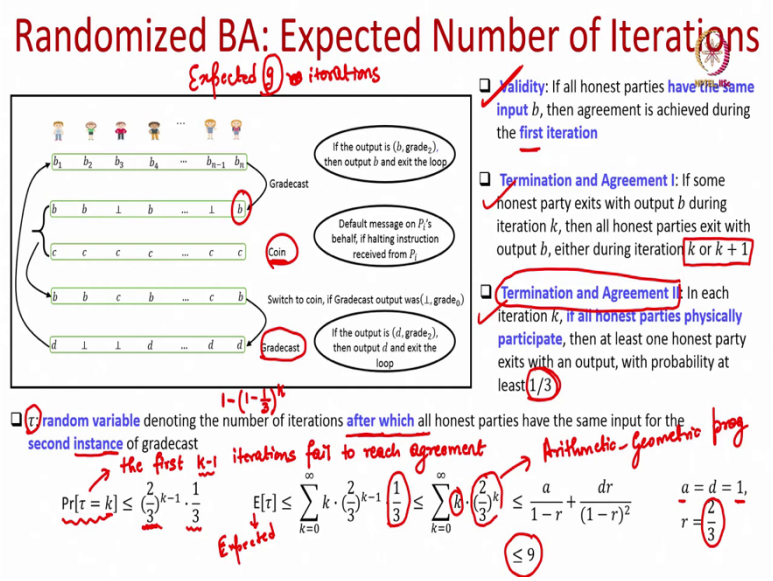


Case number 2 is, no honest party has terminated in iteration k; that means, no one has obtained any bit with the highest grade 2. Then, let us focus on what is the scenario regarding the inputs of the honest parties, during the second invocation of gradecast.

So, recall that the commonness probability, the commonness is probability of our cryptographically secure coin protocol is 2 over 3, namely p was 2 over 3. And we have argued already, that during the second invocation of gradecast, the inputs of all honest parties will be same with probability P over 2, this we have already argued earlier. Namely with probability P, the output of all honest parties from the coin protocol will be common and random. And that random output could be the output that the honest parties obtain from the first invocation of gradecast, which is not a bot output, with probability half right. So, with probability half, b equal to c, and the probability p all honest parties will have the same c, that is why with probability p over 2, we can conclude that all honest parties will definitely have the same input, at the beginning of the second invocation of gradecast, during iteration k. And p over 2, where p is 2 over 3, guarantees that with probability at least 1 over 3 all honest parties have the same input, during the second invocation of gradecast, and if that is the case at least one honest party will obtain the output b with highest grade. In fact, all the honest parties will obtain b with highest grade and they will exit the iteration. So, that is the case number 2.

So, we have proved the validity property.

(Refer Slide Time: 39:35)



And now we have proved the termination and agreement ok. So, again to stress here, there could be a difference of one iteration between the iteration number, when the honest parties terminate. But they terminate with the same output. Now, we want to calculate here the number of iterations; the expected number of iterations after which all the honest parties will terminate ok and for that, we will be basically focus on this termination and agreement 2 property.

Of course, if all the honest parties start the protocol with the same input, due to the validity, they terminate in iteration 1 itself. But it could be possible that the honest parties start the protocol with a mixed bag of inputs, in which case it could be possible that the protocol keeps on getting dragged, iteration after iteration. So, we now want to measure after how many iterations in expectation, honest parties are bound to terminate.

So, for that we introduce a random variable tau and it is a random variable because there will be a probability associated with the values of tau. So, this variable basically denotes the number of iterations, after which all the honest parties will have the same input for the second instance of gradecast, ok. Now, the probability that this random variable tau is equal to k; that means, the first k - 1 iterations failed to reach agreement right.

And its only during the kth iteration, all the honest parties have the same input for the second invocation of the gradecast. Well, if all the honest parties have the same input for the second instance of the gradecast, we know that termination will be guaranteed in iteration number k. Otherwise, the parties have to go to the next iteration again. Now what is the probability that tau is equal to k?

Well, the termination and agreement two property says that, with probability at least 1 over 3, each iteration should have ensured that agreement is achieved; that means, with probability 2 over 3, agreement is achieved in each iteration, and we are basically considering the scenario, when the first k minus 1 iterations have failed to reach agreement; that means, the probability of that is 2 over 3 raise to power k minus 1, times the probability that agreement is achieved during the k th iteration.

Namely it is during the k th iteration, where all the honest parties have the same input at the beginning of the second invocation of gradecast, which can happen with probability 1 over 3. Now, this is the probability of the random variable tau being k, we want now to find the expected value of k. So, the expected value of k as per the formula of expectation turns out to be this.

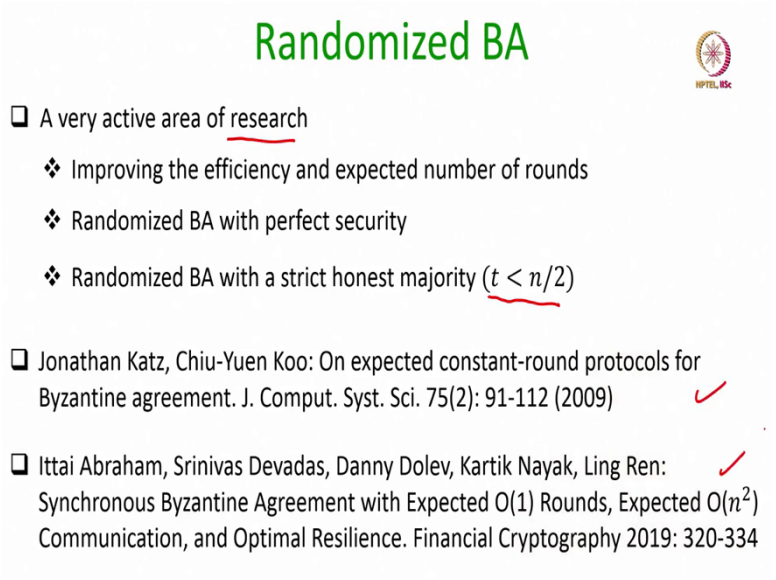
And I can bound this 1 over 3, I can replace this 1 over 3 with 2 over 3, because of this inequality. So, this inequality turns out to be this. And now this is an arithmetic geometric progression, where the summation k can be treated as an arithmetic series, where the first term is 1 and the common difference is 1, whereas this product of 2 over 3 raise to power k and summation over that can be treated as a geometric series, where the common ratio is 2 over 3.

And there is a well-known formula for the summation of this series, which is given by this formula; if I substitute the value of a and d and r, then the expected value of tau turns out to be 9; that means, it requires expected 9 iterations to be more specific, for the parties to terminate. Of course, it could be possible that parties never terminate, even after running for 10,000 or million number of iterations, and that can happen only when all the honest parties start the protocol with a mixed bag of inputs and the output of coin and the output from the first invocation of gradecast, always turn out to be different or the commonness probability is not guarantees. In that case, the second invocation of gradecast, will fail to reach agreement among the parties, in each iteration and the parties will keep on running the protocol forever.

But the probability of that is almost 0. Because we have derived that it requires only expected 9 iterations ok.

And the probability that even after k iterations, the parties do not terminate is 1 minus 1 minus p over 2 where is 1 over 3 raise to power k and this turns out to be 1 as we keep on increasing the value of k ok.

(Refer Slide Time: 45:43)



So, this whole area of randomized byzantine agreement, where we want to design protocols with expected constant number of rounds, is a very active area of research there is a lot of scope of doing research, in terms of improving the efficiency and expected number of rounds.

And the level of security we achieved and to improve the resilience. So, we have designed the protocol, the protocol that I have discussed is with t less than n over 3, but there are also protocols available with t less than n over 2.

(Refer Slide Time: 46:28)

If you are interested to know more about those protocols, you can refer to any of these two references, and this is the reference which I have used in my presentation.

Thank you.