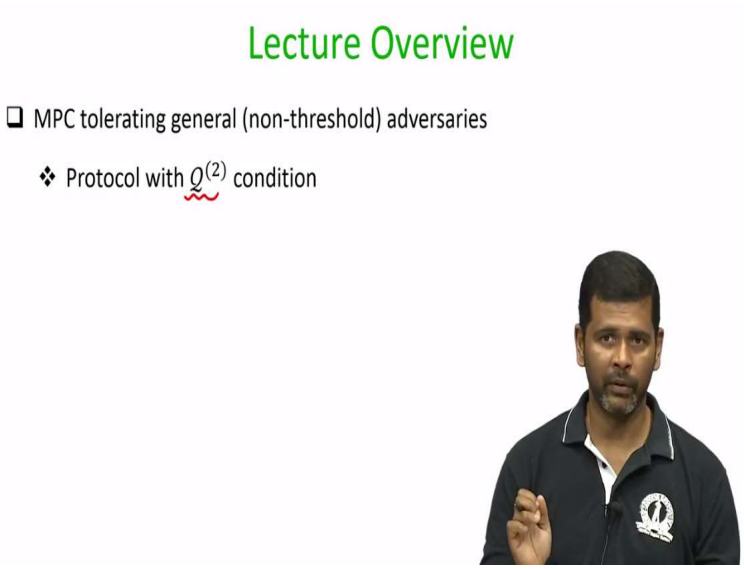
Secure Computation: Part 1
Prof. Ashish Choudhury
Department of Computer Science
Indian Institute of Science – Bengaluru

Lecture – 30
Perfectly-Secure MPC Tolerating General (Non-Threshold) Adversaries with Q^((2)) Condition

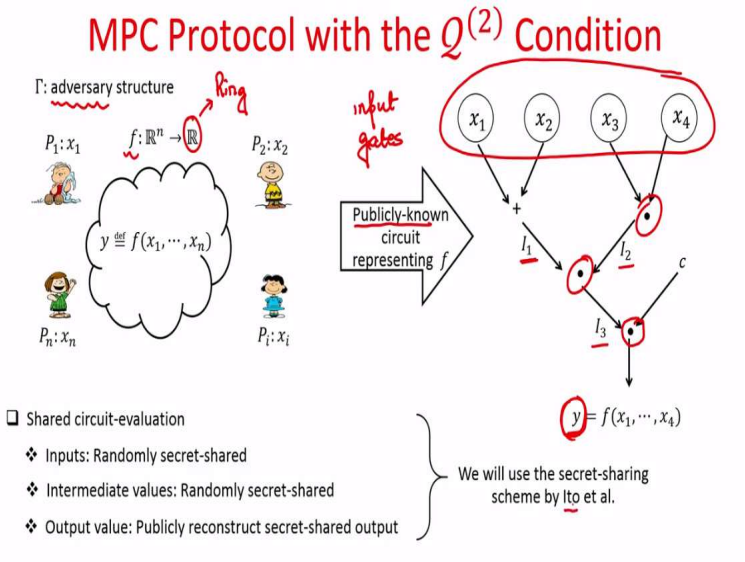
Hello everyone. Welcome to this lecture.

(Refer Slide Time: 00:31)



So, in this lecture we will see an MPC protocol tolerating non threshold adversaries where it will be ensured that the Q_2 condition is satisfied that means you will be given an adversary structure with respect to which the set of parties satisfies the Q_2 condition.

(Refer Slide Time: 00:54)



So, we will assume that we are given a function f over some algebraic structure and algebraic structure in this case it is sufficient even if it is a ring. So, unlike the protocols in the threshold setting where you require the underlying function to be a function over a field we do not require the circuit to be or the function to be over a field. It is sufficient even if the function is over a ring.

That means in terms of algebraic structures also designing MPC protocols against non threshold adversaries provides you better flexibility in terms of the adversary structure here and following the blueprint of shared secured evaluation which we have seen in the threshold setting we assume that the function which the parties want to securely compute is abstracted by a publically known circuit over this ring which will have your input gates.

And then you will have non-linear gates, you will have multiplication by public constant and then you have the final output. Again without loss of generality we will assume that each party has a single ring element as the input for the function, there is a single function output to be publically learned by everyone and a function is the deterministic function. Again all this are without loss of generality.

And the idea behind the MPC protocol since it is a generic MPC protocol is actually to do the shared circuit evaluation. Namely, we will start with the input gates and we will ensure that all the

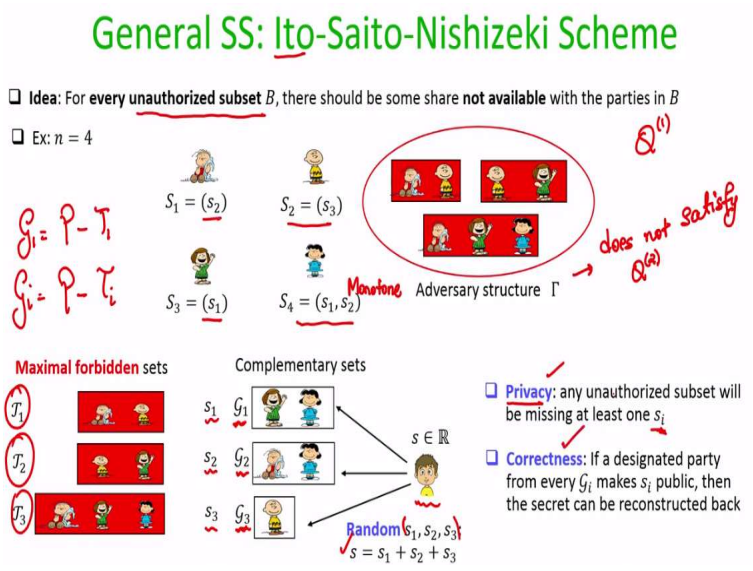
respective inputs of the parties they are secret shared randomly then the results of the intermediate computation namely the values I_1, I_2, I_3 they would not be available and clear.

And somehow it will be ensured that starting with secured shared gate inputs the parties obtain secret shared gate outputs where the gate outputs are secret shared randomly and once we have the function output available in a secret shared fashion the parties go and publically reconstruct. So, the same blueprint what we have followed for the threshold setting intuitively the privacy of the entire computation is ensured because each value in the computation right from the input stage all the way to the output stage or output gate are available in a secret shared fashion.

And they will be secret shared in such a way that even if adversary controls a subset of parties from the adversary structure the shares learned by the corrupt parties will be independent of the actual shared value, but now we have to use secret sharing schemes against the non threshold adversary because we are now modeling the corruption capability of the adversary by an adversary structure which is publically given.

So, we can use any secret sharing scheme which is linear, but we will follow the linear secret sharing scheme due to Ito et al.

(Refer Slide Time: 04:37)



So, let us quickly recap the secret sharing of Ito et al and the idea they are was the following. You do the secret sharing in such a way that or you split your secret into so many pieces and distribute the pieces in such a way that it is ensured that for every potential subset from the adversary structure there is at least one piece which is random from the view point of that subset of parties and that subset of parties is missing that piece.

That means that piece could be any element from your underlying algebraic structure which will ensure that any subset of parties from the adversary structure fails to find out what exactly is the underlying value which is secret shared. So, how this is ensured again for demonstration assume that this is the adversary structure given to you and what we do here is this adversary structure does not satisfy the Q_2 condition that is fine because I am now just demonstrating the secret sharing scheme by Ito.

The secret sharing scheme of Ito requires the Q_1 condition to be satisfied so just for the sake of recap I am demonstrating with this particular adversary structure, but when we will be designing the MPC protocol we will be instantiating the secret sharing scheme of Ito et al assuming that the adversary structure satisfies the Q_2 condition. So, what exactly is the way to do the secret sharing here?

We list down the subsets in our adversary structure and remember the adversary structure is monotone namely it is the collection of maximal forbidden subsets that means any subsets of those subsets any proper subset of those subsets are also by default elements of the adversary structure, but they are not explicitly listed down and now with respect to this forbidden set, maximal forbidden sets we can find out the corresponding complementary sets.

So, g_1 is equal to the set of parties minus the first potential forbidden set and in general g_i is equal to the set of parties and from that I subtract the ith potential forbidden set and so on. Now, the dealer who want to secret shares its value it will know the details of the forbidden sets, it will know the details of the complementary sets so it will know how many complementary sets are there.

In this example there are three complimentary sets, but if assume there are k such complementary sets if your adversary structure has k subsets. So, what the dealer does is the following. It generates three random shares of the secret such that the sum of those three pieces is the secret s that ensures the randomization in the secret sharing that means every time dealer wants to secret share the same input s.

It will be picking s_1, s_2, s_3 independently it would not be picking the same s_1, s_2, s_3 again and again and now once it has picked s_1, s_2, s_3 the piece s_1 is given to all the members of group g_1 over the secured channel, the piece s_2 is given to all the members of the group g_2 over the secret channels and the piece s_3 is given to all the members in the group g_3.

Now, what will be the overall share for each party? The overall share for each party will be all the pieces that party obtains based on how many groups it is present in. So, for instance, this first party P_1 it is present only in the group g_2 so that is why its overall share is only the piece s_2. The second party 2 is a member of only g_3 so that is why its share has only one element.

The third party is a member of only the group g 1 that is why its share is only one element from the ring and the fourth party is present in two groups g_1 and g_2 that is why its overall share is s_4. The privacy property it is easy to see here or argue here if you take any subset from the adversary structure that means if it is say the set T_1 or T_2 or T_3 which gets under the control of the adversary.

Then corresponding to that subset there is some piece s_i so if the subset T_i gets corrupt by the adversary then corresponding to that there is a missing piece s_i and then since the missing piece s_i is a random element it could be any element from the ring and hence it could be the case that any value from the underlying ring has been secret shared by the dealer that is intuitively the privacy argument.

Now for correctness if you want to reconstruct back the secret what we can do is the following. There are several ways to do that in the context of MPC protocol imagine there is a value s which has been secret shared in this form namely the members in g_1 has the piece s_1, the members in the

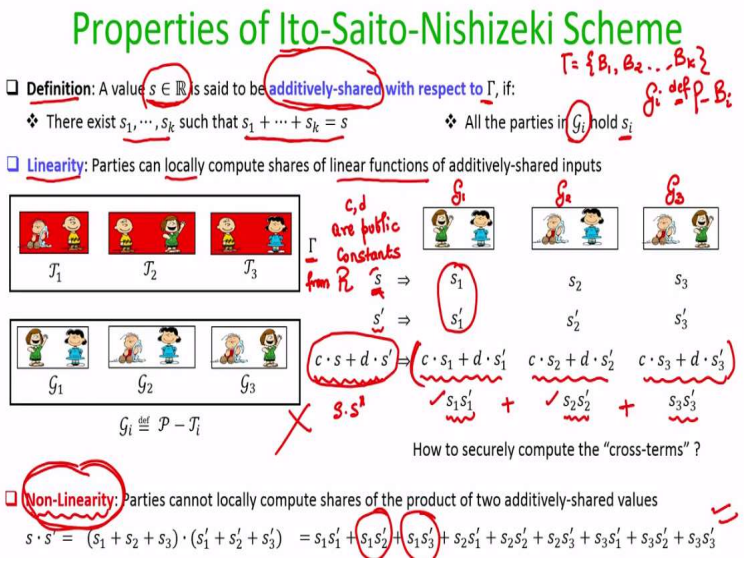
group g_2 has the piece s_2 and the members in g_3 has the piece s_3 and there is a requirement that the secret has to be publically reconstructed.

If it is supposed to be publically reconstructed then we need s_1, s_2 and s_3 and then we can add them and get back the secret s. How we can get s_1? Well we can ask any member from g_1 not both the members because g_1 might have more than one member. So, we can ask any member from g_1 ask in the sense it can be designated as part of the protocol itself that okay if g_1 has these members.

Say, for instance, focus on the least indexed party and that least indexed party is supposed to make s_1 public that could be the protocol code. Similarly, the least indexed party in g_2 is supposed to make s_2 public, the least indexed party in g_3 is suppose to make s_3 public and so on. If this is the protocol code then s_1, s_2, s_3 will be made public and then once they are made public these pieces can be added and the secret s can be reconstructed back.

So, that could be the reconstruction protocol. So, this very nice secret sharing scheme just based on set theoretic properties satisfies both the privacy condition as well as the correctness condition.

(Refer Slide Time: 11:59)



So, now we will see some nice properties of secret sharing scheme due to Ito et al. So, let us first introduce the definition here. I will say that a value s from the ring is additively shared with respect to the given adversary structure gamma if the following holds. Assume that your adversary

structure gamma has subsets B_1, B_2 up to B_k I will say that the value s is additively shared with respect to gamma.

If there exist k pieces s_1 to s_k from the ring such that their summation is equal to s and all the parties in the group g_i where the group g_i is defined to be the difference of the ith potential that subset from the set of parties. So, all the parties in the group g_i should hold the value s_i. If this arrangement has been done I will say that a value s is additively shared with respect to your adversary structure.

It is like saying the following for the threshold setting we continuously use the term that a value is Shamir secret shared by that we meant that there is some t degree polynomial whose constant term is that secret and every party P_i has the value of that polynomial at α_i. If that arrangement has been done then we use the term that the value is Shamir secret shared. I am just trying to use similar term.

But now since we are going to use the secret sharing scheme of Ito et al our nomenclature for secret sharing or the semantic of the secret sharing will be different. Our semantic of the additive secret sharing is the value is called secret shared if every member in g_i has a piece s_i and if we sum all the pieces s_1 to s_k namely from the group g_1 take the piece s_1, from the group g_2 take the piece s_2 from the g_k take the piece s_k. The summation of those key pieces should be equal to s.

If that distribution of information has been done then we will say that value s is additively secret shared. Now it turns out that this scheme of Ito et al satisfies the linearity property and if it satisfies the linearity property then it turns out that parties can non interactively compute shares of linear functions of a secret shared inputs or secret shared values. So, let me demonstrate that assume this is the adversary structure.

And in this adversary structure you are given three subsets, three potential subsets they are maximal forbidden subsets and with respect to each subsets I have computed the corresponding group g_1, g_2, g_3. So, imagine a value s is secret shared as per additive secret sharing with respect to this adversary structure that means for s you have three pieces s_1, s_2, s_3 all the members in g_1 has s_1, all the members in g_2 has s_2, all the members in g_3 has s_3.

And similarly you have another value s' which is additively shared that means you have three pieces s_1', s_2', s_3' all the members in g_1 have s_1', all the members in g_2 have s_2' and all the members in g_3 have s_3' and say c, d are public constants from the ring R it is known to everyone. Now, if the parties non interactively compute the following.

That means each member in g_1 computes this value, each member in g_2 computes this value, each member in g_3 computes this value and they can compute this value non interactively, it does not require any interaction then altogether this vector of three values now can be treated as the shares for the value c times s plus d times s' as per additive secret sharing that is what I mean by the linearity property here.

It does not require any interaction among the members of g_1, g_2 or among the members of g_1, g_3 and so on to compute a secret sharing of c times s plus d times s'. It can be done completely in the non-interactive way. However, again the bottleneck is the non linearity property here. We face similar issue for the case of Shamir secret share as well. Shamir secret sharing it allows to compute linear functions of secret shared inputs.

That means if the inputs are secret shared then every party can compute their respective shares of the output as well by applying the linear function on the shares of the input, but when it comes to the non-linear gates we saw that we have to solve the degree reduction problem and so on. In the same way for the case of Ito et al the scheme by Ito et al the schemes satisfies the linearity property nice linearity property, but it does not have the non linearity property.

So, for instance, if s is secret shared and s' is secret shared then s times s' can be written like this and if I expand s times s' then that expansion will have several summands here, in this example it has 9 summands. Now, if at all we want s times s' to be available in a additive secret shared fashion we would require the members of g_1 to hold something.

We would require the members of g_2 to hold something and we would require the members of g_3 to hold something such that collectively if we add them we should get back the value s times s'.

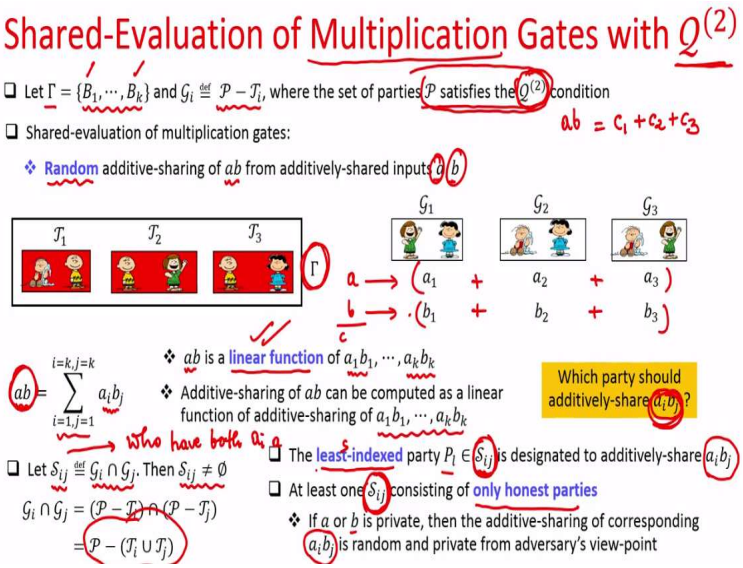
The members in g_1 they can compute this value fine because they have both s_1 as well as s_1'. The members in g_2 could compute s_2 times s_2', the members in g_3 could compute s_3 times s_3', but what about the other terms?

We cannot say that add these three things and it gives you s times s' that is not the case you have other cross terms like s_1 times s_2' s_1 times s_3' and so on. We cannot ask the members in g_1 that okay you give s_1 to the members in g_2 and we cannot ask the members in g_2 that okay give s_2 and s_2' to the members of g_1 because that will end up breaching the privacy of AND as s'.

When we want to perform computation on s and s' our goal is to compute shares of s times s' without actually revealing anything additional about s and s' respectively that means we are now having difficultly to compute non-linear functions of secret shared inputs. Namely, we will now see how to compute shared evaluation or how to perform shared evaluation of multiplication gates assuming that the set of parties satisfies the Q_2 condition.

We have seen already that we had the non linearity property and just by performing operations on the shares of s and s' the group members g_1, g_2, g_3 cannot afford to have their respective shares of s times s' we have to do something more that means it will require interaction among the parties.

(Refer Slide Time: 20:49)



So, imagine you are given an adversary structure and the set of parties and it is ensured that the set of party satisfies the Q_2 condition remember that is a necessary condition and with respect to the ith potential that subset we have the group g_i. So, let us first see what exactly is our goal here. Our goal is the following you have inputs a and b which are additively shared as per the definition that we have given in the previous slide.

And we want that at the end of this shared multiplication gate evaluation for the value a, b there should be three pieces, three random pieces that is important c_1, c_2, c_3 such that the summation of c_1, c_2 and c_3 should give you ab that is first condition. All the members in g_1 should have c_1, all the members in g_2 should have c_2 all the members in g_3 should have c_3 that is the second condition.

And the third condition is no additional information about the inputs a and b should be revealed in this process. Namely, similar requirements that we had for the degree reduction problem, in the degree reduction problem remember your a was shared through a t degree polynomial, b was secret shared through a t degree polynomial and our goal was to compute the shares of a times b lying on a random t degree polynomial.

And in the process not revealing anything additional about the input say a and b that was our requirement from the solution of the degree reduction problem. Now we are generalizing that requirement in the context of additive secret sharing of Ito et al and the idea here is the following. We know that since a is secret shared and b is secret shared as per additive secret sharing then I can rewrite a times b as summation of several summands.

Namely, the summands a_i times b_j where i ranges from 1 to k and j ranges from 1 to k. Namely, we have k number of potential bad subsets and a was divided in k pieces, b was divided into k pieces. So, if a is secret shared like this, b is secret shared like this and our goal is to compute c which is a times b then that is equivalent to saying that we want to sum up all these pieces, we want to sum up all these pieces and we multiply.

So, if we expand then basically we get this formula that means ab is a linear function of this k square summands why k square because i ranges from 1 to k, j ranges from 1 to k. In this example

we have 9 summands starting from a_1 b_1 all the way to a_3 b_3. So, the thing is ab can be written as summation of this k square summands and summation is basically a linear function.

And we know how to compute linear function in a secret shared fashion if the inputs of the linear function are secret shared that means if we ensure that each of this inputs, each of this summands a_1 b_1, a_2 b_2 a_i b_j. If each of these summands a_i b_j is secret shared by some party we will see who is going to do that and if all this k shares summands are available in secret shared fashion then basically computing shares of ab is equivalent to adding shares of each of this summands.

So, additive sharing of ab can be computed as a linear function of additive sharing of this k square summands, but now the question is who is going to additively shared a_i times b_j, a_i might be available with some group b_j might be available with some group. Is there a possibility that there is some common party who has both the piece a_i as well as the piece b_j because if there is a party who process both a_i as well as b_j then it will know the value a_i times b_j which is one of the summands in this big sum formula.

And that particular party can secret share a_i times b_j by acting as a dealer. So, if we can ensure that for each of this k square summands there is some party who has the capability to compute that summand in clear and then secret share it then our problem is solved. So, let us see whether for each of the summand a_i times b_j there is some party who can compute a_i times b_j without interacting with anyone.

And if it can then it can just compute a_i times b_j and secret share it. So, for that let me introduce this notation S_{ij} and remember we are doing this task for each summand a_i times b_j where i ranges from 1 to k and j ranges from 1 to k. We are not just doing for one summand a_i times b_j. So, we are actually analyzing the possibility that for this k square summands there is some party who can compute it.

So, let S i, j be the set of all parties who can compute a_i times b_j namely $S_{i,j}$ is the set of parties who have both a_i as well as b_j as per the secret sharing of a and b. Namely, it is the intersection of the ith group g_i and the jth group g_j. My claim is that if the set of parties P satisfies the Q_2 condition

then this set $S_{i\,j}$ is non empty there is definitely one party at least one party who has both a_i as well as b_j.

And again this comes from simple set theoretic properties. What is the intersection of group g_i and group g_j basically that is the intersection of these two sets and then if I expand it further this is the difference of union of ith bad set and the jth bad set from the set of parties. Now, if this set if this difference is phi empty then that is the violation of the Q_2 condition that means we now have two bad sets T_i and T_j.

In your adversary structure gamma whose union covers the entire set of parties P if the difference would have been 0 or empty not 0 if the difference would have been empty set, but since we are assuming that the set of parties P satisfies the Q_2 condition that means this intersection of g_i and g_j which is equivalent to the difference of union of the bad sets i and j from the set of parties is definitely non empty.

There is at least one party in the set $S_{i,\,j}$ and this is true for any i and any j in the range 1 to k. So, now we know that for each of the summands that means you take a_1 b_1 there is at least one party who has both a_1 as well as b_1 so that party can just secret share a_1 times b_1. If I take a_1 times b_2 there is definitely one party which has both a_1 as well as b_2 so it can compute a_1 times b_2 and secret share it and so on.

But now the problem is this set $S_{i,\,j}$ need not be a single ton set there might be several members in $S_{i,\,j}$ it depends upon your exact adversary structure and the corresponding groups. So, now what we can do is to ensure that the piece a_i b_j is not secret shared multiple times because remember we just need to ensure that the summand a_i times b_j is secret shared once that is all.

We do not want a_1 b_1 to be secret shared multiple times and keep on adding all those shares of a_1 b_1 because that would not give ab. So, to ensure that for each summand a_i times b_j just a single party secret shares a_i times b_j we designate the least indexed party say P l in that set to additively share times a_i times b_j by acting as a dealer and executing an instance of Ito et al secret sharing scheme.

Of course if $S_{i,j}$ is a single ton set then there is no question of bringing the least indexed party would not come, but if $S_{i,j}$ has more than one party then definitely we have to ask only one of the parties to secret share a_i times b_j and basically we are following the rule that we will assign the least indexed party and identity of the least indexed party will be publically known because the description of $S_{i,j}$ will be publically known.

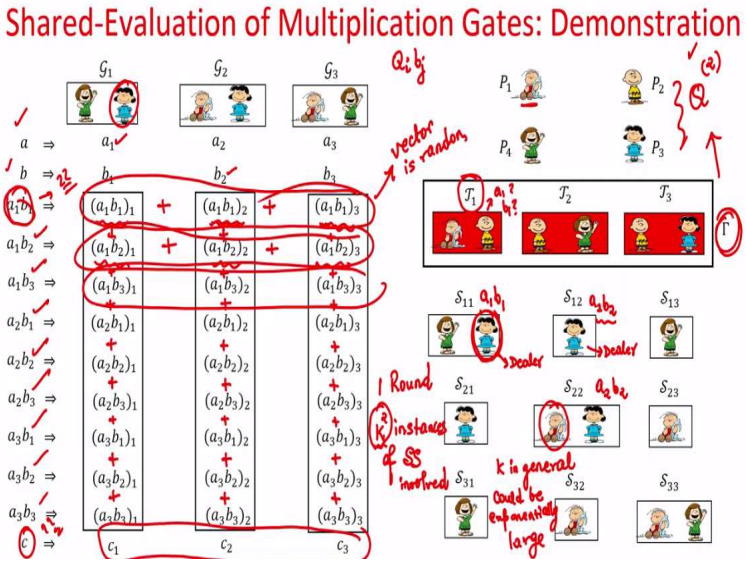
Remember the set of parties adversely structure the groups everything are publically known. Also the claim is that at least one $S_{i,j}$ will be there consisting of only honest parties that means during the execution of the MPC protocol it could be either the set B_1 or the set B_2 or the set B_k which can be under the control of the adversary. The corresponding group g_i will be completely honest.

And if I focus on the set $S_{i,i}$ and it will consist of only honest parties and if it consist of only honest parties then the corresponding summand which is assigned to that corresponding subset $S_{i,j}$ will be unknown from the view point of the adversary. So, that means if a or b the inputs of the multiplication gate are private to begin with then the corresponding additive sharing of a_i times b_j which the least indexed party in the set $S_{i,j}$ will be performing or will be doing will be random and view point from adversary's view point.

And $S_{i,j}$ will be consisting just honest parties because we know that at least one such $S_{i,j}$ is always there. So, I am focusing on that $S_{i,j}$ which consist of only honest parties my claim is that the parties in the $S_{i,j}$ are suppose to take care of this summand a_i times b_j. The least indexed party from this $S_{i,j}$ consisting of only honest parties will secret share a_i times b_j, but since a_i as well as b_j are random from the view point of the adversary.

A_i times b_j will be random from the view point of the adversary and hence the least indexed honest party from this set $S_{i,j}$ who will be secret sharing a_i times b_j will be random from the view point of the adversary and that will ensure that the overall ab which is finally obtained at the end of the shared multiplication gate evaluation is randomly shared from the view point of the adversary.

(Refer Slide Time: 33:25)



So, let us see a demonstration here for the shared evaluation of the multiplication gate here. So, I am taking this adversary structure and the set of parties consist of four parties it is easy to see that this set of parties satisfies the Q_2 condition because you take the union of any two bad subsets from the adversary structure it does not cover the entire set of four parties and imagine a and b are already is secret shared as per the additive secret sharing.

So, the first thing is remember we have k square summands we have to identify that who is supposed to take care of the summand a_i times b_j. Here we have 9 summands starting from a_1 b_1 and going all the way to a_3 b_3. So, I have written down for each summand a_i times b_j I have written down the subset of parties who have the capability to compute the term a_i times b_j in clear.

So, for instance, S_{11} will consist of all the member in group g_1 because they have both a_1 as well as b_1. If I focus on the set S_{12} basically I am focusing on the set of parties who have the capability to compute a_1 times b_2. Now who can compute a_1 times b_2? So, P_1 alone cannot compute a_1 times b_2 because P_1 this first party it does not have a_1 and it has only b_2, but it does not have a_1.

So, that is why it is not a member of this set S_{12}, but if this take this party and it has both a_1 as well as b_2 and it is the only party who has both a_1 and b_2 that is why it is listed in S_{12} and so on. Similarly, S_{13} will have only one member because there is only one party she is the only party who has both a_1 as well as b_3 and so on. So like that I have listed down here the various subsets here.

And now from S_{11} we can assign the least indexed party namely P_3 to take care of a_{11} because both P_4 as well as P_3 could have secret shared a_1 times b_1, but we do not want a_1 times b_1 to be secret shared twice so that is why we are asking only one of the parties to secret share a_1 times b_1 which is the least indexed party S_{12} is anyhow single term, S_{13} is anyhow single term.

Now, again if I come to S_{22} both P_1 as well as P_3 has the ability to secret share a_2 times b_2, but we are going to assign P_1 this task because it is the least indexed party and so on. So, now here is how the multiplication gate will be evaluated. So, we will start with P_1 because it is the least index not P_1 we will start with P_3 which is the least indexed party in S_{11} and it will act as a dealer compute three random pieces for a_1 b_1.

Those three random pieces I am denoting by this notation such that it sums up to a_1 times b_1 and this first piece this third party will give to all the members in g_1 the second piece this dealer so it is acting as the dealer here. So, as a dealer it will give this second piece to all the members in g_2 and as a dealer it will give this third piece to all the members in g_3. In parallel for the second summand a_1 times b_2 this party acts as a dealer.

Create three random pieces for a_1 times b_2 and secret shares it and like that every designated party from each of the group $S_{i,\,j}$ in the same round remember all this secret sharing instances they happen in parallel, it is not that first a_1 times b_1 has to be secret shared and then the next summand has to be secret shared and then the next summand has to be secret shared no there is no dependency whatsoever.

So, that is why each designated party from the set $S_{i\,j}$ will act as a dealer and run a random instance of Ito et al secret sharing scheme to secret share the summand a_i times b_j. Now, once all the k square summands have been secret shared the members in g_1 can do the following. They can just add up all their shares that they have received in all the sharing instances here call that piece as c_1.

In the same way the members in g_2 can add up all the pieces that they have got in the various secret sharing instances in this multiplication gate evaluation process and call the resultant piece as c_2

and let all the members in g_3 at their respective pieces that they have got in the various secret sharing instances during the multiplication gate evaluation let me call the resultant piece as c_3.

And it now is easy to see that if you take c_1, c_2, c_3 then together it constitutes a vector of secret sharing for the value c. Now why the privacy of a and b will be preserved and why the value c is randomly shared here. Imagine during the protocol the set T 1 gets corrupt either set T 1 gets corrupt it might have already a_2 b_2 a_3 b_3 it might already have, but it would not have the pieces a_1 b_1.

So, it would not have the piece a_1 it would not have the piece b_1 they are random from the view point. Now if a_1 b_1 is random and a_1 b_1 is secret shared by a random instance of Ito secret sharing then this first instance here is a random instance here that means this is a vector of three random pieces which sum up to a_1 times b_1 and now if this vector is added with the other vectors which may or may not be random.

But anyhow they are random, but we know definitely this vector is random and that random vector added with all the other remaining vectors will finally produce a vector of values which is also random. So, that ensures c is secret shared through a random vector of shares and now since a_1 times b_1 was not known to the adversary it could be any a_1 times b_1 that means if even if the other pieces they are known to the adversary somehow.

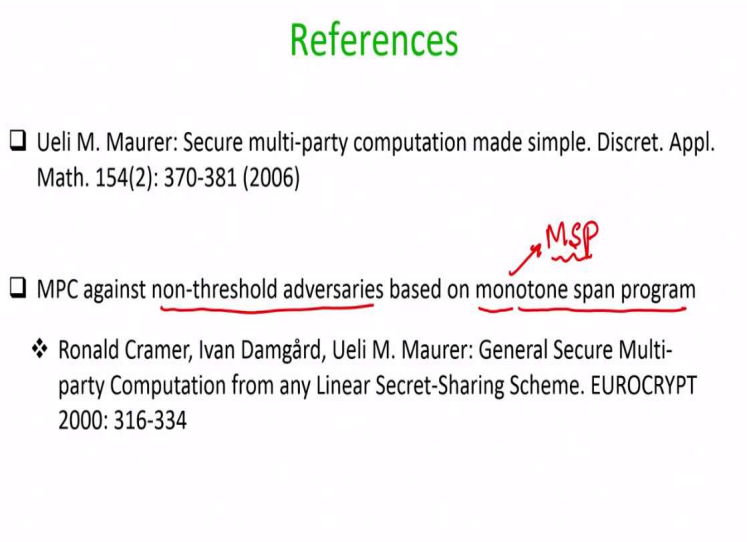
This a times b_1 could be any value and hence this c could be any value that is intuitively the privacy argument. We can formalize it by saying that whatever information adversary is obtaining while the secret sharing of this summand a_i times b_j is happening that can be simulated if a_i times b_j was shared by an honest dealer. So, I am not going into the full privacy proof.

I hope that you are getting the intuition here that why the privacy of a, b and finally c is preserved here. So, that is a way you can perform the shared evaluation of multiplication gates. So, this will require one round of communication because all this summands a_i times b_j has to be secret shared and there will be k square instances of secret sharing involved not threshold secret sharing remember.

All these instances of secret sharing are the instances of Ito et al scheme. Now this k in general so k in general could be exponentially large and that will ensure that the overall communication complexity of your protocol is also exponentially large, but we cannot do anything regarding that because the nature of Ito et al secret sharing is that its share size is proportional to k namely the number of subsets in your adversary structure.

And the number of subsets in the adversary structured could be exponentially large itself. In fact recall when we discuss non threshold adversary structure in the context of secret sharing. One of the main open problems that is left there whether we can design a secret sharing scheme against any given non-threshold adversary structure where the share size is polynomial in number of parties.

(Refer Slide Time: 43:12)



So, these are the references used for today's lecture the multi MPC protocol that I discussed is taken from this paper. This is a very simple paper to understand and read and there is another line of work for designing MPC protocols against non-threshold adversary's which are not based on the additive secret sharing scheme of Ito et al that we had seen in today's lecture, but rather they are based on notion of secret sharing based on monotone span program which are also called as MSP.

But they are slightly advanced to understand and because of interest of time I am not going to discuss the MSP based MPC constructions in this lecture, but if you are interested then you can refer to this paper. Thank you.