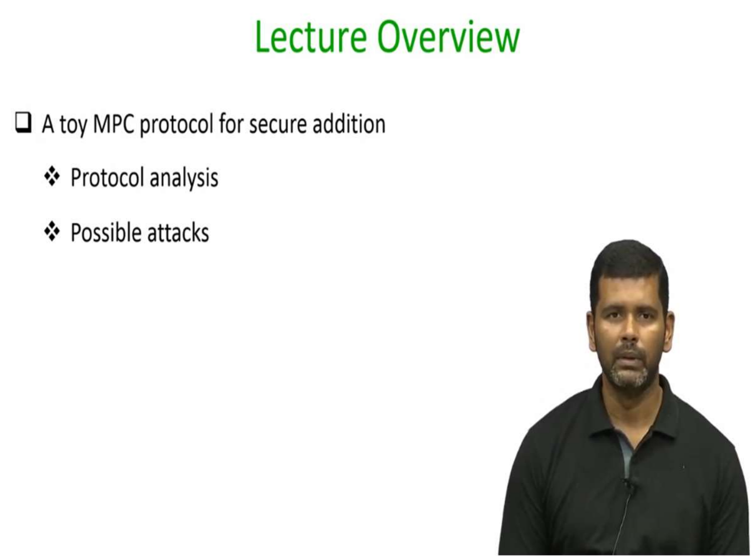
Secure Computation - Part I
Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

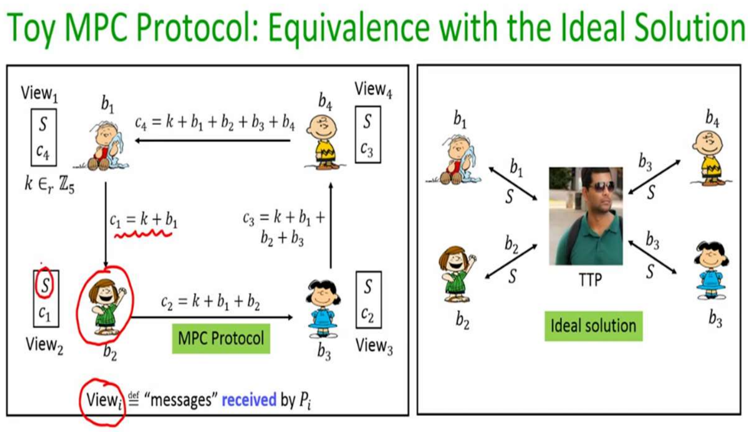
Module - 3
Lecture - 18
A Toy MPC Protocol Contd.

(Refer Slide Time: 00:33)



Hello everyone. Welcome to this lecture. The plan for this lecture is as follows: In this lecture, we will continue our discussion regarding the toy MPC protocol that we had seen for performing secure addition, and we will perform the analysis of the protocol and possible attacks which can be launched on the protocol.

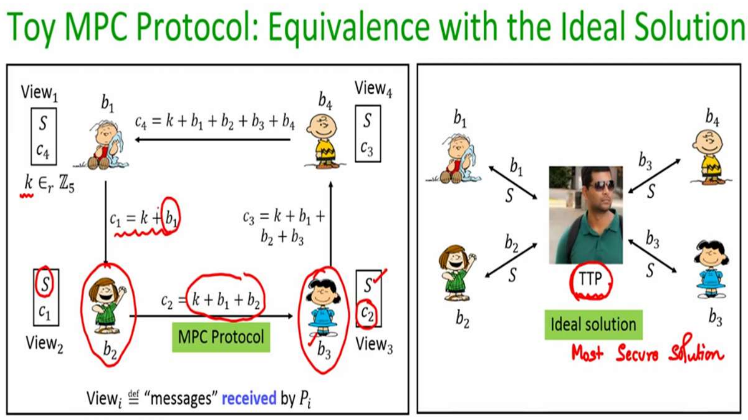
(Refer Slide Time: 00:51)



So, in the last lecture, we had seen that the toy MPC protocol, we had described what we call as the view of the ith party, namely, the view denotes the messages which are received by the party P_i. So, if you see each party in the MPC protocol that we had designed, receives only a single message from its neighbour, namely the encrypted summation of all the inputs of the parties appearing before that party. What does that mean?

So, if I consider the party number P_2, there is only 1 party appearing before it, namely the party P_1. So, the view of party 2 basically consists of by encrypted input b 1; of course, it learns the final sum that is also a part of its view.

(Refer Slide Time: 01:55)



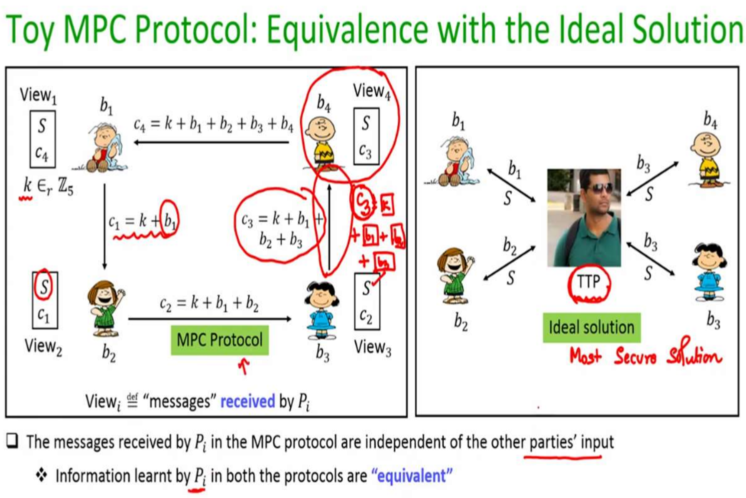
If I consider party number, P_3, then, its view is, of course its own input; and the final sum, because that is what it receives from the party P_1; and the ciphertext c_2, which is the encryption of the summation of the inputs b_1 and b_2 with respect to the key k which has been picked randomly by party P_1. And we want to compare this MPC protocol with what we call as the ideal solution, where we assume that there is some trusted third party.

And the ideal solution ensures the privacy of the inputs of the respective parties; because, in the ideal solution, the parties do not interact with each other at the first place, there is no scope of interaction among the parties. The only interaction that is happening in the ideal solution is between every party and the trusted third party. And the trusted third party is believed or assumed to not disclose the inputs that it is receiving from the respective parties to any other party.

So, the ideal solution is actually the most secure solution that you can think of, for computing this sum function, because it allows every party to just learn its own input and the final output; it does not provide any scope to interact with the other parties and learn anything about the inputs of the other parties. But in the MPC protocol, there is interaction happening among the parties.

So, for instance, if I consider party P_2, there is an interaction happening; it is receiving something related to b_1, namely, the input of the first party, from the first party, in the form of its encryption.

(Refer Slide Time: 03:55)



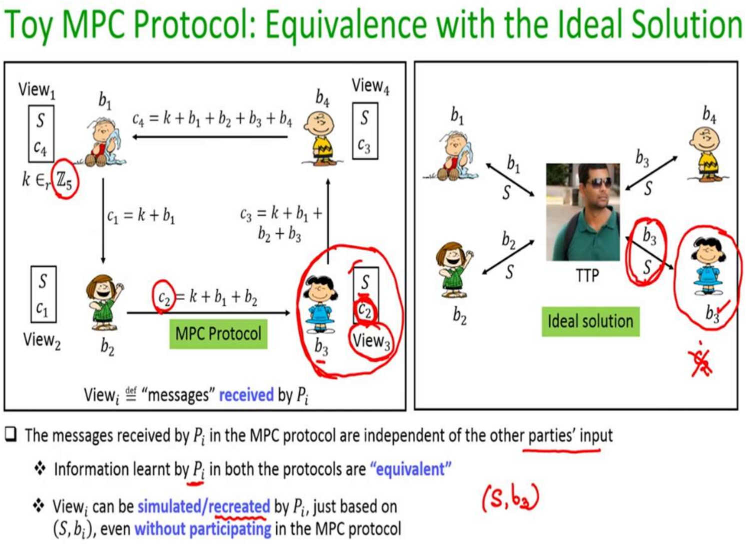
In the same way, if I consider party number say P_4, it is interacting with party P_3, unlike your ideal solution. In the ideal solution, P_4 has no scope of interacting with party P_3, because P_3 directly go and speak to the TTP, P_4 directly goes and talk to the TTP. But in the MPC solution, this party P_4 is receiving this information from the party P_3. But what we had proved in the last lecture is that the messages which are received by every party P_i in the MPC protocol, they are independent of other parties' input.

We had seen this by example, and we had rigorously argued this. So, for instance, if I consider party P_4, even though it is receiving something from P_3, that something is kind of independent of what are the values of b_1, b_2, b_3. That means, we had seen that the c_3 that party P_4 is going to receive, it could be the summation of any candidate k along with any candidate b_1, b_2, b_3.

That means, this ciphertext c_3, even though it is receiving from the party P_3, the probability distribution of c_3 is independent of the actual values of b_1, b_2, b_3; because, for every candidate b_1, b_2, b_3 that P_4 assumes in its mind, there is some corresponding k which sum up and gives the ciphertext c_3, which P_4 would have received in the MPC protocol. That means, now I can say that, in some sense, the information learnt by every party P_i in the MPC protocol is equivalent to the information that P_i would have learnt in the ideal solution.

So, I have written down here equivalent in quote-unquote, because right now, we are not going to formally define what do we mean by equivalent here, but let us try to understand what do I mean by equivalent here, the intuition behind that. Let us consider, say the party number P_3.

(Refer Slide Time: 06:10)



What is its view in the MPC protocol? Its view is the final sum, its own input and this ciphertext c_2. Now, let us compare this view with the view of the same party P_3 if it would have participated in the ideal solution, with its input b_3. So, what would be the view of this party P_3 in the ideal solution? There also it would have its own input bid; it would learn anyhow the final sum, but it will not have this ciphertext c_2 here.

This piece of information is not available in the ideal solution to the party P_3, because there is no ciphertext communicated among the parties in the ideal solution. So, now, there is some piece of information, namely this c_2, which is a part of the view of party P_3 in 1 protocol, but it is not part of the view of the same party in another protocol. So, will this be considered as a breach of privacy? The answer is no; because, even though there is some additional thing which is present in the view of party P_3, it is independent of what exactly are the values of b_1 and b_2. It could be any b_1 and b_2 summed up with any k which would have produced this c_2. So, that means, even though there is some piece of information related to b_1 and b_2 available in the view of party P_3, in the MPC protocol, compared to ideal solution, that an additional piece of information is completely useless as part as the view is concerned.

What I mean by that is that this additional piece of information, namely c_2, which is present as part of View 3, can be recreated by P_3 itself, without even participating in the MPC protocol. What does that mean, without even participating in the MPC protocol? By that I mean that, even if P_3

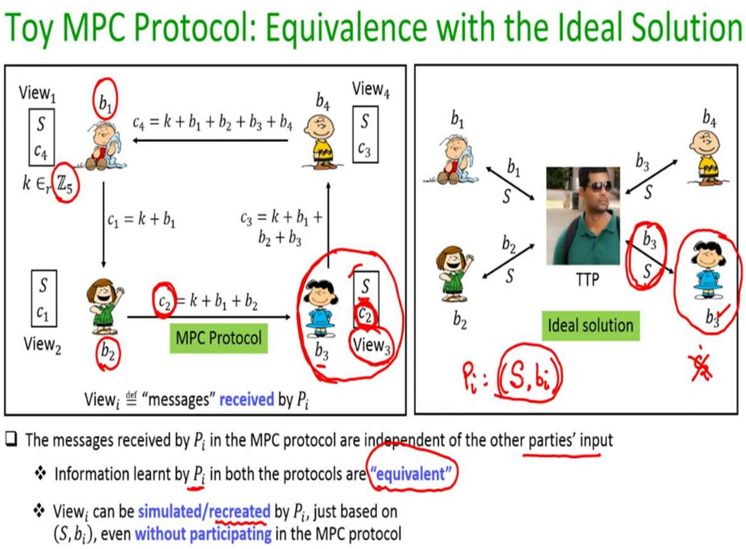
does not talk with party P_2 and participates in an instance of an MPC protocol, just based on what P_3 could have learnt in the ideal solution; in the ideal solution, P_3 would have learnt b_3, S, from the TTP; from that, whatever it can infer about the inputs of the other parties, it can infer.

By saying that it can recreate or simulate this ciphertext c_2, what I mean is that even without participating in an instance of the MPC protocol, it can already conclude that, okay, if I participate in the MPC protocol, I would receive a ciphertext c_2 from the party P_2, whose value will be independent of the actual value of b_2. That means, it could be any ciphertext c_2 from the set Z_5, which I would have received if I would have participated in the MPC protocol and interacted with party P_2.

And that piece of information, she is already aware of that she is going to receive. And this, she is aware based on just the values S and b_3. That means, she could have recreated this view, View 3 herself, without even participating in the MPC protocol. That means, without even participating in the protocol, she can recreate the probability distribution, namely, the set of information which she could have obtained by participating in a real instance of the MPC protocol.

And this is a very powerful statement, because what it shows is the following: If she can recreate, whatever she can from her own input and the final output, then it is equivalent to saying that whatever she would have actually learnt by interacting with the parties in a real execution of the MPC protocol, that is kindly, completely a useless information for her. Useless in the sense, it will not allow a semi-honest P_3 or a corrupt P_3 to learn anything additional about the inputs b_2 and inputs b_1.

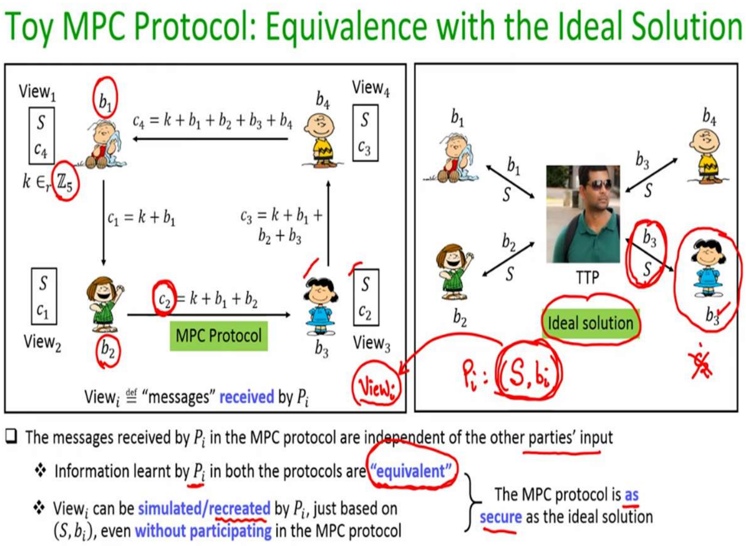
(Refer Slide Time: 10:32)



Because, whatever she could have learnt from a real value of c_2, by seeing the real value of c_2 in the execution of the MPC protocol, even without participating in the MPC protocol and talking to P_2, she could have recreated this information herself. That means, that information c_2 is not a value addition, as far as the View 3 is concerned. And that is what I mean by saying that, the information learnt by party P_i in the MPC protocol is equivalent to whatever the same party P_i would have learnt in the ideal solution.

Equivalent in the sense that if I consider party P_i in the ideal solution, the view of party P_i will be the final output and b_i. By equivalent, I mean to say that, just based on this information, namely, the input and the final output, party P_i could reproduce the same probability distribution with which it could have observed the View i in the MPC protocol.

(Refer Slide Time: 11:38)

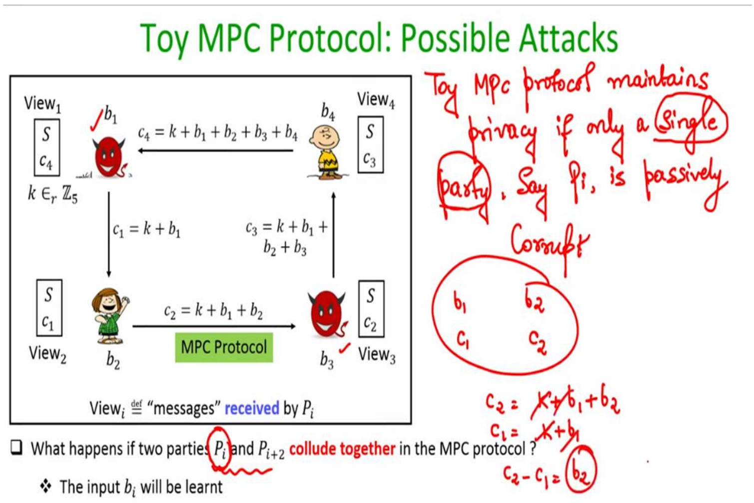


That means, this information itself is sufficient to recreate this. That means, even if P_i does not participate in the MPC protocol and if I just give and tell P_i that, okay, if your input is this, and this is the final output, can you recreate, reproduce the messages which you would have seen by participating in the MPC protocol? The answer is yes. And if those messages could be recreated even without participating in the MPC protocol, that means, whatever information is learnt by P_i, by actually participating in the MPC protocol is of no use.

And that is why this protocol satisfies the privacy condition. And that is why we say that this toy MPC protocol is as secure as the ideal solution. And you cannot have any better secure solution than this ideal solution, because that is the only solution where every party just gets to know its own input and the function output. And if I show that I have designed an MPC protocol whose security level is equivalent or it is as secure as the ideal solution, then that basically ends up showing that my MPC protocol is secure.

So, this is a rough security analysis of our toy MPC protocol. We will make this intuition, namely, the information or the view can be recreated just based on the input and the output of the party later on, but I hope that you are able to understand that why this toy MPC protocol ensures the privacy of the inputs of the other parties.

(Refer Slide Time: 13:25)



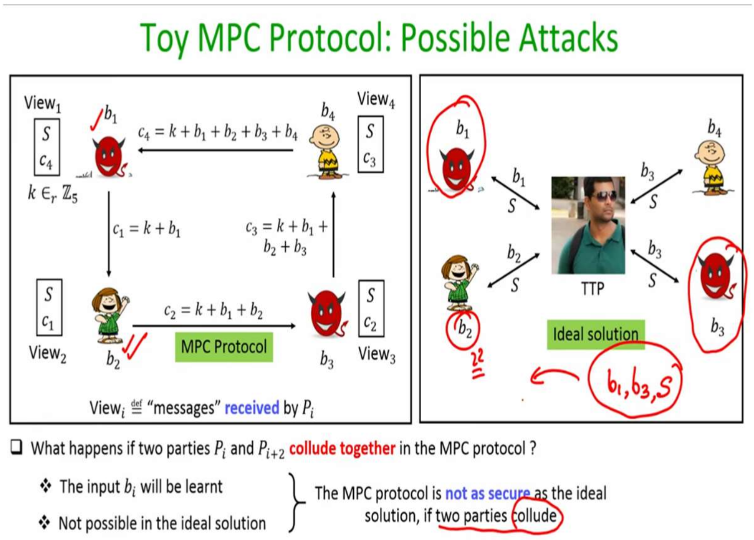
So, now, let us see some possible attacks on this toy MPC protocol. So, till now, the toy MPC protocol, achieves or maintains privacy if only a single party say P_i is passively corrupted. That is what we had proved till now. Only when as long as single party is corrupt, the privacy of the inputs of the other party will be maintained. But what if 2 parties are allowed to get corrupt and collude together in this toy MPC protocol?

Can I say that the privacy of the inputs of the other parties is still maintained in this toy MPC protocol? And the answer is no. For instance, if say the ith party and (i + 2)th party collude together; that means, I take a party P_i and I want to target or attack that party P_i, and my goal is to learn something about the input of that party. And I am now giving you the freedom to let the adversary control any 2 parties in a passive fashion.

So, the attack that can happen, that can be launched here is the following: Imagine that I want to attack the party P_i; say for instance, I want to attack the party number P_2 and want to learn the input b_2; then, if there is a scope of getting 2 parties passively corrupt, then consider a scenario where party number 1 and party number 3 collude together. That means, now, 2 parties can collude together.

Now, if the 2 parties collude together, in this case, P_1 and P_3, then, what are the information they have access to? So, they have collectively got access to c_1 and c_2, and they have the respective inputs b_1 and b_2; all these things are collectively under the control of the adversary. Now, c_2 is the OTP encryption of b_1 and b_2, and c_1 is the OTP encryption of just b_1, under the same key k_1. So, based on these 2 things, if I subtract c_1 from c_2, then basically, that ends up revealing the input b_2. And this attack can be launched against any P_i. If you want to, say find out anything about the bid b_3, then P_2 and P_4 can collude, and so on. So, what this shows is the following: As long as just a single party is allowed to be passively corrupt, my protocol will ensure the privacy property, but if 2 or more number of parties or even if say 2 parties are allowed to collude together, then I can no longer conclude the privacy property.

(Refer Slide Time: 16:55)



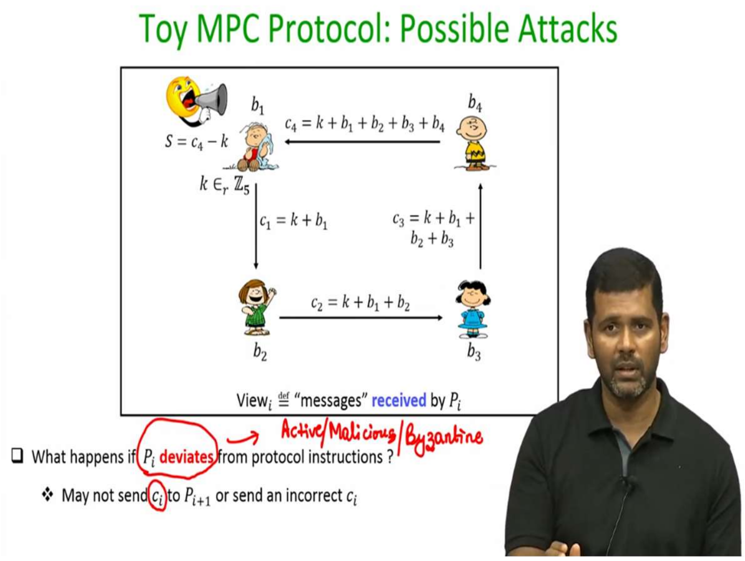
Now, you will be saying that, if 2 parties collude together in the ideal solution, then, there also the input of ith party might be revealed; but the answer is no. Imagine that, again in the ideal solution, P_1 and P_3 collude together, what they will learn? They will learn only b_1, b_3 and S. From this, whatever they can conclude about b_2 and b_3 together, that is fine; but this will not allow the adversary to learn the exact value of b_2.

That is not allowed to be learnt in the ideal solution if P_1 and P_3 collude together. But we have seen an attack scenario in the MPC protocol where if P_1 and P_3 collude together, they come to know completely the value of the second input b_2. That means, now I can conclude that the MPC protocol is definitely not as secure as the ideal solution, if the 2 parties collude together, any 2 parties.

And that itself is a witness for the fact that my MPC protocol is not secure against 2 corruption, because there is something which is not possible in the ideal solution, or now that that attack is not possible in the ideal solution for the adversary to launch, but the same attack is possible for the adversary to launch in the MPC protocol. So, there is a mismatch, and that is why the MPC protocol is not secure for this particular case.

But anyhow, we designed the toy MPC protocol under the assumption that only a single party is allowed to get passively corrupt, as long as this condition is ensured, my toy MPC protocol is secure; but if 2 parties are allowed to corrupt, the toy MPC protocol does not provide me any privacy.

(Refer Slide Time: 18:53)



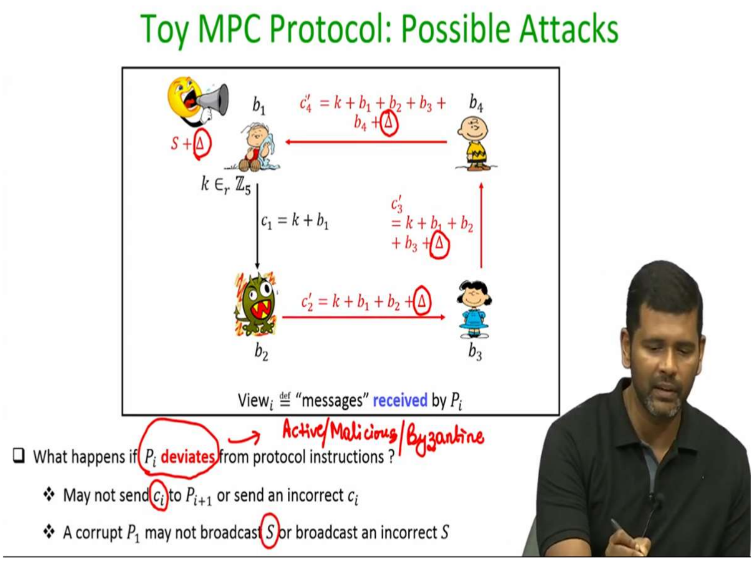
Now, let us see another possible attack here. So, till now, we assumed that in the toy MPC protocol, all the parties will follow the protocol instructions, no one will deviate from the protocol instructions; and even if there is 1 single corrupt party, the corrupt party will behave passively. That means, it will not deviate from the protocol instructions in the sense, whatever values it is supposed to compute and communicate, it will do so.

But what if P_i deviates from the protocol? That means, if P_i is a corrupt party and say it deviates from the protocol instructions; that means, now I am considering active or what we call as also malicious or Byzantine corruption. So, the toy MPC protocol ensures, achieves the privacy correctness property assuming that there is only a single semi-honest corruption. But now, I am trying to analyse the same toy MPC protocol, assuming that I execute it, and during the protocol execution, one of the parties get maliciously corrupt.

That single party could be any of the n parties; it could be the first party, it could be the second party, it could be ith party, it could be the nth party. And depending upon which party gets corrupt and how it deviates from the protocol, we have different consequences. So, let us try to see some of these consequences. So, imagine if P_i gets corrupt. Then, there are 2 possibilities. First of all, it may decide not to forward the encrypted sum of the inputs that it has seen till now, up to that point, to the next party.

That means, it is supposed to compute c_i, which will be the OTP encryption of all the inputs starting from the first party till the ith party, and send it to the next party. But if P_i gets corrupt by an adversary who can cause P_i to deviate from the protocol instructions in any arbitrary fashion, then one possible attack scenario could be that P_i simply gets crashed, the computer gets crashed, and this ciphertext c_i is not getting forwarded to the next party.

(Refer Slide Time: 21:11)

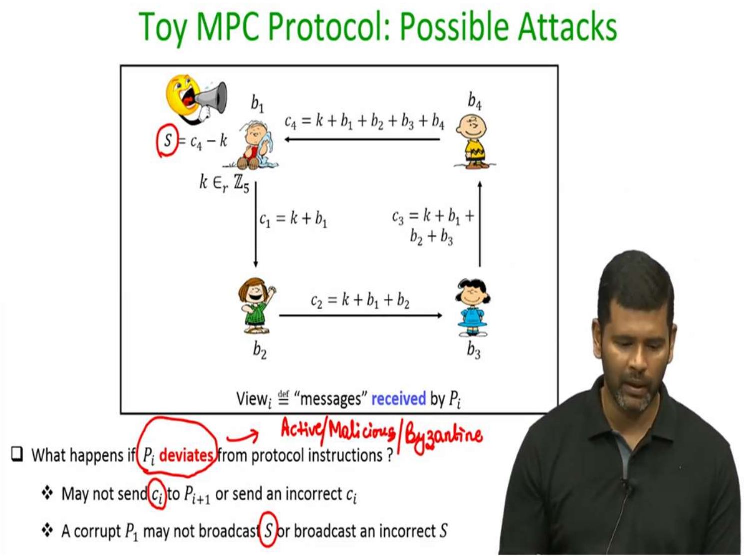


So, for instance, if P_2 gets maliciously corrupt, it is supposed to send this c_2 to party number P_3, but it may decide not to send it. And if it does not send, then P_3 cannot send anything to P_4, P_4 cannot send anything to P_1, and then, the result will not get announced. That is one possible attack scenario. Or worse what can P_2 do is the following: Instead of sending the correct value of c_2, it can send an incorrect value of c_2 to party number P_3.

So, for instance, it can add an error, say delta, and send it to party number P_3; and party number P_3 has no way of verifying as per the toy MPC protocol, whether it is receiving the correct value of c_2 or not. And party number P_3 simply adds her input to this OTP encryption, then actually she is forwarding an incorrect sum to P_4. And P_4 will be further forwarding the incorrect sum to P_1. And now, P_1 will end up announcing the wrong result.

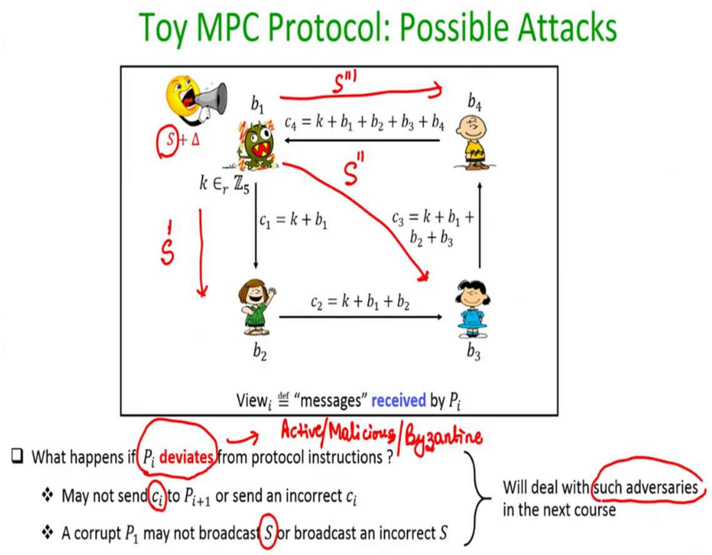
So, that will be a violation of the correctness property. Another attack which may be possible here is that, if P_1 itself gets maliciously corrupt, then what P_1 can do is, it can learn the sum S, because finally it is receiving the ciphertext c_4.

(Refer Slide Time: 22:47)



And then, it decrypts it by subtracting the one-time pad and recovering the sum S. And now, after learning the sum S, it is supposed to announce it publicly to everyone. Now, if P_1 gets maliciously corrupt, then there can be several possibilities which are there. First of all, P_1 may decide not to announce the result S to any other party; but it has learnt the sum. So, that itself is a security breach.

(Refer Slide Time: 23:21)



Or even if it announces the result, it may announce different value of S to different parties. It is supposed to send the same value of S to every party as the result. But if P_1 gets maliciously corrupt, then to say P_2, it gives the value S prime which is different from S; to P_3, it can give a different value of S, say S prime prime; and to P_4, it can announce the value say S triple prime.

That means, just this simple toy MPC protocol completely fails if there is 1 malicious corruption. To ensure that everyone receives the same sum value from P_1, after receiving the value of sum, the remaining parties have to do some cross verification. That means, now the simple MPC protocol will become more involved if I want to tolerate malicious corruption or Byzantine corruption.

So, the purpose of showing you possible attacks on this simple toy MPC protocol is that, even though we have a candidate protocol which works against the semi-honest corruption, the same protocol will become very complicated if I try to now deal with malicious corruptions. So, fortunately, in this course, our scope is restricted only to deal with semi-honest corruptions or passive corruptions. In the next course, we will see how to deal with malicious adversaries or Byzantine corruptions. With that, I conclude this lecture. Thank you.