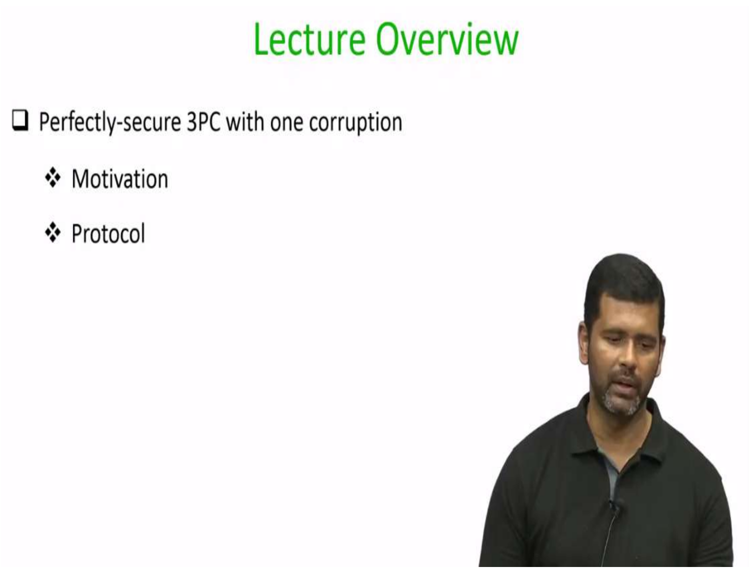
Secure Computation: Part 1
Prof. Ashish Choudhury
Department of Computer Science
Indian Institute of Science – Bengaluru

Lecture – 31
Perfectly-Secure MPC for Small Number of Parties

Hello everyone. Welcome to this lecture.

(Refer Slide Time: 00:31)

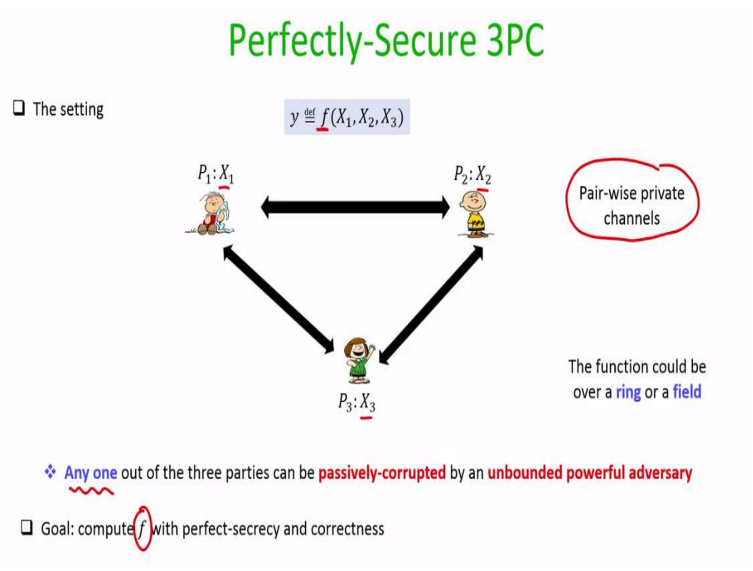


Lecture Overview

- ❑ Perfectly-secure 3PC with one corruption
  - ❖ Motivation
  - ❖ Protocol

So, the plan for this lecture is as follows. In this lecture, we will focus on perfectly secure multiparty computation involving 3 parties and among those 3 parties there could be one semi honest corruption. We will first see the motivation for studying this special case of multiparty computation namely the case of 3 party computation which we also called as 3PC and then we will see a concrete protocol.

(Refer Slide Time: 01:02)



Perfectly-Secure 3PC

- The setting

$y \overset{\text{def}}{=} f(X_1, X_2, X_3)$

$P_1 : X_1$
$P_2 : X_2$
$P_3 : X_3$
Pair-wise private channels
The function could be over a ring or a field

- Any one out of the three parties can be passively-corrupted by an unbounded powerful adversary

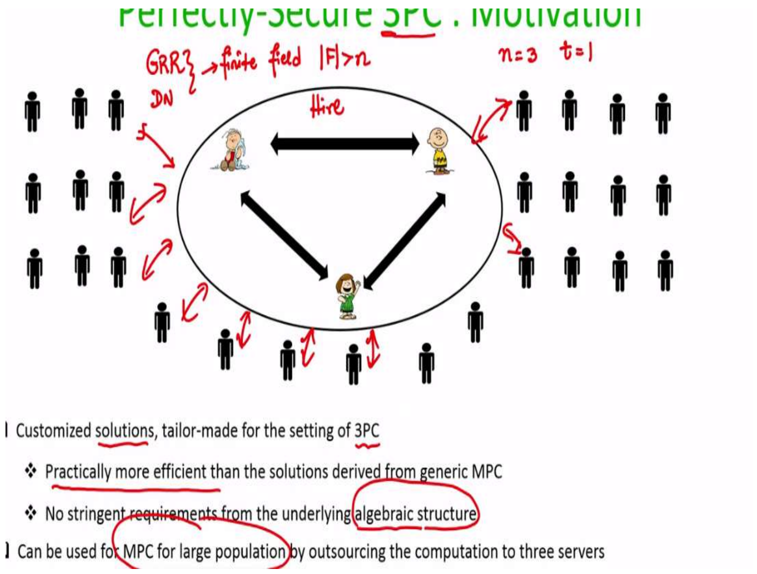
- Goal: compute f with perfect-secrecy and correctness

So, let us first understand the problem setting we have a set of three mutually (()) (01:08) parties P $_1$, P $_2$, P $_3$. Of course, we are in the pair-wise private channel model where we assume that there is a private channel between every pair of parties here and each party has some private input. So, the inputs are x $_1$, x $_2$, x $_{\text{vaq}}$. They could be from a ring or from a field there is no restriction on the algebraic structure.

And there is a publically known function f and the goal is to securely compute the function f and corruptions scenario here is that we assume that any one party out of these 3 parties can be passively corrupted by an unbounded powerful adversary that means after the protocol it gets over any one of these 3 parties might try to analyze the protocol transcript and try to infer anything additional beyond its own input and the function output.

And we want to prevent any kind of leakage from happening through the protocol. So, the goal is to design a protocol which allows the parties to compute the function with perfect secrecy and correctness. Perfect secrecy means that the view of the semi honest corrupt party should be independent of the inputs of the honest parties that means it can be simulated by a simulator and by correctness we mean there is the output y which is finally obtained by the parties is the correct output.

(Refer Slide Time: 02:57)



Perfectly-Secure 3PC : Motivation

- Customized solutions, tailor-made for the setting of 3PC
  - ❖ Practically more efficient than the solutions derived from generic MPC
  - ❖ No stringent requirements from the underlying algebraic structure
- Can be used for MPC for large population by outsourcing the computation to three servers

So, the question is why we want to focus on this specific case of 3PC because till now we are talking about n party computation out of them t could be corrupt or we could have non-threshold adversary, but then suddenly we are now talking about this special case of multiparty computation where n = 3 and t = 1 and there are several reasons for doing this for studying this special case of 3PC.

The first reason is that there are customized solutions which are tailor made specifically to deal with the setting of 3 party and tolerating one corruption that means these techniques will not be applicable for general n party computation and tolerating t corruptions not applicable in the sense you would not get the same efficiency which you would have obtained by applying those techniques to the special case of 3 party computations tolerating t equal to 1 corruptions.

And those solutions those customized solutions as they are tailor made for 3PC they gives you practically more efficient MPC protocols compared to what you could have derived by a plugging in n = 3 and t = 1 in any generic MPC protocol. So, remember we have generic perfectly secure MPC protocols tolerating t less than n / 2 corruptions. So, for instance, we had seen the protocol based on Gennaro Rabin-Rabin degree reduction method.

We also have the protocol by Damgaard Nielsen and so on. So, those protocols are for any number of parties and tolerating any t number of corruptions where t is less than n over 2. So, now I substitute n = 3 and t = 1 in those protocols I will have a solution for the 3PC. I will have a perfectly secure MPC protocol among 3 parties tolerating one corruption, but those solutions will be inefficient.

They will require more communication, more number of rounds compared to the customized solution the tailor made solutions that we have for the special case of 3PC. Moreover, another motivation for studying this special case of 3PC is that in the protocols we do not require any stringent requirements from the underlying algebraic structure over which the computations are performed.

So, for instances both these protocols Gennaro Rabin-Rabin, Damgaard Nielsen they require a finite field, they require a function over a finite field and that finite field should have cardinality greater than n the number of parties whereas as we will see later these 3PC protocol they can be designed, the computations in these 3PC protocols they can be performed both over the ring as well as over the field.

They are on no stringent requirements that you need to necessarily have the function over the field. In fact we will be seeing 3PC protocols where the computations can be performed over a Boolean ring itself. So, you do not require a field whose cardinality is greater than the number of parties so that is another motivation and the third motivation is that the special case of 3PC can be used for MPC among a large population.

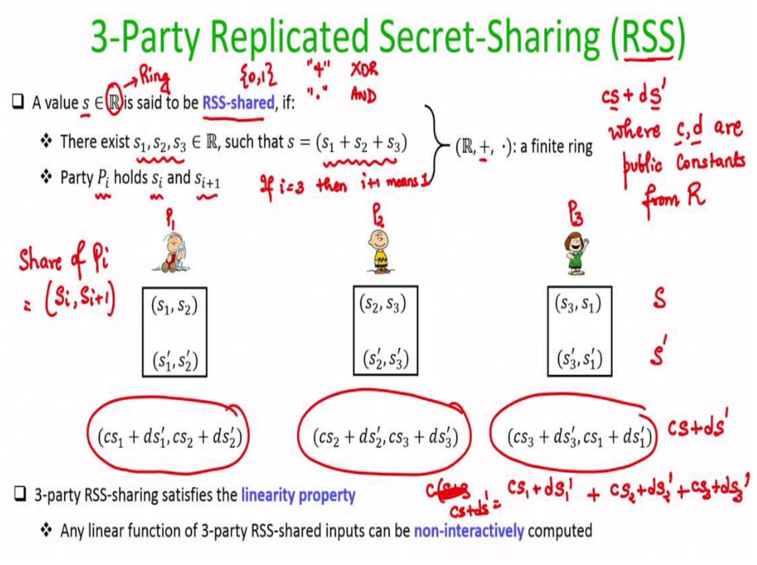
So, you might be wondering that these 3PC solutions it cannot be used in a scenario where there are more than three number of parties to actually who are interested to perform computation on their private data, but that need not be the case. What we can do is we can outsource MPC for those large population and that outsourcing can be done to this three servers.

So, imagine you have enormous large number of parties spread geographically and they are not interested to run the generic MPC protocols because running the generic MPC protocol over such a huge population might be inefficient. Instead what those parties can do is they can hire these three servers and they can make their inputs available in secret shared fashion or whatever secure way they want among these three servers.

And then let these three servers perform the computation for all those parties compute the function output and give back the function output to the parties and the guarantee that we obtain here is that as long as only one of these three servers the (()) (08:06) servers is semi honestly corrupted the privacy and the correctness of the computation is guaranteed. So, it is not the case that by deriving techniques tailor made only for 3PC.

We cannot perform MPC computation secure MPC computation. We can actually perform secure MPC involving multiplies parties by outsourcing the computation of those n parties to this three servers whose service can be hired and indeed there are several practical deployments of several 3PC protocols and very nice results have been obtained. So, now we have several 3 party protocols which can perform very fast computation in a private fashion.

(Refer Slide Time: 09:05)



3-Party Replicated Secret-Sharing (RSS)

- ❑ A value $s \in \mathbb{R}$ is said to be **RSS-shared**, if:
  - ❖ There exist $s_1, s_2, s_3 \in \mathbb{R}$, such that $s = (s_1 + s_2 + s_3)$
  - ❖ Party P_i holds s_i and s_{i+1}

$(\mathbb{R}, +, \cdot)$: a finite ring

P_1	P_2	P_3
(s_1, s_2)	(s_2, s_3)	(s_3, s_1)
(s_1', s_2')	(s_2', s_3')	(s_3', s_1')
$(cs_1 + ds_1', cs_2 + ds_2')$	$(cs_2 + ds_2', cs_3 + ds_3')$	$(cs_3 + ds_3', cs_1 + ds_1')$

- ❑ 3-party RSS-sharing satisfies the linearity property
  - ❖ Any linear function of 3-party RSS-shared inputs can be non-interactively computed

Ring
{0,1}
"+" XOR
"·" AND
If i=3 then i+1 means 1
Share of Pi = (Si, Si+1)
S
S'
cs+ds' where c,d are public Constants from R
cs+ds'
cs+ds' = cs1+ds1' + cs2+ds2' + cs3+ds3'

So, there are several motivations we have seen already so now will see the concrete protocol and again the protocol will be based on that blueprint of shared circuit evaluation where it will be ensured that all the values in the computation remains secret shared among the parties with threshold being t = 1 because there could be up to one corruption so the threshold will be t = 1.

So, it will be ensured that all the values in the computation will be secret shared and all the gates will be evaluated in a secret shared fashion and then finally once the function output is ready in secret shared fashion the parties go and reconstruct the function output, but now the difference here will be we would not be using the Shamir secret sharing protocol because we now want more efficient protocols because if we finally resort to Shamir secret sharing protocol then it is basically resorting to the generic MPC protocol.

That is not what we want to do here. We want to design special secret sharing protocols, special way of performing multiplication, addition, reconstruction and all those techniques will be tailor made for the case of 3 parties. So, the secret sharing that we will be using in this MPC protocol, in the 3PC protocol is based on replicated secret sharing. So, let us first try to understand what is this replicated secret sharing which I will also call as RSS.

So, imagine you are given a ring R and there is no restriction on the cardinality of the ring, it could be as simple as the Boolean ring consisting of just the values 0 and 1 where the plus operation is actually XOR operation and the dot operation is the AND operation, but in general it could be any ring. So, imagine there is a value s in the ring. We will say that the value is secret shared as per replicated secret sharing or RSS shared if there exist three values from the ring s_1, s_2, s_n.

Such that the summation of s_1, s_2, s_3 is equal to the secret s where the summation means the plus operation of the ring and each party P_i holds the piece s_i and s_{i+1} where if $i = 3$ then $i + 1$ means 1 because $3 + 1$ will be 4, but there is no fourth party or fourth piece. So, basically we do the wrap around and then basically by $i + 1$ I mean first $i + 1$ will denote the value 1.

So, pictorially you can imagine that P_1 will have the pieces s_1, s_2 P_2 will have the pieces s_2, s_3 and P_3 will have the pieces s_3 s_1 and it is easy to see that this 3 party replicated secret sharing satisfies the linearity property what does linearity mean? That means that any linear function of secret shared inputs can be computed locally that mean if there is a linear function whose inputs are available in a secret shared fashion among these 3 parties as per this sharing semantic then the parties can apply the same linear function.

It can compute the linear function on the secret shared value itself that means by applying the linear function on their respective shares of the inputs they can obtain their respective shares of the output where the shares actually constitute a replicated secret sharing of the function output. So, imagine you have two values S and s' secret shared as per replicated secret sharing namely you have three pieces s_1, s_2, s_3 summing up to s.

And you have three pieces s'_1, s'_2, s'_3 prime summing up to s' then each party can do the following. It can P 1 can add up component wise its pieces of s and s'. So, let us take a more generic linear function addition also is a linear function, but imagine that the parties want to compute a function which is c times s plus d times s' that is a function which parties want to compute where c ,d are public constants from R.

So that is a requirement but s and s' should not be revealed in the process. If that is the case then what the parties can do is the following. The parties compute a linear combination of their respective shares of s and s' as per the linear combiners c and d. So, party 1 it computes this linear combination so it will obtain a pair of values, P_2 will obtain a pair of values and independently P $_3$ will obtain a pair of values.

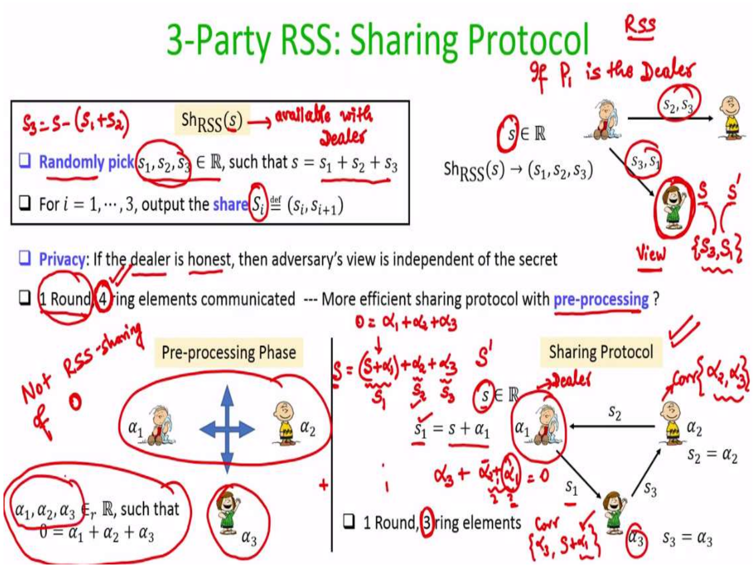
And now if you see that this three pieces in the sense that the values c times s_1 + d times s_1' and then c times s_2 + d time s_2' and then c times s $_3$ + d times s_3'. If you sum all of them together you get the values c times s plus d times s' and as per the semantics of replicated secret sharing the value c times s plus d times s' is now available in a secret shared fashion.

So that means any linear function can be computed locally and in this replicated secret sharing the share for each party will be a pair of elements it is not one element. So, share of each party, share of P_i = the values S_i and S_{i+1}. So, definitely in terms of share size this is bad because if we consider the Shamir secret sharing there the size of each share is the same as the size of the secret.

That means if secret is a single field element and the ith share also will be a field element, but here if the secret is a field element of course all the operations I am doing here over the ring, but the same operations can be performed over the field as well. So, imagine if my secret would have been from a field then the share size for each party will be twice the number of elements or twice the number of bits basically that you require to represent one field element.

So, the share size is twice compared to the secret size so that is bad, but later will see that even though the share size is more than the secret size for each party if you perform secret shared circuit evaluation as per the semantics of replicated secret sharing then we get more efficiency. So, that is the sharing semantic of replicated secret sharing.

(Refer Slide Time: 17:12)



3-Party RSS: Sharing Protocol

$\mathsf{Sh}_{\mathsf{RSS}}(s)$

- Randomly pick $s_1, s_2, s_3 \in \mathbb{R}$, such that $s = s_1 + s_2 + s_3$
- For $i = 1, \cdots, 3$, output the share $S_i \overset{\text{def}}{=} (s_i, s_{i+1})$

$s \in \mathbb{R}$

$\mathsf{Sh}_{\mathsf{RSS}}(s) \rightarrow (s_1, s_2, s_3)$

- Privacy: If the dealer is honest, then adversary's view is independent of the secret
- 1 Round, 4 ring elements communicated --- More efficient sharing protocol with pre-processing ?

Pre-processing Phase

$\alpha_1, \alpha_2, \alpha_3 \in_r \mathbb{R}$, such that $0 = \alpha_1 + \alpha_2 + \alpha_3$

Sharing Protocol

$s \in \mathbb{R}$

$s_1 = s + \alpha_1$

$s_2 = \alpha_2$

$s_3 = \alpha_3$

- 1 Round, 3 ring elements

RSS
If P_1 is the Dealer
$s_3 = s - (s_1 + s_2)$
available with Dealer
s_2, s_3
s_3, s_1
s s'
View $\{s_3, s_1\}$
$0 = \alpha_1 + \alpha_2 + \alpha_3$
$s = (s + \alpha_1) + \alpha_2 + \alpha_3$
s'
Not RSS-sharing of 0
α_1
α_2
α_3
Dealer
Corr $\{\alpha_2, \alpha_3\}$
$\alpha_3 + \alpha_2 + \alpha_1 = 0$
Corr $\{\alpha_3, s + \alpha_1\}$

Now we have to see the protocols for sharing a value as per replicated secret sharing and now if a value is secret shared as per replicated secret sharing how you can reconstruct it and so on. So, imagine s is a value available with a dealer and it needs to be secret shared by the dealer. So, what the dealer is going to do as part of the secret sharing protocol it will randomly pick three values from the ring.

So, I am assuming that the secret is from the ring and all the computations are also performed over the ring. So, it will randomly pick three values from the ring such that the summation of those three values is equal to the secret S. How it can do that? So, it can randomly pick s_1 and s_2 and it can then set s_3 to be the difference of s and $s_1 + s_2$ and then the share for the ith party will be the pair of values s_i and s_{i+1}.

So, pictorially if P_1 is the dealer then it will do the following if it wants to secret share the value s then it will pick three random pieces s_1, s_2, s_3 which sums up to s. His own share for the secret share of s will be s_1, s_2 so we need not have to communicate s_1, s_2 to himself, but he has to communicate the shares to the other two parties so he communicates those required shares over the private channel.

Similarly, if P_2 would have the value s' which it needs through secret share it will communicate the values s'_1 , s'_2 , s'_3 summing up to s' and then distribute the respective shares and so on. Now, it is easy to see that in this protocol if the dealer is honest then the privacy of s is maintained namely among these 3 parties if one of the parties is corrupt.

And of course I am considering the case when that corrupt party is not the dealer because if the dealer himself is corrupt then it itself it knows the full secret because itself has generated the shares. We have to argue about the privacy when the dealer is honest. So, my claim is that if the dealer is honest then an adversary who can corrupt any one of the remaining 2 parties does not learn any information about the secret namely its view will be independent of the secret.

Namely whatever share it receives will be independent of the secret and that follows from the properties of additive secret sharing which we had already seen earlier. So, imagine P_2 is bad so what exactly will be your view? The view will be the probability distribution on the pieces s_3 and s_1. Why probability distribution because this values s_3 and s_1 they are randomly picked they are not fixed values.

So, that is why her view will be having a probability distribution over s_3 and s_1, but this s_3 and s_1 could be equal likely the shares for the secret s or equal likely they could be the shares for s' as well because it is quite possible that there is a value s_2 which along with this s_1 and s_3 will give as the secret and it is equiprobably the case that there is another value say s_2' which along with the same s_1 and s_3 would result in the secret s'.

But since the piece s_2 is randomly chosen in the secret sharing algorithm in this replicated secret sharing this corrupt party cannot infer whether she is seeing the shares for the secret s or for the secret s' that means whatever communication happens in this protocol and whatever is the view generated for the adversary that can be easily simulated that means this adversary's view can be simulated and the distribution of that simulated view will have the same as the view generated in the actual secret sharing protocol.

The protocol will require one round of communication because it is only the dealer who has to talk and distribute the shares and it can distribute the shares independently. It is not the case that it has to first communicate the shares s_2, s_3 and then it has to communicate the shares s_3, s_1 no because it has independent channels with independent parties. So, it can use those channels at the same time.

And the total communication here is 4 ring elements that means it either has one ring element to secret share then it has to totally communicate 4 ring elements. So, now what we will do here is the following. We will see that can we reduce the cost of this secret sharing protocol in terms of communication of course interaction has to be there. Dealer has to distribute the shares because if the dealer does not distribute the shares then how can secret be made available in a shared fashion.

So, definitely we cannot compromise on the number of rounds here because that is the optimal thing here one round. We have a possibility of further improving the communication complexity assuming that there is some pre processing phase which will be done before the actual sharing happens and remember we had already seen the motivation for pre processing based MPC.

The idea is that if there is a scope of shifting or preponing all the complex operations in a phase which is independent of the circuit and then using the data generated in the pre processing phase if we can optimize, if we can make the actual shared circuit evaluation faster then there is a big advantage and we are trying to follow the same principle here as well.

We are trying to see whether there is a possibility that the parties generates some raw data from pre processing information in a function independent, circuit independent, input independent pre processing phase and with the help of that raw data the actual communication complexity of this sharing protocol gets reduced. So, the answer is yes. Imagine the pre processing phase the parties generate random values α_1, α_2, α_3 in a secret shared fashion.

Such that those α values are random, but they sum up to 0. So, you might be wondering that if they sum up to 0 then how can you say α_1, α_2, α_3 are random? Well, it is quite possible that α_1 and α_2 are randomly chosen and then I set α_3 to be the difference of or minus of α_1 plus α_2 even that will be a random vector of three values summing up to the value 0.

So, imagine in the pre processing phase the parties interact there is a mechanism by which they interact and generate this setup. How this setup will be generated that will be focus of our next lecture, but imagine for the moment that whenever there is a requirement of generating such system here namely a system of three random values summing up to 0 and P_1 having the first component, P_2 having the second component, P_3 having the third component that is always possible.

And remember here this is not RSS sharing of 0 no because the parties know that the value 0 has been shared, there is nothing called secret here which has been shared P_1, P_2, P_3 will know that okay it is a zero value which has been shared P_1 will have only one of the shares namely α_1, P_2 will have one of the shares namely α_2 and P_3 will have the third share namely α_3.

And you require here at least 2 parties to collaborate to learn the full αvector. So, if say P_1 and P_2 collaborate then they can find out what is $\alpha 3$ because they know that $\alpha 3$ when summed up with $\alpha 1$ and $\alpha 2$ should have produced a_0, but if only one party tries to find out what are the remaining two pieces it cannot find out that means if I just consider P_3 will have α_3, but it will not be knowing what is the α_2.

And it will not be knowing what is the α_1 and it knows this sum up to 0 so it could be any α_2 and any α_1 with along with α_3 could sum up to 0 not any once it fixes α_2 corresponding to that α_2 there will be a corresponding α_1 which along with the fixed α_2 and the α_3 which is available with P_3 will sum up to 0 so that is the set of parties would have generated.

Assume for the moment that such pre processing is possible. Now assuming such pre processing is possible and now say P_1 is the dealer who wants to secret share a value s then it can do it as follows. Similar actions can be done if P_2 would have been the dealer or P_3 would have been the dealer. So, it is not necessary that for this sharing protocol it is only P_1 who is the dealer.

For the purpose of demonstration I am just taking the case where only P_1 is the dealer, but similar actions can be performed if P_2 is the dealer or P_3 is the dealer. Whenever a value has to be shared by the dealer we will assume that in the pre processing phase a random vector of α, α_1, α_2, α_3 have been generated it is summing up to 0. Now, the idea behind this efficient sharing phase protocol is the following.

We know that the summation of α_1 and α_2 and α_3 is 0. Now what can we say about the summation of $s + \alpha_1$ and α_2 and α_3? Well, we can say that the summation of these values is actually s and our goal is to generate a secret sharing of s as per replicated secret sharing. So, based on this observation what we can do here is the following. Party 1 who is the dealer actually will compute this value s_1 and this value s_1 is said to be the summation of s and α_1.

So, you can imagine it has an OTP encryption of the secret s which will be now given to P_3. Why given to P_3? Because remember as per the sharing semantics of RSS the secret which has to be shared has to be divided into three pieces summing up to the secret and each party should have two of those three pieces. So, basically we know here from the mathematics from the algebra here that s is actually the summation of these three pieces.

So, I can set the term here in the parenthesis as s 1. I can set α_2 to be s_2, I can set α_3 to be s_3. So, now we have three pieces whose sum is actually s and we need to ensure that each party should have two of these three pieces as per the semantics of replicated secret sharing. So, for instance, P_3 should have the third piece as well as the first piece. Third piece is already available with him α_3 from the pre processing phase itself.

The first piece needs to be communicated to him and the dealer can communicate that piece to him. In the same way the second party he already have the second piece here namely α_2, but he also needs the third piece as per the sharing semantic of replicated secret sharing and now P_3 can provide a third piece to him the third piece namely α_3 and in the same way P_1 should have both the first piece and the second piece.

First piece it already has second piece party P_2 can provide and this will complete the replicated secret sharing of the value s it still requires one round, but now the communication for the actual sharing is 3 ring elements compared to 4 ring elements. So, earlier if there was no pre processing data available and at the run time when the value s is available for the dealer to be shared if it directly tries to secret share it as per the replicated secret sharing.

It would have required 4 ring elements, but now assuming that some raw data is available from the pre processing phase the cost of the actual sharing be reduced from 4 to 3 ring elements. Now you might be wondering that how much saving is from whether it is a big saving reducing the communication from 4 ring elements to 3 ring elements. Imagine there are millions of values which the dealer d or the party P_1 would have wanted to secret share during the MPC protocol.

And indeed even for very simple functions the encounter cases where each party has to secret share as a dealer several values, millions of values. So, if there are millions of values which needs to be secret shared then reducing the cost communication cost from 4 to 3 ring elements is indeed a big saving because we are now saving one million ring elements in terms of communication at the actual time of sharing the protocol.

Of course, you have to pay the price of the pre processing phase, but remember we are assuming that the pre processing could have been done in advance when actual circuit, actual function is not yet ready for computation for evaluation. Now why this modified secret sharing protocol is still private like the earlier protocol? So, again we are taking the case when P_1 is the dealer here.

So, if the dealer is honest then we have to argue that if P_2 or P_3 is corrupt then its view is independent of the actual secret. So, imagine if P_2 is corrupt if P_2 is corrupt then what will be its view? Its view will be α_2 because that is s_2 and α_3 which is s_3, but α_2 and α_3 they are completely independent of the value s which is actually shared because this α_2 and α_3 would have been generated in the pre processing phase.

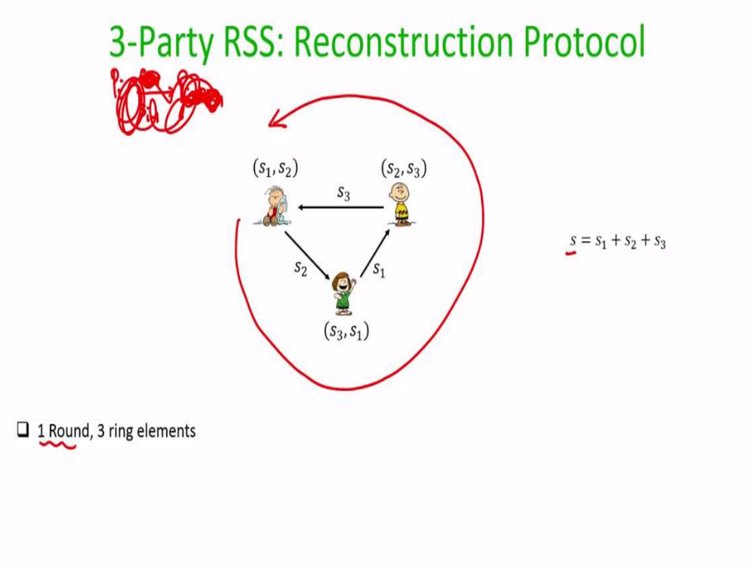
And it has absolutely no relationship with the secret s. So, that is why a corrupt P_2 does not learn any information whereas if we consider the case where P_3 is corrupt its view will be s_3 which is α_3 and s_1 which actually has something to do with the secret, but s_1 is actually s + s, α_1 and α_1 is a random element α_1 is actually acting as here a onetime pad key and s_1 is acting as the one time pad encryption.

Now since the pad α_1 is not known why the pad α_1 is not known to this potentially corrupt P_3? It knows α_3 fine, but it does not know α_2 and α_1. Even though it knows that this sum up to 0 so it could be any α_1 and corresponding to that α_1 there is an α_2 which would have summed up along with α_3 to be 0 and now that α_1 is acting as the pad and he is getting s + α_1 so that would not reveal any information.

Now remember next time again the dealer wants to secret share a values say if P_1 again wants to share a values say s'. We need a fresh pre processing data to be available. So, we cannot use the same α_1, α_2, α_3 and let the dealer use the same α_2, α_3 set up for sharing all the values which it wants to secret share. For every value s which it wants to secret share there has to be an independent.

There will to be a fresh α_1, α_2, α_3 which has to be made available from the pre processing phase. So, you can imagine that if in the sharing phase there are billions of values which needs to be secret shared then in the pre processing phase we need this setup of α_1, α_2, α_3 billion number of times to be generated.

(Refer Slide Time: 34:58)



3-Party RSS: Reconstruction Protocol

$s = s_1 + s_2 + s_3$

- 1 Round, 3 ring elements

Now, let us see the reconstruction protocol. Assume that we have a value s which has been shared in the replicated secret shared fashion and now we want to publically reconstruct it. Now to publically reconstruct it we need to ensure that all the three pieces are available with every party. Right now every party will just have two pieces. So, the way we ensure that each party has all the 3 parties have three pieces is the following.

We ask P_i to sent s_{i+1} to the next party next party in the sense along the circle. So, you imagine a circular arrangement here where P_1 is followed by P_3, P_3 is followed by P_2, P_2 is followed by P_1 and of course you can change the circular order as well that is not important. Once we fix the circular order then we have to follow the protocol as per that circular order itself.

So, we can ask P_1 in the protocol to send the piece s_2 to P_3 that will ensure that P_3 have now all the three pieces. We ask P_3 to send s_1 okay to P_2 and actually it should be P_i to P_{i-1} because 3 is sending the piece number s_{i-1}. Of course there can be some other ways also because if I change the circular order then that would have followed what I have written down earlier.

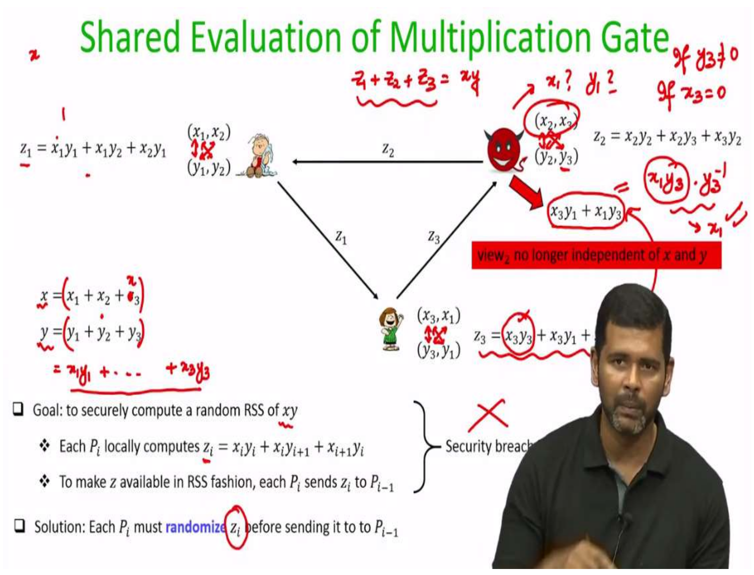
But for the demonstration I am assuming this order. So, P i sends the piece s_{i-1} to the i – 1th party where if I take a P_3 to be P_i then it is sending the piece s_1 so again this is not a P_{i-1} so let us not go into the formal thing here. We can come up with exact formal thing, but you can imagine that the parties are arrange here in the circular order and we follow the circular notation in terms of communicating the messages that P $_1$ should send P $_2$ to s $_3$ P $_3$ should send s $_1$ to P $_2$ and P $_2$ should send s $_3$ to P $_1$.

And that will ensure that all the 3 parties will have all the three pieces and then they can reconstruct back the secret. So, now the nice thing about this reconstruction is that each party now needs to talks to only one party. It is not the case that every party has to send its share to every other party for the full public reconstruction. Each party just need to talk to one of its neighbor.

And we can fix that one neighbor as per our circular order in the protocol. So P 1 we can fix that it talks only to P $_3$, P $_3$ talks only to P $_2$, P $_2$ talks only with P $_1$ and this can happen in parallel so that is why it takes one round of communication and total 3 ring elements are communicated and this nice feature that each party talks only to one party was also followed in the efficient secret sharing protocol that we had discussed earlier.

In that efficient secret sharing protocol also the dealer who was P $_1$ was talking only to P $_3$. P $_3$ was talking only to P $_2$ and P $_2$ talks only to P $_1$. It is not the case that everyone is talking with everyone or interacting with everyone. So, that makes this hearing protocol very nice and we will see that we will maintain this nice property of circular communication where each party just sends or communicate to only one of its neighbors during the circuit evaluation protocol as well.

(Refer Slide Time: 39:08)



Shared Evaluation of Multiplication Gate

- Goal: to securely compute a random RSS of xy
  - Each P_i locally computes $z_i = x_iy_i + x_iy_{i+1} + x_{i+1}y_i$
  - To make z available in RSS fashion, each P_i sends z_i to P_{i-1}

  } Security breach

- Solution: Each P_i must **randomize** z_i before sending it to to P_{i-1}

So, now we had seen the secret sharing protocol, we had seen the reconstruction protocol, we know that this replicated secret sharing satisfies the linearity property. Now what is left is to see whether we can evaluate the multiplication gates also in a secret shared fashion assuming that the gate inputs are secret shared as per replicated secret sharing. So, consider a scenario where you have a multiplication gate in the circuit and the inputs of the gates are x and y.

And this could be any intermediate gates in the circuit so we are considering the setting when x and y are secret shared as per the replicated secret sharing and now we want to ensure that the output Z which is x dot y should also be somehow made available to the parties in a replicated secret shared fashion and we will be interested to see whether this can be done non interactively because we will prefer a method where all the gates can be evaluated in a shared fashion.

And that too non interactively because that possible that means that once all the function inputs are secret shared then parties do not need to interact. They can just keep on evaluating their local copies of the circuit gate by gate by gate and go all the way to the output gate and then reconstruct a shared output, but it turns out that for multiplication gate it is not possible here to do it non interactively.

So, the goal of the parties is to compute replicated secret sharing of x times y and x times y if I consider then basically that will be the product of $x_1 + x_2 + x_3$ and $y_1 + y_2 + y_3$ and then this can be written down as summation of 9 terms here 9 summands here. So, sorry for this typo this should be x_3 so there are 9 summands here starting from $x_1 y_1$ going all the way to $x_3 y_3$.

Now, we have to see whether there is a mechanism by which we can finally write down x times y as the summation of three pieces and each of these 3 parties having two of those three pieces because that is the sharing semantic of replicated secret sharing. Now, with that goal in our mind let us propose the following. Let each P_i computes this term Z_i. So, what does this mean? That means the following.

So P_1 it can compute x_1 times y_1 it can compute x_1 times y_2, it can compute y_1 times x_2. Of course it can also compute x_2 times y_2, but we do not want to have one party computing more summands compared to other parties remember we have 9 summands here and we want to distribute 3 summands to each of the parties. So, 3 summand should be one of the parties another 3 summands with another party and so on.

So, P 1 can take care of these 3 summands he can sum them locally and he obtain Z_1 .P_2 can compute this summand and P_3 can compute this summand. So, it is easy to see that indeed the summation of Z_1, Z_2 and Z_3 that is equal to x times y because of the arithmetic and now you might be tempting to do the following to ensure that the value x times y is made available in a replicated secret shared fashion because remember we will be finally evaluating the entire circuit where the values will be secret shared as per the semantics of replicated secret sharing.

So, if you know want to ensure that x times y is available in a replicated secret shared fashion you might be tempting to do the following. Ask P_1 to send Z_1 to P_3 that will ensure that P_3 has both Z_3 and $Z_1$and in the same way ask P_3 to send Z_3 to P_2 that will ensure that P_2 has both Z_2 and Z_3 and so on. Now the question is will this protocol lead to a security breach.

Remember our goal is to ensure that when we are evaluating the multiplication gates nothing about the inputs x and y should be revealed. So, before the multiplication gate evaluation started what information about x and y was available with the adversary. So, for simplicity assume that P_2 is the corrupt person adversary. So, before the multiplication gate started P_2 has no information about x_1 it has no information about y_1.

Both of these two pieces are random from its view point, but now if I ask P_3 that you give Z_3 to P_2 so Z_3 is this value. Now out of these value P_2 already have x_3 ,y_3 so it can subtract it from Z_3 and hence that will reveal this value x_3 times y_1 and x_1 times y_3. Now, you might be wondering that okay this does not reveal the full value of y_1 or x_1, but what we can now say is that the view of the corrupt party the party number 2 is no longer independent of x and y because now it is learning an additional information for x_1 and y_1 which was not available earlier.

Specifically consider the case where say $x_3 = 0$ it could be poss1 i1.

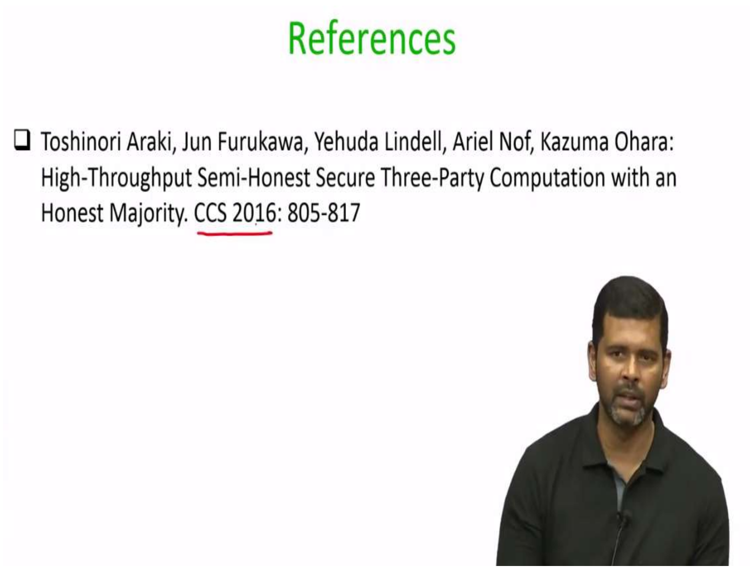
Now if it has x_1 then along with x_2 and x_3 it reveals the full x to the corrupt P_2 that means this protocol definitely is not secure. Even though allowing asking the parties to compute Z_1, Z_2, Z_3 produces three pieces whose sum is equal to xy, but as soon as we ask the parties that okay you interact these three pieces with each other to ensure that the final thing is available in a replicated secret shared fashion the security is breached.

And the problem here is something very similar to what we faced during the degree reduction problem. There also we saw that if x was secret shared through a t degree polynomial y was secret shared through a t degree polynomial and if the parties just locally multiply their respective shares of x and y not only it leads to the blow up in the degree, but it also causes security breach because the resultant x times y is no longer shared through a random 2t degree polynomial.

And the same issue is coming here so now what we have to do is, but this protocol is very nice because it has this property that each party just talks with only one of the neighbors and remember our whole motivation is to get a efficient 3PC protocol where we would like that each party talks only once for every multiplication gate and that too only with one of the neighbors.

So, we would like to retain this blueprint, but to prevent the security breach what we have to do is we have to ensure a mechanism where party P should randomize the piece Z_i before sending it to the neighbor.

(Refer Slide Time: 47:36)



References

- Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, Kazuma Ohara: High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. CCS 2016: 805-817

Now how exactly that randomization will happen that will be the focus of our next lecture. So, the 3PC protocol based on replicated secret sharing that I discussed today in today's lecture is based on this paper published in 2016. Thank you.