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

Lecture – 26
Shared Circuit-Evaluation via GRR Degree-Reduction Method

Hello everyone. Welcome to this lecture.

(Refer Slide Time: 00:31)

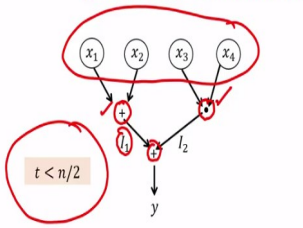
Lecture Overview

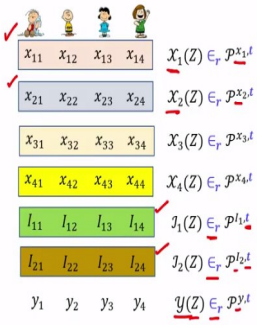
- Shared Circuit-evaluation via GRR Degree-Reduction Method
  - Protocol
  - Demonstration
  - Security analysis

So, the plan for this lecture is as follows. We will see now the full protocol for shared circuit evaluation assuming we have the GRR degree reduction method, we will see the protocol, we will see an example and we will do the rigorous security analysis for the full protocol.

(Refer Slide Time: 00:53)

Shared Circuit-Evaluation with GRR





- Inputs: Randomly (n, t) secret-shared
- Shared gate-evaluation: if gate-input(s) is/are randomly (n, t) Shamir-shared, then gate output is randomly (n, t) Shamir-shared
  - Linear gates: non-interactive
  - Multiplication gates: GRR degree-reduction
- Output: publicly reconstruct the secret-shared output

So, we will assume that we are given a publically known arithmetic circuit over some finite field and we are in the setting where t is less than n over 2 later we will prove that this condition is necessary if you have non-linear gates in your circuit. So, the protocol for shared circuit evaluation is as follows. So, the inputs for the function which are available with the respective parties are secret shared by the corresponding parties who act as the dealer/

And share them by executing instances of Shamir secret sharing. So, in this example we have four parties P 1, P 2, P 3, P 4. P 1 acts as the dealer and it secret shares the value x 1 through a random t degree polynomial denoted by this fancy x 1 polynomial. Similarly, P 2 secret shares its input x 2 through a random t degree polynomial through this fancy x 2 polynomial. Similarly, P 3 secret shares its input through a random t degree polynomial and similarly P 4.

And all this four instances of secret sharing can happen in parallel. So, there is no dependency that P 1 should secret share the input x 1 and then only P 2 go and secret share the input no because all these polynomials fancy x 1, fancy x 2, fancy x 3, fancy x 4 they are picked independently by the respective dealers and now once the inputs are secret shared the parties start interacting to maintain the shared circuit evaluation.

For certain gates maintaining the BGW invariant need not require interaction among the parties, but for multiplication gates it will require interaction and the invariant that the parties have to maintain is the following. If the inputs of the current gate which the parties are now trying to evaluate are randomly secret shared by randomly secret shared I mean that there is some random t degree polynomial whose constant term is actually the respective inputs of the gates.

And parties hold their respective shares on that polynomial. So, invariant that the parties now want to maintain is that if the inputs of the gate are randomly n, t secret shared then somehow they should maintain a vector of n, t random secret sharing of the gate output and in this whole process of the privacy of the gate inputs and the gate output should be reserved. So, linear gates does not require any interaction.

Whereas the multiplication gate we have to run instances of GRR degree reduction. So, in this specific circuit once the inputs are secret shared the first gate that they will consider for evaluation is this plus gate which is a linear gate and what are the inputs of this plus gate?

The inputs are x 1 and x 2. The parties have the respective shares of x 1, their respective shares of x 2.

So, they just locally go and add their respective shares of x 1 and x 2 and collectively they will obtain a vector of shares which will now lie on a random t degree polynomial whose constant term will be the actual output which should have been there if the inputs are x 1 and x 2. So, that means this gate is done now the next gate is the multiplication gate and this is a non-linear gate.

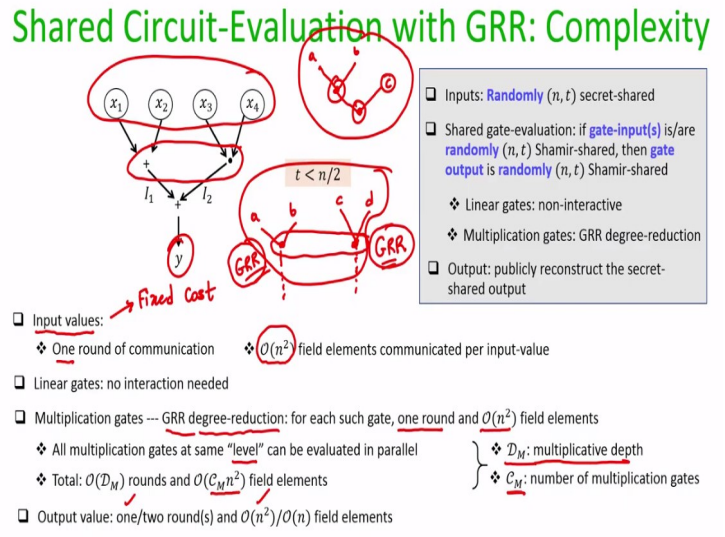
So, the parties have to run an instance of GRR degree reduction so the inputs for this gate are x 3 and x 4. So, they will take these two vector of shares as the input for GRR degree reduction and then they will obtain collectively a vector of shares with Ith party holding the Ith share in the vector and collectively this vector constitute shares on random t degree polynomial whose constant term is I 2.

So that will finish the evaluation of this gate now they go to the next gate in the circuit which is this plus gate which is a linear gate and both the intermediate outcomes I 1 and I 2 are right now secret shared. So, the parties hold their respective shares of I 1 their respective shares of I 2. So, they just have to go and locally add their respective shares of I 1 and I 2 and collectively now they will obtain a vector of shares lying on a random t degree polynomial whose constant tem is y and after this there are no more gates.

So, it is now time to publically reconstruct the function output. So, the parties exchange their respective shares so P 1 sends y 1 to everyone, P 2 sends y 2 to everyone, P 3 sends y 3 to everyone, P 4 sends y 4 to everyone that could be one method of publically reconstructing the function output y or another method could be if you want to save communication then you can do the following let P 1 first reconstruct y for that everyone send their shares of y 2 to just P 1.

P 1 reconstructs this polynomial fancy y gets back the function output y and then it release it to everyone. So, you can follow any of these approaches. So, that is the way our shared circuit evaluation will be performed assuming you have the GRR degree reduction method.

(Refer Slide Time: 06:36)



So, now let us try to analyze the round complexity and communication complexity of the full shared circuit evaluation protocol. So, there is always a fixed cost associated with the input value. So, if there are n parties and they want to secret share their respective inputs there is always a fixed cost associated with that. There is no scope of improvement there it requires one round of communication because every party has to act as a dealer.

Compute shares of its function input and distribute the shares to the respective parties and this can be done in parallels that is why this is a one round of communication and each instance of secret sharing will require order n fields elements to be communicated. So, assuming that there are total n inputs for the function one input coming from each party, sharing the input values will require a total communication of n square field elements.

And if you multiply it with log of cardinality of your field and that will give you the total communication in bits namely the number of bits which are communicated in the entire protocol so that is a fixed cost for any generic MPC protocol then once the party start evaluating the intermediate gates if you encounter a linear gate then that does not require any interaction among the parties.

So, evaluating the linear gates is free in this paradigm and if there is a multiplication gate then the parties have to run an instance of GRR degree reduction and for each instance of GRR degree reduction the parties have to interact once among themselves and in total order n square field elements are communicated. So, for this the students are referred to the previous lecture where we did the analysis of the GRR degree reduction method.

Now in this example circuit we just have one multiplication gate and that is why only one instance of GRR degree reduction is involved, but that need not be the case. You might have a circuit where there are several multiplication gates at several layers and depending upon where exactly you encounter the multiplication gate you have to run instances of GRR degree reduction.

So, it turns out that if you consider an arbitrary arithmetic circuit all the multiplication gates at the same or the same layer can be evaluated in parallel. By same level I mean that there is no dependency among the inputs of those multiplication gates. So, let me demonstrate what I mean by that consider an example circuit where you have a scenario like this you have this one multiplication gate whose inputs are a and b and then you have the rest of the circuit.

And you have another multiplication gate whose inputs are c and d where c and d are different from a and b. So then I will say that these two multiplication gates are independent and hence they can be evaluated in parallel whereas if you have a scenario like this that you have this multiplication gate whose inputs are a and b and then the output of this multiplication gate serves as the input for another multiplication gate.

Then these two multiplication gates are no longer independent and they will be considered to be present at two different layers. Why they are not independent because until and unless I do not get this intermediate result namely the product of a and b in a secret shared fashion even though I have c available in secret shared fashion until and unless the result of product of a and b is not available in a secret shared fashion the GRR degree reduction cannot be invoked.

So, that is why this second multiplication gate can be considered as if it is at a lower level compared to the earlier multiplication gate. So, that is what I mean by saying that if you have independent multiplication gates then they can be imagined considered to be present at the same layer and the GRR degree reduction can be invoked in parallel for all such multiplication gate and as a result for all such multiplication gate.

So, if I consider this example circuit then there are two instances of GRR degree reduction, but both those instances can be executed in parallel that means whatever parties are supposed

to communicate while running the GRR degree reduction for this gate and whatever the parties are supposed to communicate and interact among themselves for the GRR degree reduction or the GRR degree reduction instance for this multiplication gate that can be clubbed together.

And can be sent in one shot that will be sent in the same round that would not be considered two separate rounds of communication. So, that is why in terms of rounds all the instances of GRR degree reduction for evaluating the multiplication gates at the same level can be clubbed together, but communication wise you have to communicate whatever you have to communicate for this GRR degree reduction instance and whatever you have to communicate for this GRR degree reduction instance.

So, in total number of rounds which are required for evaluating the multiplication gate will be proportional to the multiplicative depth of your circuit. So, this D of M denotes the multiplicative depth of your circuit. So, for example, if this is my circuit then the multiplicative depth is 2 whereas this is the circuit then the multiplicative depth is 1, the multiplicative depth of this circuit is 1 and so on.

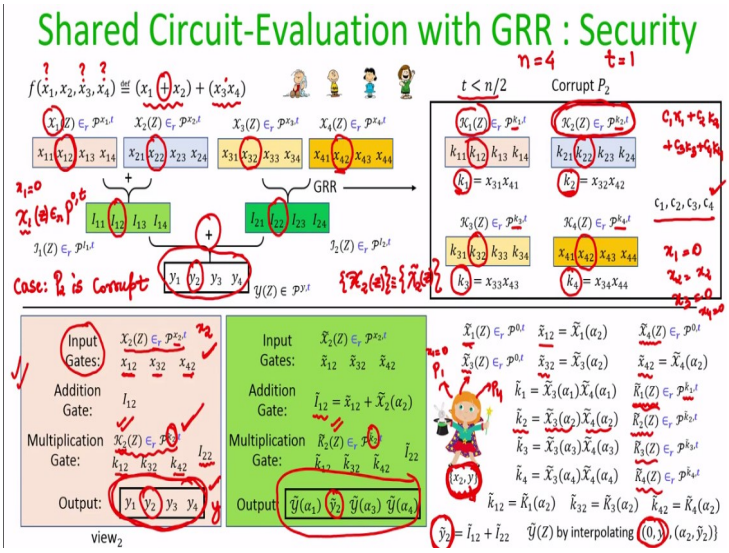
I stress it is a multiplicative depth not the entire depth of the entire circuit. So, depending upon how many independent layers of multiplication gates you have that many rounds of interaction will be required if I count the total number of rounds of interaction which are involved in all the instances of GRR degree reduction whereas the total amount of communication which is required across all the instances of GRR degree reduction is n square times the number of multiplication gates in the circuit.

So, this C M will be a parameter which will be publically known D M also will be publically known because the parties know the full description of the circuit and then when it comes to the output value again that is a fixed cost the parties have to publically reconstruct the function output and then again you have two approaches. If you want to save round then ask every party to make public its respective share of y.

So that will require n square field elements communication or if you want to save number of rounds then let a designated party first reconstruct the function output and then it send to

everyone that will require two rounds and order n field elements in communication so that is a overall complexity analysis.

(Refer Slide Time: 14:06)



Now we have to argue about the security of this entire protocol. So, again for demonstration purpose I will take this candidate function, but now you can imagine that there is a huge circuit and which is evaluated as per the shared circuit evaluation whatever I am saying that can be generalized for any arbitrary circuit, but for demonstration purpose I will consider this example circuit.

And we will be in the setting where t is less than n over 2 and the value of t is publically known. So, again let me quickly go through what exactly are the values which are computed communicated. The inputs will be secret shared through random t degree polynomials and then this plus gate will be evaluated locally and then this multiplication gate is there which will require an instance of GRR degree reduction.

So, if I further go into the details of the instance of GRR degree reduction as part of the GRR degree reduction the parties would compute the publically known Lagrange interpolation coefficients and component wise each party will compute the product of its shares of x 3 and x 4. So, if k 1, k 2, k 3, k 4 together will lie on a polynomial of degree 2t and that will be a non-random polynomial.

So, we have to now convert this vector of shares k 1, k 2, k 3, k 4 into the vector of shares I 21, I 22, I 23, I 24 lying on a random t degree polynomial with constant term being still I 2

and for that what the party do is the following. The first party will act as a dealer and secret share of value k 1 through a random t degree polynomial. Party P 2 will act as a dealer and secret share the share k 2 through a random t degree polynomial.

P 3 will act as a dealer and secret share k 3 through a random t degree polynomial and P 4 will act as a dealer and secret share k 4 through a random t degree polynomial and then the parties compute linear combination of the shared k 1, k 2, k 3, k 4 namely the parties compute the linear function this c 1 times k 1 c 2 times k 2, c 3 times k 3 + c 4 times k 4. So, k 1, k 2, k 3, k 4 they are available in secret shared fashion.

So, applying the same linear combination on the shares the parties obtain this vector of shares I 21, I 22, I 23, I 24 which will be now lying on a random t degree polynomial whose constant term is t and then the parties will locally add their shares of their output of this plus gate and the output of this product gate to get the shares of y and then they will publically reconstruct it.

So, that will be the entire set of values which are computed and communicated in the protocol. So, imagine that we have here n = 4 and up to t parties could be corrupt where t is less than n over 2 that means t could be at most one you cannot have two corruptions possible here because if two parties are corrupted in the system then the protocol will not provide you privacy.

We will formally prove that later that this is a necessary condition, but for now you have to believe me. So, the value of t which we can tolerate here or the number of faults which we can tolerate here is 1. So, that means our protocol should satisfy the privacy property even if P 1 is corrupt or P 2 is corrupt or P 3 is corrupt or P 4 is corrupt and even it is computationally unbounded.

And we have to formally prove that the corrupt party does not learn anything additional beyond the function output and the inputs of the corrupt parties whatever it could infer from those two things apart from that nothing additional should be revealed. So, I am taking the case where P 2 is corrupt and if P 2 is corrupt then the view of P 2 namely its input, its randomness whatever messages it has communicated.

And whatever messages it has received that whole collection of messages is denoted in this rectangular box. So, I have categorized what are the things that are present in view 2 as per the gates. So, if I consider the input gates P 2 view consists of the following if it is under the control of the adversary. So, when I say view of 2 I mean view of the adversary because right now I am considering the case when P 2 is under the control of the adversary.

The same analysis you can do if P 1 is corrupt or if P 3 is corrupt or P 4 is corrupt. So, for the input gates the view of P 2 is the following. It will know that it has picked this random t degree polynomial whose constant term is x 2 because P 2 itself has selected that polynomial that is a randomness component of P 2 with respect to the input gates and now what does P 2 learns regarding the inputs of the other parties.

So, for x 1 it will be knowing this share x 12 namely the evaluation of the polynomial fancy x 1 at alpha 2. Of course, it knows the full sharing polynomial through which x 2 is shared so I am not writing it out explicitly in the view of P 2 namely the share of P 2 for x 3 it will know one of the shares namely x 32 which P 3 would have communicated to P 2 and that is the evaluation of P 3 sharing polynomial which is a random t degree polynomial.

And that polynomial is evaluated at alpha 2 the value is given as a share to P 3 and similarly for P 4 inputs P 2 view will be this x 4. So, that is what I have written down here. Now, what will be the view of P 2 with respect to the addition gate? Well P 2 itself could have added x 12, x 22 and would have obtained this share with I 12. So that is what I have written down here. So, let us see what is the view of P 2 with respect to the GRR degree reduction.

For the GRR degree reduction P 2 will be under the control of k 2 and it itself would have shared k 2 through this polynomial fancy k 2 which is a random t degree polynomial so that is what is included in view 2 and for the other k shares namely k 1, k 3 and k 4 what will be the information available with corrupt P 2 only one share namely k 12, k 22, k 32 and k 42 of course k along for the second share k 2 it will have the full vector of shares of k 2.

Namely k 21, k 22, 23, k 24 because P 2 itself has acted as a dealer and secret share them, but for the other k values here k 1, k 3, k 4 adversary will have only one share namely the evaluation of the corresponding sharing polynomials at alpha 2 so that is what I have written

down and then once P 2 has k 12, k 32, k 42 and of course the share of k 2 evaluated at alpha 2.

It applies this Lagrange linear combination on this k 12, k 22, k 32 and k 42 and obtains the share I 22 which is a part of this vector of shares that will be the overall view of P 2 with respect to the multiplication gate and then P 2 will add I 12 and I 22 so it will obtain y 2 so that will be a part of view of view 2 and now during the reconstruction phase apart from y 2 every other party would have communicated their respective shares of y 2 P 2.

Namely y 1 would be made public, y 3 also will be made public and y 4 will be also made public. So, the full vector of y shares will be now included in view 2 that is what is the information which is seen by a corrupt P 2 a potentially corrupt P 2 in an execution of the BGW MPC protocol assuming that we are following the GRR degree reduction method and now we have to argue that this view is of no use for the corrupt P 2 to infer anything additional about the inputs x 1, x 3 and x 4.

Of course from y so this vector of y shares will anyhow leak the full y to the corrupt P 2 and of course it knows the input x 2. We want to show that whatever could be inferred from x 2 and y only that much is revealed in this protocol and nothing additional about x 1, x 3 and x 4 is leaked and how do we prove that remember our privacy definition is the following. We have to argue formally that whatever information a corrupt P 2 or the adversary would have seen from the honest parties by participating in the protocol.

If that can be recreated or simulated just based on the input of the bad guy and the function output then it is equivalent to saying that whatever information or communication which adversary has seen in this by participating in the MPC protocol is of no use because only when communication happens from the honest parties and adversaries sees those communication.

Then only there is a possibility that something about the inputs of the honest party is revealed because if no communication happens among the parties then my protocol will be definitely secure and if somehow we can compute the function output we have to argue that whatever communication has happened from the honest parties to the bad guys that is of no use to the adversary.

And the way we formally prove is that we show that the information which honest parties would have communicated to the bad guys during the execution of the protocol can be recreated or simulated just based on the inputs and output of the bad guys. So, the bad guy in this case is P 2 we have to show that there is some simulation algorithm a simulator who can reproduce the entire view just based on x 2 and y.

And without even knowing the exact values of x 1, x 3 and x 4 if the simulator is able to do that then it is equivalent to saying that from this view 2 nothing additional about x 1, x 3, x 4 can be revealed because the entire view if it can be reproduced, regenerated without the knowledge of x 1, x 3, x 4 then how it can be possible that view 2 (()) (26:37) something about x 1, x 3 and x 4.

So, that is the security proof we have to see the simulation here how exactly the simulator is going to reproduce the information and when I say reproduce the information she has to generate a sequence of values whose probability distribution is same as this view 2 because remember view 2 does not consist of fixed values because say for instance this polynomial fancy x 2 is a random polynomial.

It could be any polynomial from the set of all possible polynomials of degree t whose constant term would have been x 2. So, it is not the case that every time P 2 gets corrupt for the input gates it is only uses this fancy x 2 polynomial this fancy x 2 polynomial could be any random t degree polynomial. In the same way it is not the case that if P 2 gets corrupt for every execution of the protocol it will see the same values of x 12, x 32 and x 42 because they are random shares picked or computed by the other parties and communicated.

So, that is why we have also discussed this earlier the view is always a random variable. So, the goal of this simulator will be to produce a view whose probability distribution is identical to the real view 2 which the corrupt party would have obtained by interacting with the honest parties in the execution in the MPC protocol and before going into the simulation strategy let us try to intuitively understand that why exactly it is possible to simulate or recreate the view 2 without even the knowledge of the inputs of the honest parties.

So, let me give you the intuition so what exactly are the components of the view with respect to the inputs gates P 2 has its sharing polynomial whose constant term is x 2, but that polynomial is a random polynomial. Now since x 2 is anyhow known to the simulator, simulator can just write down a random t degree polynomial whose constant term is x 2 and that it is probability distribution will be identical to the probability distribution of this fancy x 2 polynomial.

That means that picking this polynomial by P 2 and distributing shares on this polynomial is not a privacy breach because P 2 itself is doing it. So, this action can be simulated, it can be recreated just based on the knowledge of x 2 itself. So, that is what the simulator is going to do here for the sharing polynomial picked by P 2 the simulator just writes down a random t degree polynomial whose constant term is x 2 I am calling that polynomial as tilde fancy x 2.

And the obvious claim here is that if I consider the probability distribution of the polynomial x 2 which is picked by the party P 2 in the MPC protocol its probability distribution is identical to this simulated x 2 polynomial picked by the simulator because both of them are a random t degree polynomial whose constant term is x 2 so their distribution are identical. Now how can the simulator recreate this information x 12, x 32 and x 42.

Well what is x 12? X 12 is a share on this random t degree polynomial picked by the party P 1 and party P 1 is not under the control the adversary. So, during the protocol execution during the real MPC protocol execution a corrupt P 2 would have just obtained a share x 12 and remember from the privacy property of Shamir secret sharing that value x 12 could be a share for any random t degree polynomial whose constant term could be any value.

That means even if x 1 would have been equal to 0 and if a polynomial would have been picked from x 1 from the set of all possible polynomials whose constant term is 0 the probability that x 12 is actually the evaluation of this fancy x 1 polynomial is identical that the probability that this x 12 is actually the evaluation of a random sharing polynomial of degree t whose constant term is another value from the field.

So, this we had proved when we discuss the properties of t degree polynomials we have proved that if you take any t random values from the field they could be equally likely the shares of a random t degree polynomial whose constant term is S and equally likely the same

t values could be the shares or the evaluation of a random t degree polynomial whose constant term is S prime.

There distribution of those fixed t shares or t values is independent of what exactly is the concrete sharing polynomial picked and that is precisely ensures that x 12, x 32, x 42 even though those are the shares which P 2 is seeing in the real MPC protocol it is of no use for P 2 because from P 2 view point it could be the share of any candidate x 1 any candidate x 3, any candidate x 4.

So, based on this intuition it is very easy, very simple to simulate those shares. So, what the simulator is doing is it is just picking a random t degree polynomial on the behalf of a honest P 1 that means it is playing the role of P 1 in its mind, it is playing the role of P 3 in its mind and it is going to play the role of P 4 in its mind and pick sharing polynomial on behalf of P 1, P 3 and P 4.

When I say behalf of P 1, P 3, P 4 that means it is now simulating the actions of P 1, P 3, P 4 as per the BGW shared circuit evaluation assuming that x 1 would have been 0, x 3 would have been 0 and x 4 would have been 0. So, how exactly P 1 would have behaved in the BGW shared circuit evaluation if it would have wanted to share the value x 1 equal to 0. Well it would have picked a random t degree polynomial whose constant term would have been 0 and to party P 2 it would have given this share.

Similarly, if P 3 would have wanted to secret share the input 0 then it would have picked a random t degree polynomial whose constant term will be 0 and P 2 would have received this share and similarly if P 4 wanted to secret share the values 0 as its input it would have picked a random t degree polynomial like this and P 2 would have obtained this shares and these are precisely the shares which are now simulator, recreating as part of x 12, x 32, x 42.

And my claim now is the following. If I consider the probability distribution of the share x 12 which P 2 has received in the MPC protocol its probability distribution is identical to the probability distribution of this simulated x 12 which simulator is writing down because this again comes from the fact that x 12 is a value on a random polynomial of degree 1 because t is equal to 1.

So, x 12 is the value of a random one degree polynomial at alpha 2 and x 12 simulated x 12 is also a value on a random polynomial of degree 1. Of course, the constant terms of the polynomial fancy x 2 and the simulated x 2 they are different, but we have already proved earlier that irrespective of what exactly is the constant term of the random sharing polynomial the probability distribution of t shares are identical.

Following the same logic it follows that the simulation of the real x 32 share means which P 2 is seeing in the MPC protocol is identical to the simulated in the x 32 shares from the view point of a corrupt P 2. P 2 cannot tell whether this x 32 could also be the share of this simulated x 3 polynomial and simulated x 32 share also could be the share for the real x 2 polynomial both can happen with equal probability.

And due to the same argument x 42 the probability distribution of the real x 42 share is identical to the probability distribution of the simulated x 42 shares that means from the view point of anyone till now whatever the simulator has simulated its probability distribution is identical to the corresponding components of real view 2. Now what exactly the simulated view for I 12.

Well the simulator now for the rest of the similar is going to play the role of P 1, P 2, P 3, P 4 why it can play the role of P 1, P 2, P 3, P 4 because for P 2 it knows what polynomials P 2 is going to pick. For P 1 it has picked the polynomial itself for P 3 it has picked the polynomial itself and for P 4 it has picked the polynomial itself. So, now what this simulator is going to do? Simulator is going to run the full instance of shared circuit evaluation.

Assuming that this simulated x 1, simulated x 2, simulated x 3 and simulated x 4 are the sharing polynomials of P 1, P 2, P 3, P 4. So, for addition gate the share of party P 2 as per the view 2 should have been this value that is what simulator writes down. Now the simulator has to simulate the information which adversary or the corrupt P 2 receives as part of view 2 during the GRR degree reduction instance.

Again let us pause here and try to understand that why the information that adversary or the corrupt P 2 receives during the GRR degree reduction is of no use for P 2. This we have already argued, we have argued earlier when we discussed the analysis of the GRR degree

reduction that whatever information adversary sees during the GRR degree reduction it can be easily simulated, but again let us go through that.

So, P 2 in the actual GRR degree reduction will have this full polynomial fancy k 2 because it itself is picking that polynomial, but for the other polynomials fancy k 1, fancy k 3 and fancy k 4 it is receiving only one share namely the evaluation of those polynomials at alpha 2 and what are those polynomials? Each of this polynomials fancy K 1 Z, fancy K 3 Z, fancy K 4 Z all of them are random t degree polynomials.

So, that means whatever information the corrupt P 2 is receiving during the GRR degree reduction instance that is of no use because those shares could be the shares of any random t degree polynomial whose constant term could be anything. Based on that intuition if this is how the simulator is going to reproduce the information which is there in view 2 as part of the GRR degree reduction.

This simulator it now knows the full sharing polynomials of P 1, P 2, P 3, P 4 for P 2 it itself has picked the sharing polynomial, for P 1, P 3, P 4 it has picked the corresponding simulated sharing polynomials. So, it performs the role of P 1, P 3, P 4, P 2 as per the GRR degree reduction.

Namely whatever P 1 would have done, whatever P 2 would have done, whatever P 4 would have done simulator does all those computation in its mind namely on the behalf of P 1 it picks a simulated K 1 polynomial which is a random t degree polynomial whose constant term is the simulated k 1. On behalf of P 2 it picks a random t degree polynomial random t degree simulated K 2 polynomial whose constant term is the simulated k 2 and so on.

And now the view that it is reproducing for the multiplication gate is the following it will say that P 2 will pick this random sharing polynomial and indeed the probability distribution of this polynomial is identical to the probability distribution of this. Do not get confuse that the values of k 2 and k 2 tilde are different. Yes they are different because it could be the case that the honest parties have actually participated in the protocol with x 1 = 0, x 3 = 0, x 4 = 0 that would have produced this simulated k 2 tilde.

Whereas if the parties would have started the protocol with x 1, x 3, x 4 some other values then that would have produced this value of k 2. So, it is not the case that k 2 is a fixed value. Based on what are x 1, x 3, x 4 that determines the value of k 2 and what are the shares of basically x 1, x 2, x 3, x 4 which were used in the protocol. So, k 2 is not a fixed value we have to just compare the probability distribution.

So, my claim is that the probability distribution of this real k 2 polynomial which is there in view 2 which is identical to the probability distribution of this simulated k 2. Both of them are random t degree polynomial the constant term of this real k 2 polynomial is the share k 2 where k 2 would have been the product of x 32 and x 42 and what is the property of this simulated K 2 polynomial?

Well it is also a random t degree polynomial its constant term is the simulated k 2 and the simulated k 2 is actually the product of the simulated x 32 share and simulated x 42 share. So, component wise probability distribution are same and for the other parties namely for k 1, k 3 and k 4 for the other inputs k 1, k 3, k 4 for which the P 2 would have received some shares, simulator is just going to write down the corresponding shares as per the simulated K 1 polynomial, simulated K 3 polynomial and simulated K 4 polynomial.

And again it is easy to see that the probability distribution of the simulated k shares which simulator is reproducing here is identical to the real k shares which P 2 would have received in the MPC protocol. So, till now what simulator is basically done is it knows the input of the bad guy and the function output fixing that it is just assuming or making the assumption that inputs of the other honest parties are 0.

And basically executing their steps as per the shared circuit evaluation and reproducing all the information that they would have communicated and computed and till now we have seen that component wise the probability distribution of view 2 is identical whatever the simulator has reproduced. Now comes the final gate that is the output gate. So, this procedure will be followed if there would have been any other gates also in the circuit.

But now since there are no more gates in the circuit we have reached a final gate the simulator has to reproduce this vector of shares y 1, y 2, y 3, y 4. Now simulator at this point has simulated the entire computation assuming that x 1 was 0, x 2 is actually equal to x 2, x 3

= 0 and x 4 = 0, but the real execution may not have happened with x 10, x 30, x 40 that means the y which was the real function output could be a function output where the inputs might be a non zero inputs.

But till now simulator has run the entire shared circuit evaluation assuming that the inputs of the other honest parties are 0 that means if I consider the simulated y 1 share simulated, simulated y 3 share, simulated y 4 share together they will lie on a t degree polynomial whose constant term may not be the y which view 2 has. So, y is also a part of view 2 because that is a function output.

So, the simulator has to produce a vector of y shares which should interpolate to the actual function output y and those simulated vector of shares should be a t degree polynomial then only we can say that overall the simulated view is identical to the real view 2. Simulator cannot just write down some arbitrary y shares here that simulator cannot do here because even if they lie on a t degree polynomial its constant term may not be y.

And that will be a contradiction to the claim that the simulated view is identical to the real view because in the real view the property of this y vector of shares is that they interpolate to the function output y. So, how exactly simulator is going to produce the simulated vector of output share well it will be identical to the way we have simulated the view of the adversary when we analyze the BGW protocol for the linear function.

So, remember apart from this simulated y 2 share the simulator also have access to now the function output because that is also given to the simulator and simulator knows that okay 0, y is also going to lie on the final t degree polynomial which everyone would have reconstructed when the shares of y would have been made public. So, now altogether how many shares of y are available with or how many points on the y sharing polynomial are available with the simulator, simulator have now t number of shares or t number of points.

It already have t number of simulated shares corresponding to the t corrupt parties and it also knows that 0, y is also going to be a point on that polynomial together that fixes the y sharing polynomial and once the y sharing polynomial is fixed that automatically fixes what are the shares of y which the honest parties would have set. Basically this models the intuition that in

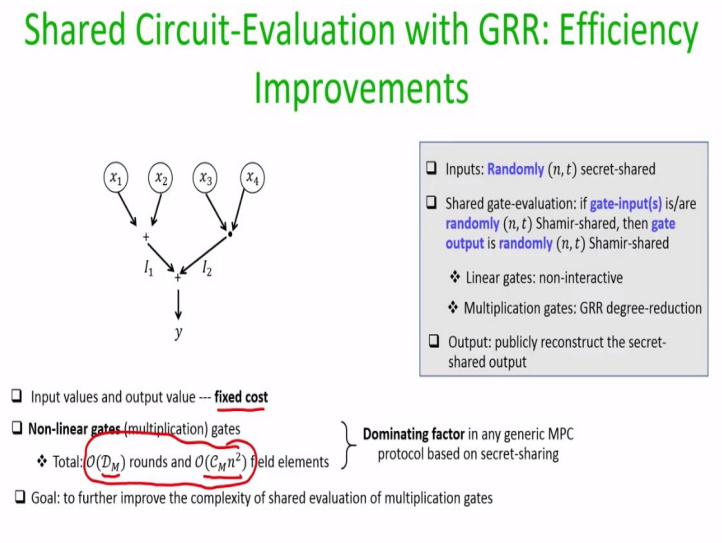
the real MPC protocol leaking this or making public this full vector of y shares is not of any harm to the adversary.

It is not going to cause any harm as for the privacy is concerned because anyhow for this y value adversary had t shares till now if y shares are not made public and anyhow if it knows that the function output is going to be y it itself can interpolate the full sharing polynomial and write down what are the shares which honest parties would have communicated in case we ask everyone to make the y shares public and that is what a simulator is basically doing here.

It takes the point 0, y and along with that interpolate the simulated y shares and that is the vector of y shares which it writes down as part of the final y vector and now you can see that the distribution of the real y shares as per the view 2 and distribution of this simulated y shares they have identical probability distribution. Both of them have t degree polynomials lie on a t degree polynomial with constant term being y.

And the second component is the component which P 2 would have in both the views and that shows that whatever has been regenerated by this simulator is identical to the view 2 and that proves that this shared circuit evaluation approach using the degree reduction is secure.

(Refer Slide Time: 48:16)



So, to summarize we have fixed cost with the input and the output gates, but for the non-linear gates we require these many number of rounds and this much communication and this is the dominating factor in any generic MPC protocol. So, that is why whenever we study any

generic MPC protocol the focus will be to reduce the communication and the number of rounds required to evaluate the multiplication gates in the circuit because that is a dominating factor and that will be the focus for our next lecture. Thank you.