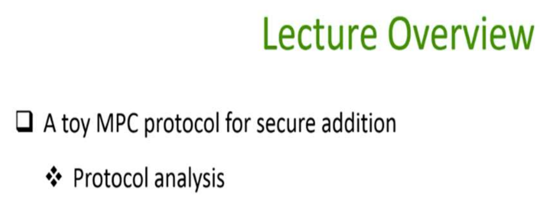
Secure Computation - Part I

Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

Module - 3
Lecture - 17
A Toy MPC Protocol Contd.

(Refer Slide Time: 00:35)

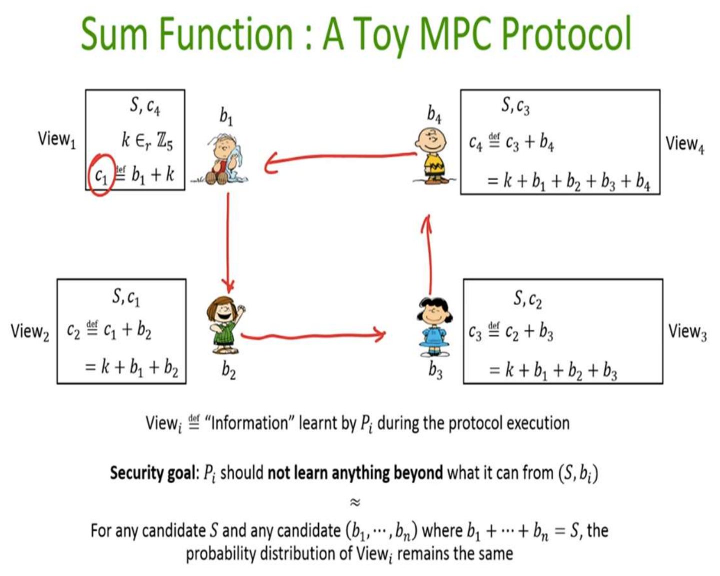


Lecture Overview

- ❑ A toy MPC protocol for secure addition
  - ❖ Protocol analysis

Hello everyone. Welcome to this lecture. So, in this lecture, we will continue with our toy MPC protocol for securely computing the sum, and we will perform the security analysis.

(Refer Slide Time: 00:40)



Sum Function : A Toy MPC Protocol

View$_1$
S, c_4
$k \in_r \mathbb{Z}_5$
$c_1 \overset{\text{def}}{=} b_1 + k$
b_1
b_4
View$_4$
S, c_3
$c_4 \overset{\text{def}}{=} c_3 + b_4$
$= k + b_1 + b_2 + b_3 + b_4$
View$_2$
S, c_1
$c_2 \overset{\text{def}}{=} c_1 + b_2$
$= k + b_1 + b_2$
b_2
b_3
View$_3$
S, c_2
$c_3 \overset{\text{def}}{=} c_2 + b_3$
$= k + b_1 + b_2 + b_3$

View$_i$ $\overset{\text{def}}{=}$ "Information" learnt by P_i during the protocol execution

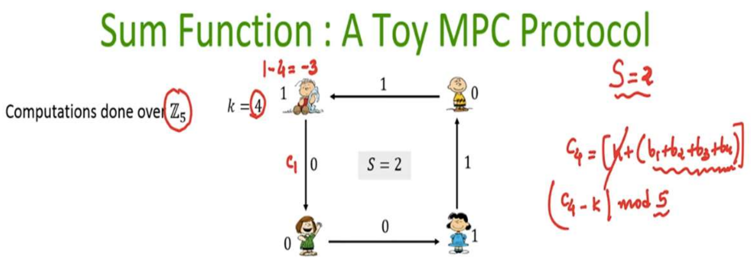
Security goal: P_i **should not learn anything beyond** what it can from (S, b_i)

$\approx$

For any candidate S and any candidate $(b_1, \cdots, b_n)$ where $b_1 + \cdots + b_n = S$, the probability distribution of View$_i$ remains the same

So, just to summarise, this was the toy protocol for computing the sum function. Party 1 computes an OTP encryption of its bid. And then, each party just keeps on adding its own bid in the received ciphertext, and further release it. And finally, by the time the updated ciphertext comes back to P_1, it will be an OTP encryption of the sum of all the input bids, which P_1 can decrypt by unmasking the pad, and announce the result to everyone. And now, we want to analyse that the probability distribution of View i is independent with respect to the inputs of the other parties.

(Refer Slide Time: 01:34)


Sum Function : A Toy MPC Protocol
Computations done over $\mathbb{Z}_5$
$k = 4$
$1-4 = -3$
1
0
1
c_1 0
$S = 2$
1
0
0
1
$S=2$
$c_4 = [k+(b_1+b_2+b_3+b_4)]$
$(c_4 - k)$ mod 5

So, let us try to demonstrate that. And for that, I will take a concrete execution of the protocol, where all the computations, suppose they are performed over Z_5. And now the private inputs in this particular execution for the respective parties are 1, 0, 1, 0. That means, everyone will learn finally that the sum is 2. That means, everyone will know that, every party whose input is 0, from its viewpoint, there are 2 parties who have participated with input bid 1.

We should show here that, from the viewpoint of the party whose input is 0, it could be any 2 parties who have participated with input 1. Whereas, for a party whose input is 1, from its viewpoint, it is the case that there is only 1 other party who has participated with the input 1. We will show here that the messages which are actually exchanged in the protocol does not help him to point out which is the other party who has participated with the input 1.

So, let us see the run of the protocol here. So, imagine that a pad which is used by the first party, which is picked randomly, is the element 4. I stress here, it is only P_1 who knows the value of the pad, it is randomly picked. Everyone will know that a pad could be either 0 or 1 or 2 or 3 or 4, that is all. What exactly is the value of the pad, that is known only to the party number 1. So, the OTP encryption here will be the sum 4 + 1, which is 5.

And remember, all the computations are performed modulo 5. So, $5 \% 5$ will turn out to be 0. Now, that will be the value of c_1 here. Party 2 will add its bid to this ciphertext; so, $0 + 0 \% 5$ will be 0. Party 3 will add its bid to the received ciphertext; $0 + 1 \% 5$ will be 1, which it will communicate to the fourth party. And the fourth party, it will add its input 0 to the ciphertext 1, which will give 1, which it will communicate back to the first party.

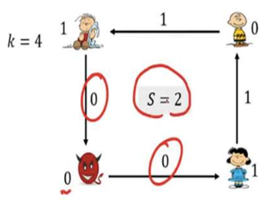
And now, the first party has to subtract the pad. So, it will be computing 1 - 4. 1 - 4 will be -3, but $-3 \% 5$ will be 2. And that is the sum which will be publicly announced. That means, the correctness is guaranteed here definitely. And correctness is very easy to argue here, because the final ciphertext c_4 which is received by the first party, is going to be the summation of k plus $b_1 + b_2 + b_3 + b_4$.

And it is computing $c_4 - k \% 5$. So, $c_4 - k \% 4$ will be this entire value minus k. So, this entire value minus k; k and k cancels out; we will get $(b_1 + b_2 + b_3 + b_4)\%5$. And since the summation of b_1 and b_2, b_3, b_4 is upper bounded by 4; 4 mod, the effect of mod will not happen; so, the correctness will not be violated. So, correctness is very easy to guarantee here. So, now, let us see whether the view of every party is distributed with equal probability with respect to every candidate input values of the parties, which sum up to 2.

(Refer Slide Time: 05:27)

Sum Function : A Toy MPC Protocol

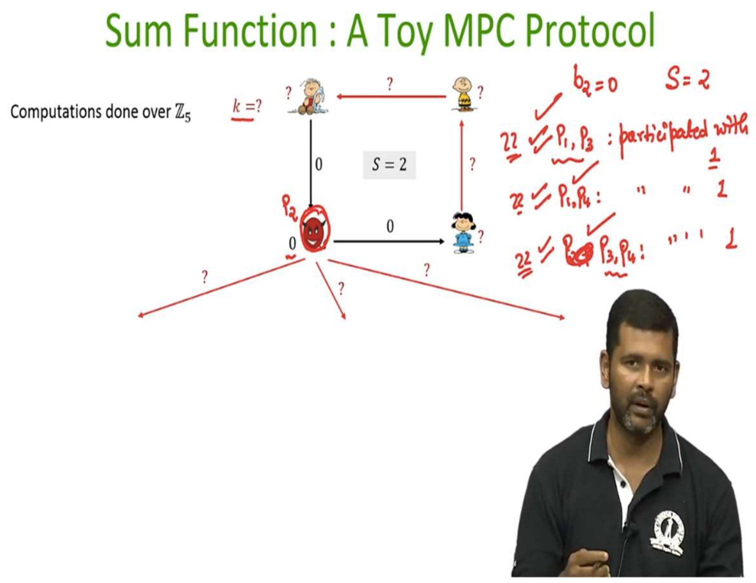
Computations done over $\mathbb{Z}_5$



So, let us assume that, say P_2 is corrupt. P_2 has executed this protocol; now it has gone back to its workplace or home; it has now unbounded resources; it knows the protocol description; it has learnt the final sum; it has received the ciphertext is 0; and it has communicated the ciphertext 0; and now, it is trying to analyse and try to learn something about the inputs of the remaining parties.

If our protocol is secure, then even if it is analysing, even if it analyse with whatever resources, it should not learn anything beyond what it can learn from its input being 0 and the sum being 2.

(Refer Slide Time: 06:09)


Sum Function : A Toy MPC Protocol
Computations done over $\mathbb{Z}_5$
$k = ?$
$S = 2$
P_2
0
?
$b_2 = 0$ $S = 2$
?? ✓ P_1, P_3 : participated with 1
?? ✓ P_1, P_4 : " " 1
?? ✓ P_3, P_4 : " " 1

So, from the viewpoint of this corrupt P_2, what are the things which are not known? The value of the one-time pad which is picked by the first party is not known; what was the input of the first party, that is not known; it has received the ciphertext 0, that is part of its view, so, that is there; and it has communicated ciphertext 0 to the next party, that is also part of its view; but rest of the things which are in question mark, are kind of unknown for this corrupt P_2.

So, now, based on b_2 equal to 0, and S = 2, this P_2 can think of many possibilities. It can think in its mind that it could be the case that P_1 and P_3 participated with 1. That is one possibility, right? Because, in that case, the sum will be 2. Or, it could be the case that P_1, P_4 participated with 1; or it could be the case that 3 and 4 participated with 1. These are the 3 possibilities from the viewpoint of P_2.

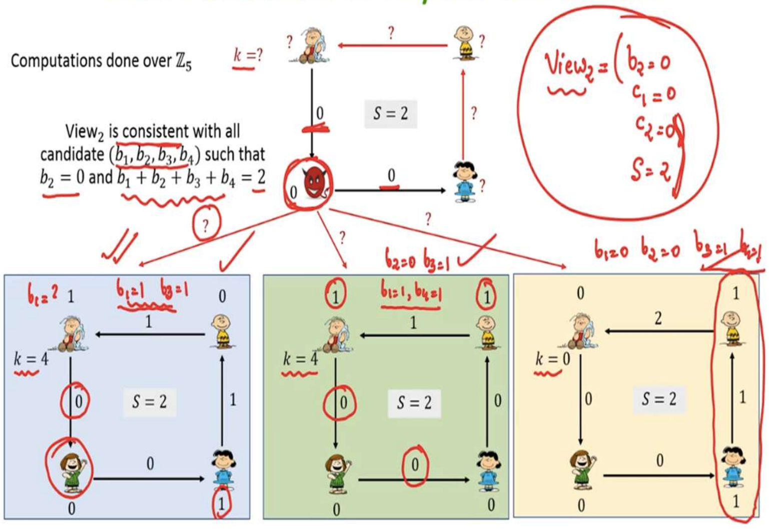
Now, we have to show here that each of these 3 possibilities could actually be the case for this currupt P_2. And it could be the case that indeed P_1, P_3 has participated with input 1; and in that execution, P_2 has received the ciphertext 0 from P_1 and communicated ciphertext 0 to P_3. We have to show that. As well as, we will show that it could also be the case that the first party and the fourth party had participated in input 1; and in the execution, this P_2 has received the ciphertext 0 from P_1.

That is also, we will show. And we will also show that it is also equiprobably the case that P_3, P_4 has participated with input 1, and this P_2 has received the ciphertext 0 from P_1, and communicated ciphertext 0 to the next party. That means, we will show that all these 3 possibilities are equiprobable from the viewpoint of P_2, even if it is given unbounded resources.

And hence, it cannot make out whether it has actually participated in this execution or this execution or this execution; which will end up showing that View 2 is kind of distributed with equal probability. It has equal probability of occurrence for this configuration as well as this configuration as well as this configuration. And hence, View 2 is kind of completely useless from the viewpoint of a corrupt P_2. That is what we are going to argue.

(Refer Slide Time: 09:21)

Sum Function : A Toy MPC Protocol



Computations done over $\mathbb{Z}_5$
$k = ?$
View$_2$ is consistent with all candidate (b_1, b_2, b_3, b_4) such that $b_2 = 0$ and $b_1 + b_2 + b_3 + b_4 = 2$
$S = 2$
$\text{View}_2 = (b_2 = 0,\ c_1 = 0,\ c_2 = 0,\ S = 2)$
$b_1 = ?$
$b_1 = 1 \quad b_3 = 1$
$k = 4$
$b_2 = 0 \quad b_3 = 1$
$b_1 = 1, b_4 = 1$
$k = 4$
$b_1 = 0 \quad b_2 = 0 \quad b_3 = 1 \quad b_4 = 1$
$k = 0$

So, let us take the first possibility from the viewpoint of this corrupt P_2. And remember, this analysis P_2 is now doing after the protocol execution, it is not supposed to analyse these messages and learn anything about the other party's input. But suppose it is semi-honest, after the protocol execution, it is trying to analyse the messages that it has received during the protocol execution and trying to learn about the inputs of the other parties.

So, suppose P_2 tries to check whether it has participated in this configuration; it is making just assumptions regarding, what could have been the input of P_1? what could be the candidate one-time pad that he has used? and is it possible that, with respect to that input and that value of one-time pad, he has sent me the ciphertext to 0? and so on. That is what it is trying to analyse.

So, it is making the assumption here that P_1 has participated with input 1. And P_3 has participated with input 1, b_3 equal to 1. Now, it is quite possible that P_1 has participated with input 1 and communicated the ciphertext 0 to this P_2, if the one-time pad which P_1 has generated has the value 4. It is quite possible. And in that case, once the Cypher text 0 is communicated to this third party, it could have added its input 1 to this ciphertext and forwarded 1 to the fourth party.

The fourth party would have added the input 0, communicated the ciphertext 1 to the first party. And now, the first party would have computed 1 - 4, which is 3. So, this is quite possible; this execution is quite possible. A corrupt P_2 cannot rule down this possibility. Whereas, it is now trying to make a hypothesis that, is it the case that it is this second execution where it has participated.

That means, it is now making the assumption in its mind that, can it be the case that the first party and the last party, they have participated with input 1, and ciphertext 1 equal to 0 was communicated to the second party? Well, that is quite possible again, if the one-time pad which was chosen by P_1 was 4. And then, you can see, if these set of messages would have been communicated, indeed, P_1 would have declared the result to be 2.

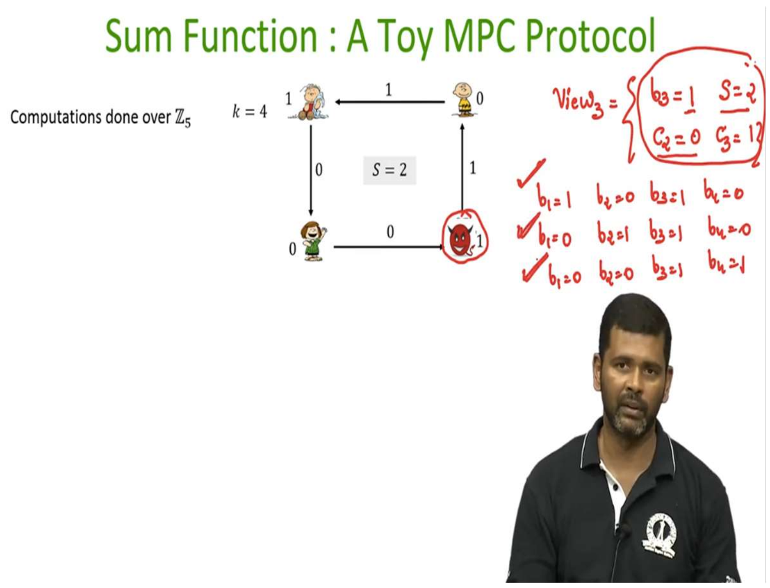
I stress here, the ciphertext 0 which P_2 has received, that is fixed here; and the ciphertext 0 which it has communicated, that is fixed; because, that is what is our actually received by participating in the protocol. All these assumptions are mental exercises which P_2 is making in her mind, trying to find out whether it has participated as per this first box, or as per the second box, or as per the potential third box, where the inputs of the third and fourth party could be 1. Well, that is quite possible that input of party 1 was 0, and it has picked the one-time pad to be 0, in which case it would have communicated the ciphertext 0 to the second party. And now you can see, the remaining messages turned out to be leaked. If the remaining messages turned out to be like this, indeed, P_1 would have announced result S = 2. That means, what we have shown here is the following: View 2 consists of the following:

View 2 consists of b_2 equal to 0, because that is actually the input of the second party, and the ciphertext 1 equal to 0, and ciphertext 2 equal to 0. So, this is one of the possible values of View 2, and of course, sum is equal to 2, the final sum; that is a overall view of the second party. What we have shown here is that, this View 2 is consistent. By consistent I mean, it could occur for the case where b_1 is equal to 1 and b_3 equal to 1; because that is one possibility which is consistent with S = 2.

Because, if S = 2 and if b_2 equal to 0, one of the possibility could be that b_1 is equal to 1 and b_3 is equal to 1. So, that is quite possible. And we have also shown that the same view, View 2, it could also occur for the case where $b_1 = 1$ and $b_4 = 1$ and $b_2 = 0$ and $b_3 = 1$; or the same view, View 2 could occur for the case where $b_1 = 0$, $b_2 = 0$, $b_3 = 1$ and $b_4 = 1$.

That means, for all candidate b_1, b_2, b_3, b_4, such that b_2 is fixed to be 0; because that is actually the input of the corrupt P_2; and such that the sum of these candidate b_1, b_2, b_3, b_4 is 2, this value of View 2 could occur with equal probability. It could occur for this case as well as this case as well as this case. So, a corrupt P_2 cannot narrow down, whether it has actually participated in the execution where b_1 was 1 and b_3 was 1; or it cannot pinpoint whether it has participated in the execution where b_1 was 1 and b_4 was 1; or it could not pinpoint whether it has participated in the execution where b_3 was 1 and b_4 was 1; if P_2 is corrupt and try to do the analysis of the communication.

(Refer Slide Time: 15:47)



Sum Function : A Toy MPC Protocol
Computations done over $\mathbb{Z}_5$
$k = 4$
1
0
1
1
0
0
$S = 2$
0
1
$\text{View}_3 = \{ b_3 = 1 \quad S = 2 \quad C_2 = 0 \quad C_3 = 1 \}$
✓ $b_1 = 1 \quad b_2 = 0 \quad b_3 = 1 \quad b_4 = 0$
✓ $b_1 = 0 \quad b_2 = 1 \quad b_3 = 1 \quad b_4 = 0$
✓ $b_1 = 0 \quad b_2 = 0 \quad b_3 = 1 \quad b_4 = 1$

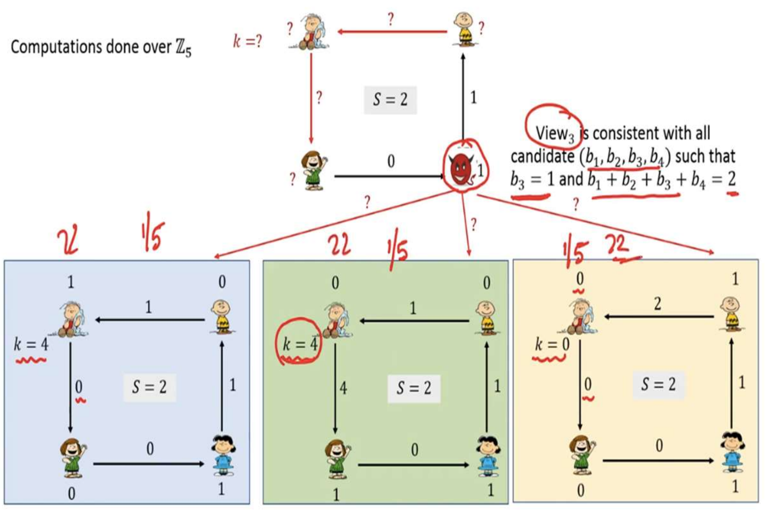
Now, let us retain the same execution and try to argue whether this is the case even for a potentially corrupt P_3. So, if P_3 is corrupt, then View 3 consists of the following: View 3 consists of the information that the input of the party is 1; the overall sum is 2; the ciphertext c_2 that it has received is 0; and the ciphertext c_3 which it has communicated, that is 1. That is the overall view of the third party.

Now, definitely, based on this input 1 and the overall sum being 2, there are the following possibilities. It could be the case that b_1 equal to 1; b_2, 0; b_3 equal to 1, because that is fixed; and $b_4 = 0$, because this configuration can also result in the sum being 2. Or it could be the case that $b_1 = 0$; $b_2 = 1$; b_3 anyhow is 1; and $b_4 = 0$. Or it could be the case that b_1 is 0; b_2 is 0; b_3 is anyhow 1; and b_4 is equal to 1, because this input configuration can also lead to the sum being 2.

We have to show that, from the viewpoint of this corrupt P_3, these set of, this candidate view, View 3 could occur even for this configuration as well as this configuration as well as this input configuration. And hence he cannot pinpoint that the ciphertext c_2 equal to 0 that it has seen, is whether for the first configuration or whether for the second configuration or whether for the third configuration. And hence, it could be any of these 3 possibilities. And hence, it does not learn anything about the actual inputs of the remaining parties. That is what we will show here.

(Refer Slide Time: 17:51)

Sum Function : A Toy MPC Protocol



Computations done over $\mathbb{Z}_5$
$k = ?$
?
S = 2
1
0
View$_3$ is consistent with all candidate (b_1, b_2, b_3, b_4) such that $b_3 = 1$ and $b_1 + b_2 + b_3 + b_4 = 2$
2/2
1/5
k = 4
S = 2
k = 0
4
2

So, these are the, again, same thing what we have done with respect to the corrupt P_2. The question marks here denotes what are the things which are right now unknown from the viewpoint of this corrupt P_3. The things which are not in question mark means, anyhow, that is part of the view of the corrupt P_3. And now, again, P_3 is making this analysis, mental analysis.

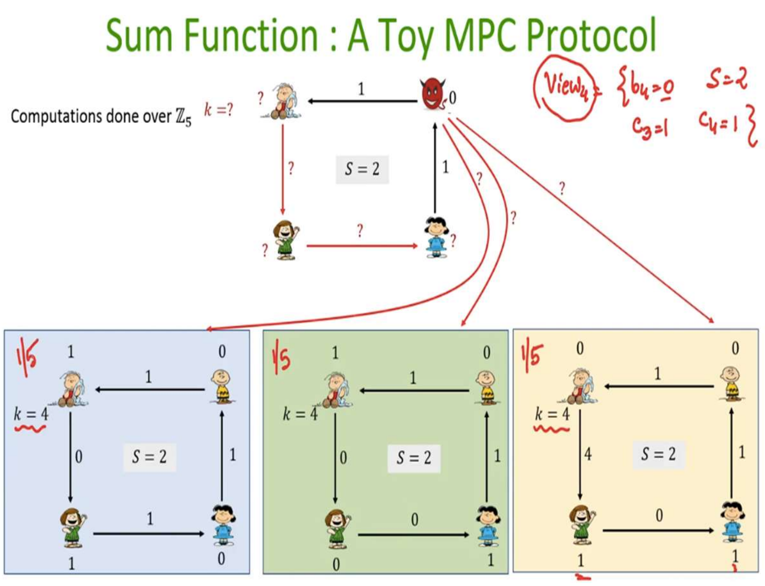
She is asking herself the question that, have I participated in an execution where the input of the first party was 1; my input is anyhow 1; I have received ciphertext 0 from my neighbour, and I have added my input and passed on the ciphertext 1 to my next neighbour; she is asking herself this question. And that is quite possible; because, it could be the case that the one-time pad which P_1 started with was 4, which would have resulted in OTP encryption of 0 being forwarded to the second party; second party would have added 0 as its input to the ciphertext and would have forwarded the ciphertext 0 to the third party.

That is quite possible. Now, the corrupt P_3 is asking in her mind, is it possible that actually the messages that I have received in the protocol corresponds to the case when the second party's input was 1, my input is anyhow 1, second party would have forwarded the ciphertext 0 to me, and I would have forwarded the ciphertext 1 to my neighbour? Again, that is quite possible if the one-time pad picked by the first party would have been 4.

And one-time pad taking the value 4 can occur with probability 1 / 5, because the one-time pad is picked uniformly at random. That means, this first box here could be a possible execution scenario from the viewpoint of the corrupt P_3 with probability 1 / 5. This second box also could be one of the possible execution scenario from the viewpoint of the corrupt P_3, with probability 1 / 5, if the pad would have been 4.

And it is also quite possible that the one-time pad picked by the first party was 0 and its input was 0, and then everything is consistent with view of what third party has seen. But what is the probability that the one-time pad picked by the first party was 0? Again, it is 1 /5. So, that means, with what we have shown here is that, for the fixed $b_3 = 1$; that is the input of the third party; now, with respect to $b_3 = 1$ and the sum being 2, for every candidate b_1, b_2, b_3, b_4, which along with this value of b_3 takes the sum to be 2; this View 3 can occur with equal probability. And hence, P_3 cannot make out whether it has participated in this execution or this execution or this execution.

(Refer Slide Time: 21:12)

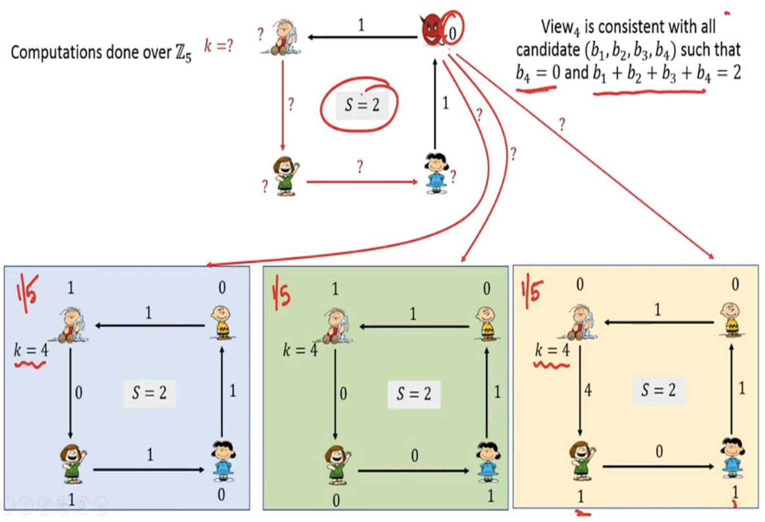

Sum Function : A Toy MPC Protocol
Computations done over $\mathbb{Z}_5$
$k = ?$
$S = 2$
View$_4$ = { $b_4 = 0$ $S = 2$ $c_3 = 1$ $c_4 = 1$ }
1/5
$k = 4$
1/5
$k = 4$
1/5
$k = 4$

Same analysis we can do even for a potentially corrupt P_4 as well. So, View 4 will be now its input being 0, final sum being 2. It has received the ciphertext $c_3 = 1$. It has communicated the ciphertext $c_4 = 1$. The question marks here are the unknown things. And now, let us see whether all potential b_1, b_2, b_3, b_4, with $b_4 = 0$ and overall sum being 2, could be equiprobable from the viewpoint of this candidate View 4.

So, this is one possibility that P_2 and P_3 has participated with input 1. That is one possibility here. And that indeed is possible if the pad which was picked by P_1 was 4. So, this execution can occur with probability 1 /5. Or P_4 is now analysing whether it is the case that the first party and the third party has participated with input 1 and remaining parties have participated with input 0. Again, that is quite possible if the pad would have been 4.
So, this execution is also quite possible with probability 1 /5. Or P_4 is asking in his mind that, is it the case that the second party and the first party have participated with input 1, and overall sum turns out to be 2, and the remaining parties' inputs are 0? Well, again, that is quite possible if the pad would have been 4. And the probability that pad would have been 4 is 1 / 5.

(Refer Slide Time: 23:08)

Sum Function : A Toy MPC Protocol

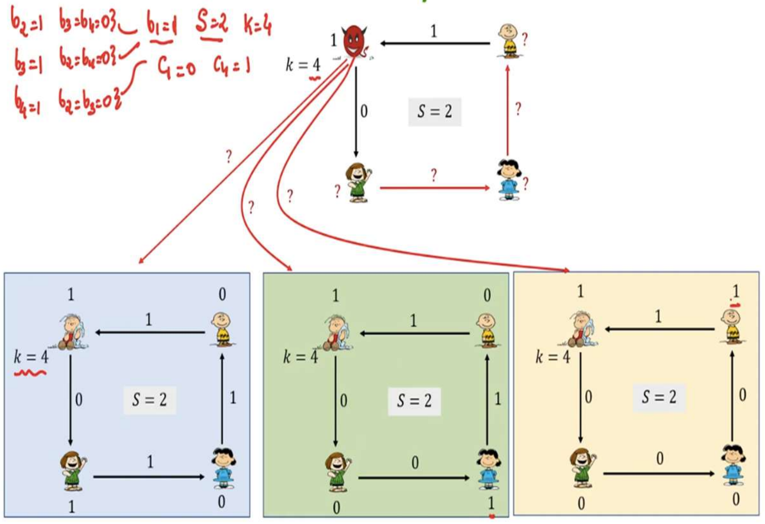


Computations done over $\mathbb{Z}_5$
$k = ?$
View$_4$ is consistent with all candidate (b_1, b_2, b_3, b_4) such that $b_4 = 0$ and $b_1 + b_2 + b_3 + b_4 = 2$
$S = 2$
1/5
$k = 4$

So, again, what we have shown here is that if you take this candidate View 4, it is consistent with every possible b_1, b_2, b_3, b_4 with b_4 being 0, because that is actually the fixed input of this corrupt fourth party; and the summation of these b_1, b_2, b_3, b_4 being 2, because that is the actual sum learnt in the protocol.

(Refer Slide Time: 23:31)

Sum Function : A Toy MPC Protocol



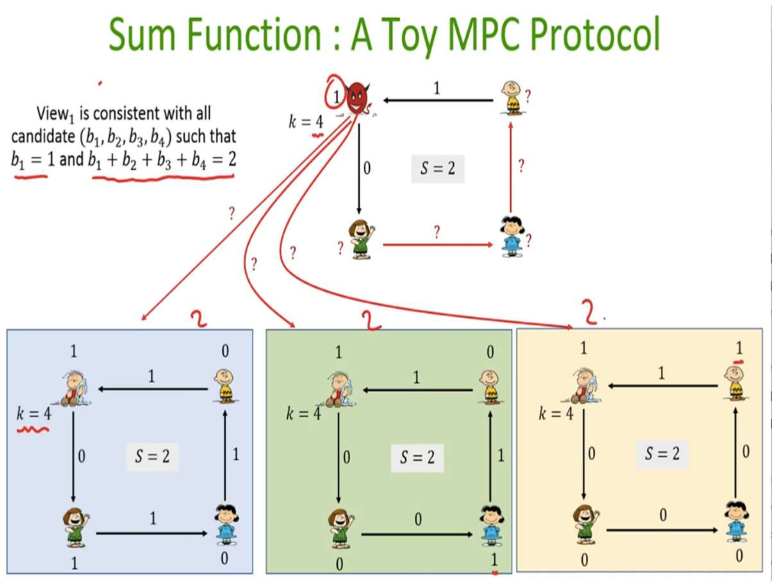
Now, what about the corrupt P_1? So, we have analysed that with respect to 2 only, the View 2 is independent; or with respect to View 3 only, the View 3 is independent of the inputs of the parties; or with respect to View 4, that is independent of the input of the parties. Now, what about View 1? You might be wondering that View 1 has something additional, because the final key which is used for decryption, that is available with P_1.

So, now, let us try to analyse View 1 itself. So, View 1 will consist of $b_1 = 1$; sum being 2; and key being 4; and ciphertext 1 being 0; and ciphertext 4 being 1. So, with respect to b 1 equal to 1 and S = 2, the other possibilities are that only b_2 is 1, remaining other inputs are 0; or b_3 equal to 1, and remaining inputs are 0; or b_4 equal to 1, and remaining 2 inputs are 0.

We have to show that each of these possibilities is consistent with this view, with equal probability. So, let us first consider the case, can it be the case that b_2 was 1, along with b_1 being 1, and k was 4? So, k will be now fixed across all possibilities that P_1 is now thinking in his mind, because that is actually the value of key which it has used, and that is part of its view.

And you can see that it is quite possible that the second party's input was 1, and that is consistent with whatever ciphertext P_1 would have sent to P_2, and whatever ciphertext it would have received from P_4. Also, it is quite possible that the third input was 1; that is again consistent with P_1's view. Or it could be the case that fourth input was 1; that is also consistent with P_1's view.

(Refer Slide Time: 26:02)



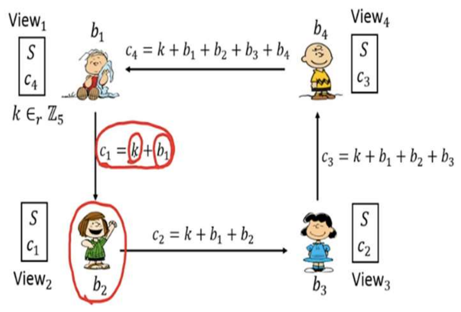
Sum Function : A Toy MPC Protocol

View$_1$ is consistent with all candidate (b_1, b_2, b_3, b_4) such that $b_1 = 1$ and $b_1 + b_2 + b_3 + b_4 = 2$

So, again, this shows that View 1 is consistent with every candidate b_1, b_2, b_3, b_4, where$b_1 = 1$; because that is actually the input of the first party; and the summation of b_1, b_2, b_3, b_4 is 2. That means, P_1 cannot pinpoint whether it is this execution or this execution or this execution.

(Refer Slide Time: 26:31)

Sum Function : A Toy MPC Protocol



❑ For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

So, now, let us try to analyse that why this magic is happening; is it just for the example that we had taken or we can argue that, if we take this general toy MPC protocol for computing the sum function, indeed the privacy property is satisfied? So, now, I have written down the views of each party. So, in the view, I have basically retained the messages which are received by every party; I have not retained the messages which it is communicating to the other parties.

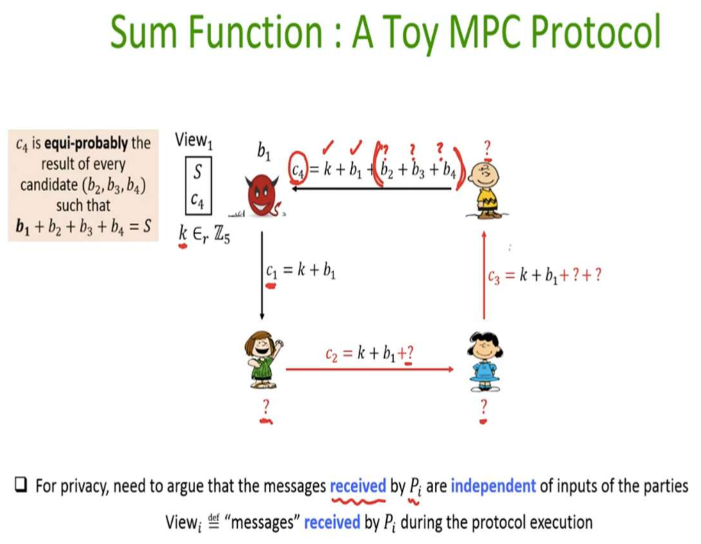
Because, the messages that each party is receiving, based on that and its own input, it can easily find out what is the message it is going to communicate to the neighbour, because that is how the message which each party is communicating to its neighbour is communicated. So, that, I am not explicitly writing down in the rectangular box, I am focusing only on the inputs and the ciphertext that every party has received.

So, to understand that why this magic is working, we have to basically see on what exactly is the ciphertext every party is receiving. So, if I consider a corrupt P_2, what exactly it is receiving? It is receiving an OTP encryption of the bid P_1. And that OTP encryption is indeed going to be a random value that will be completely independent of whatever b_1, P_2 makes as a hypothesis in her mind.

That means if P_2 thinks that, okay, can b_1 be this value? Yes, that is quite possible, because corresponding to that, there is a random key k which P_1 could have taken, and the OTP encryption of that candidate b_1 and that candidate k would have produced this ciphertext c_1. That means, based on this ciphertext c_1, P_2 cannot infer out whether he is seeing b_1 or b_1' or

b_1'' or so on, because the corresponding k is not known to him. And this is the case even for P_3 as well or for P_4 as well.

(Refer Slide Time: 28:43)



Sum Function : A Toy MPC Protocol

c_4 is **equi-probably** the result of every candidate (b_2, b_3, b_4) such that $\mathbf{b_1} + b_2 + b_3 + b_4 = S$

❑ For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

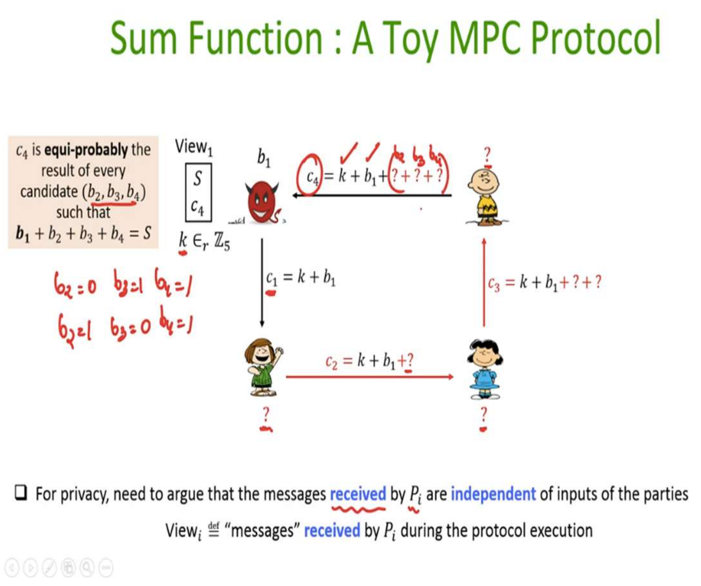
View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

So, let me make it more clear. So, if I consider a corrupt P_1 here, so, what are the messages P_1 receives here? So, this is anyhow his view, it has its own random OTP key here. And this is the ciphertext c_4 that it has received and the ciphertext c_1 which it has communicated. I have to argue here that c_4 and c_1; anyhow c_1, he will be knowing what is c_1, because he himself has communicated c_1.

We have to argue that c_4 does not help him to learn anything additional about b_2, b_3, b_4, because the k is known here, b_1 is known here. It is only the value of b_2, b_3 and b_4 which are not yet known to the corrupt P_1, if it is trying to infer anything about the inputs of the other parties. That means, this portion is not known to him. The sum as a whole is known, but what are the individual things he does not know here? It could be any question mark here.

And indeed, it could be the case that whatever candidate value P_1 makes regarding b_2, it is quite possible that he has forwarded a corresponding c_2. And with respect to that candidate b_3, that was added further to the received candidate c_2, and forwarded to b_4, and so on.

(Refer Slide Time: 30:24)



Sum Function : A Toy MPC Protocol

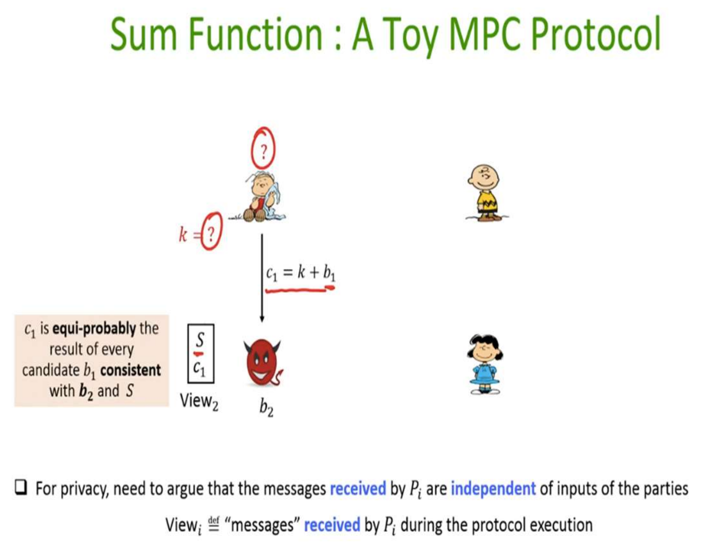
c_4 is **equi-probably** the result of every candidate (b_2, b_3, b_4) such that $\mathbf{b_1} + b_2 + b_3 + b_4 = S$
$b_2 = 0 \quad b_3 = 1 \quad b_4 = 1$
$b_2 = 1 \quad b_3 = 0 \quad b_4 = 1$
View$_1$
S
c_4
$k \in_r \mathbb{Z}_5$
b_1
$c_4 = k + b_1 + ? + ? + ?$
$c_1 = k + b_1$
$c_3 = k + b_1 + ? + ?$
$c_2 = k + b_1 + ?$
?
?
?

- For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

So, the point here is that, even though c_4 is learnt by P_1, it is basically the summation of k and b_1, and the summation of every remaining candidate b_2, b_3, b_4 which can sum up to S. So, that is why, just by analysing c_4, P_1 cannot tell whether it is a case that b_2 is 0, $b_3 = 1$, $b_4 = 1$; or is it the case that $b_2 = 1$, $b_3 = 0$ or $b_4 = 1$, or so on. It could be any candidate b_2, b_3, b_4.

(Refer Slide Time: 31:10)



Sum Function : A Toy MPC Protocol

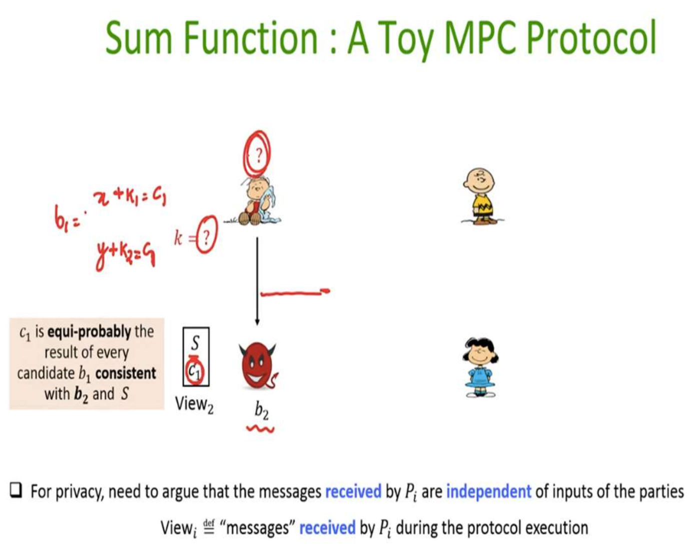
?
$k = ?$
$c_1 = k + b_1$
c_1 is **equi-probably** the result of every candidate b_1 **consistent** with $\mathbf{b_2}$ and S
S
c_1
View$_2$
b_2

- For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

In the same way, if I consider a corrupt P_2, what is the message that it is receiving in the protocol? Anyhow, sum is learnt by her, so, that is not a privacy breach here. But she is learning something about the input of the other party, first party, namely an OTP encryption, but it could be any input encrypted with any key which would have resulted in this sum c_1.

(Refer Slide Time: 31:41)



Sum Function : A Toy MPC Protocol

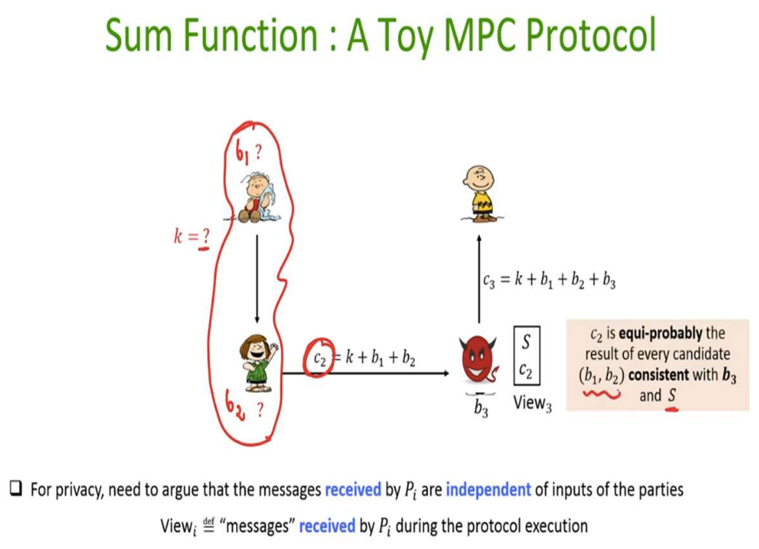
$b_1 = \cdot \; x + k_1 = c_1$
$y + k_2 = c_1$
$k = ?$
?
c_1 is equi-probably the result of every candidate b_1 consistent with b_2 and S
S
c_1
View$_2$
b_2

- For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

So, that is why, from the viewpoint of P_2, it cannot make out whether it is say b_1 equal to a value x or whether b_1 is equal to a value y; because, corresponding to this x, there is some candidate k_1, which would have resulted in the ciphertext c_1, which it has seen; or corresponding to this y, there is a corresponding OTP pad k_2 which would have summed up and produced the ciphertext c_1. So, just based on the value of c_1 that it has received, it cannot make out what is actually this b_1; and that ensures that the View 2 is independent of the input b_1.

(Refer Slide Time: 32:28)



Sum Function : A Toy MPC Protocol

b_1 ?
$k = ?$
b_2 ?
$c_2 = k + b_1 + b_2$
$c_3 = k + b_1 + b_2 + b_3$
S
c_2
View$_3$
b_3
c_2 is equi-probably the result of every candidate (b_1, b_2) consistent with b_3 and S

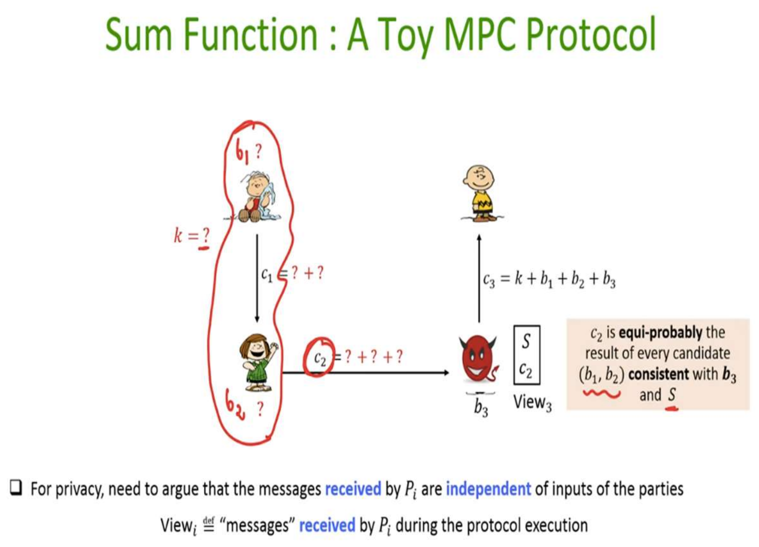
- For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

In the same way if I consider a corrupt P_3, a corrupt P_3 receives this ciphertext. Now, it does not know the following: what is b_1, what is b_2, and what is k. Now, for every candidate b_1, b_2 that P_3 makes in her mind, which is consistent with S, there is a corresponding k such that, that corresponding k along with the candidate b_1 and b_2 would have produced the actual c_2 which

P_3 has received during the protocol execution. And that is why View 3 is independent of every potential b_1, b_2 pair.

(Refer Slide Time: 33:13)



Sum Function : A Toy MPC Protocol

c_2 is **equi-probably** the result of every candidate (b_1, b_2) **consistent** with $\boldsymbol{b_3}$ and S

❑ For privacy, need to argue that the messages received by P_i are independent of inputs of the parties

View$_i \overset{\text{def}}{=}$ "messages" received by P_i during the protocol execution

And I can run the same argument even for P_4 as well. And that is why, in this whole protocol, if a single party is corrupt and try to analyse the ciphertext that it is receiving from its predecessor, it cannot make out anything about the actual value of the inputs of the other parties. The actual inputs of the other parties could be any candidate inputs, consistent with the view of the corrupt party and the overall sum.

And that ensures that this simple toy MPC protocol indeed helps the parties to compute the sum function in a secure way, which actually looked like an impossible task when we started with the problem description, but we showed here that even by exchanging messages among themselves, the parties can securely compute the sum function, without revealing anything additional about the respective inputs. Thank you.