


**Secure Computation: Part II**  
**Prof. Ashish Choudhury**  
**Department of Computer Science and Engineering**  
**Indian Institute of Science, Bengaluru**

**Lecture - 57**  
**Ingredients for Statistically-Secure MPC**

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:24)

## Lecture Overview



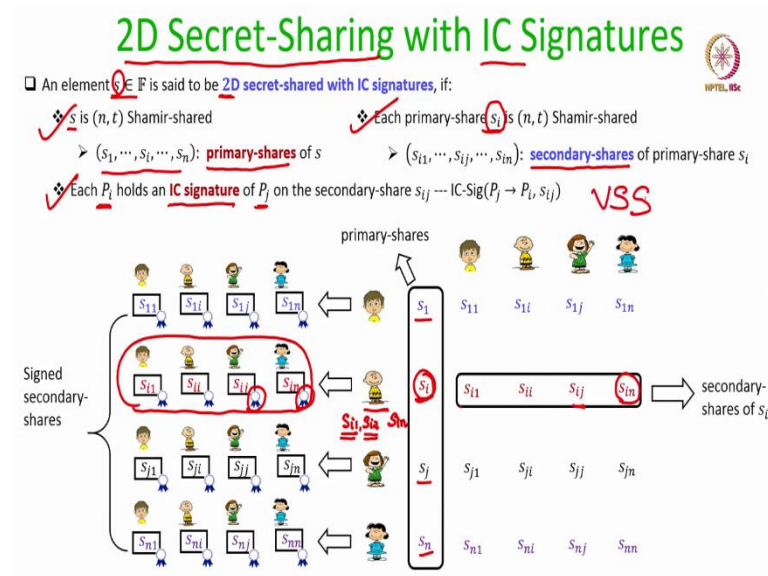
- ❑ Ingredients for statistically-secure MPC
  - ❖ 2D Secret-sharing with IC-signatures
    - Statistically-secure reconstruction protocol with  $t < \frac{n}{2}$
  - ❖ Statistically-secure protocol for verifying multiplicative relationship

So, we will now start our discussion on statistically secure MPC. We already discussed the IC protocol. So, for statistically secure MPC we will require a variant of verifiable secret sharing. The data which is generated by that verifiable secret sharing is called as 2D secret sharing with IC signatures.

So, I will introduce this 2D secret sharing with IC signatures in this lecture, and then we will see a statistically secure reconstruction protocol if we have some data which is 2D secret shared with IC signatures.

We will also see a statistically secure protocol for verifying the multiplicative relationship this will be slightly different from a perfectly secure protocol for verifying the multiplicative relationship which we had discussed earlier. Looking ahead this will be useful for generating the Beaver's random multiplication triples for our statistically secure MPC.

(Refer Slide Time: 01:27)



So, let us start with 2D secret sharing with IC signatures. So, a value  $s$  from the field will be 2D secret shared with IC signatures, if all the following hold. The data  $s$  should be secret shared as per Shamir secret sharing with degree of sharing being  $t$  with every party  $P_i$  holding a share of  $s$ , namely there should exist a  $t$  degree polynomial whose constant term should be the secret  $s$ , and party  $P_i$  should hold the value on that  $t$  degree polynomial at  $\alpha_i$ .

So, the vector of shares  $s_1, s_2, s_i, s_n$  we will call them as the primary shares of the secret  $s$ , and this is the way a secret would have been considered as secret shared in our perfectly secure MPC protocol, but now we are actually augmenting this data structure data structure in the sense whatever data is available corresponding to the secret  $s$  and we are augmenting it in 2 dimension that is why this 2D.

Now, what is the second dimension of sharing here? The second dimension of sharing is that each primary share  $s_i$  will be further secret shared among the parties; that means, if I consider the share  $s_i$  corresponding to that there should exist a  $t$  degree polynomial whose constant term should be  $s_i$ , and every party should have the value of that straight line not sorry not straight line the value of that  $t$  degree polynomial at  $\alpha_i$ .

So,  $P_1$  will have the value of that  $t$  degree polynomial at  $\alpha_1$ ,  $P_j$  will have the value of that  $t$  degree polynomial at  $\alpha_j$ ,  $P_n$  will have the value of that  $t$  degree polynomial at  $\alpha_n$ . Now,

the vector of shares corresponding to the primary share  $s_i$  will be considered as the secondary shares.

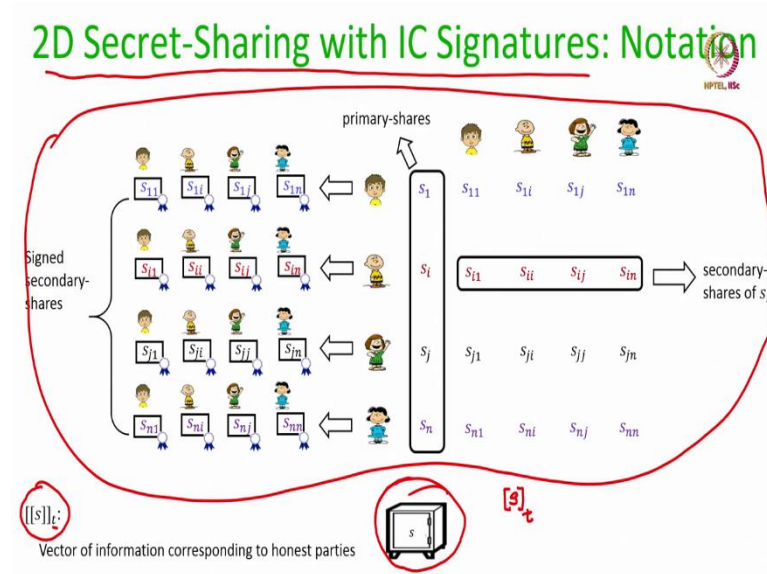
So, now you can see that you have 2 level of sharing here. One level of sharing is in terms of primary shares and now we are augmenting we are augmenting this data structure in the 2 dimensions by saying that each share first share, second share,  $i$ th share,  $j$ th share,  $n$ th share should be further secret shared resulting in secondary shares. So, that is a 2D secret sharing.

But we require some more augmentation here in the form of IC signatures. We require that each party  $P_i$  not only should have its primary share  $s_i$  and the secondary shares  $s_{i1}, s_{i2}, \dots, s_{in}$ , but it should also hold the signature of the  $j$ th party on the  $j$ th secondary share. So, those are pictorially represented by this notation.

So, this  $i$ th party it will have the full vector  $s_{i1}, s_{i2}, \dots, s_{in}$  and along with that on  $s_{i1}$  it will have party ones signature, on  $s_{i2}$  it will have the second party signature, on  $s_{ij}$  it will have the  $j$ th party signature, on the  $s_{in}$  it will have the  $n$ th party signature signed as per the information checking protocol which we had discussed earlier.

So, if such if all these three conditions hold then we say that the value  $s$  is 2D secret shared with IC signatures. How exactly such a data secret sharing, or such data structure will be generated that will be done by a statistically secure VSS protocol which we will discuss later.

(Refer Slide Time: 05:43)

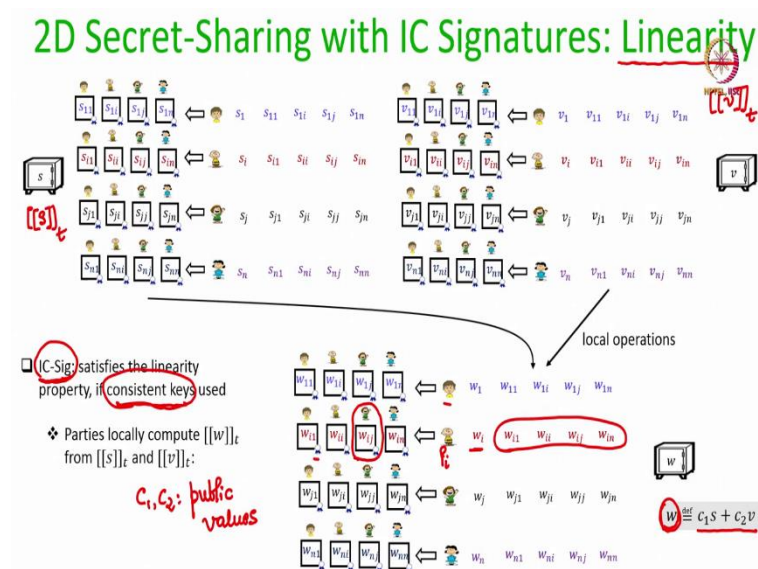


So, now, for notational convenience we will be using a box representation as we have used it for our perfectly secure protocol. So, in the perfectly secure protocol also we had used this box representation whenever a value  $s$  is secret shared through a  $t$  degree polynomial, we use this box for discussing perfectly secure protocols.

So, now I am abusing the notations and I will be using this box notation whenever a value  $s$  is 2D secret shared with IC signatures; that means, if you if you have a system of values distributed like this namely you have primary share, secondary shares and signed secondary shares, IC signed secondary shares then I will be using this box representation and sometimes I will also write this notation.

So, remember in the perfectly secure protocols we use the representation  $[s]_t$  to denote Shamir secret sharing of a value  $s$ . Now, since we have augmented the secret sharing in 2 dimensions. So, that is why I am using  $[[s]]_t$  denotes the degree of sharing for primary shares, secondary shares. So, that is a notation which we are going to now follow for the rest of our description on statistically secure VSS and MPC.

(Refer Slide Time: 07:19)



Like Shamir secret sharing which has the nice linearity property namely you can perform linear computation of any Shamir secret shared values non interactively we can perform non we can compute any linear function of 2D secret shared values non interactively; that means, suppose you have a value  $s$  which is 2D secret shared with IC signatures and say you have another value  $v$  which is also 2D secret shared with IC signatures.

And, suppose as part of the IC signatures we use consistent keys between every pair of parties every pair of prover and verifier; that means, whenever so, recall that the IC protocol which we had discussed in the last lecture it is based on information theoretic MAC and what I am saying here is that as part of that information theoretic MAC if we use consistent MAC keys between every pair of parties in the system then that will lead to an IC protocol where the IC protocol will have the linearity property; that means, the IC sig will have the linearity property.

So, if IC sig has the linearity property, then what we can say here is that we can perform linear we can we can compute linear functions of secret shared data non interactively. So, if  $w$  is the result of  $c_1 s + c_2 v$  where  $c_1$  and  $c_2$  are some public values from the field, then everyone can compute a 2D secret sharing for  $w$  with IC signatures.

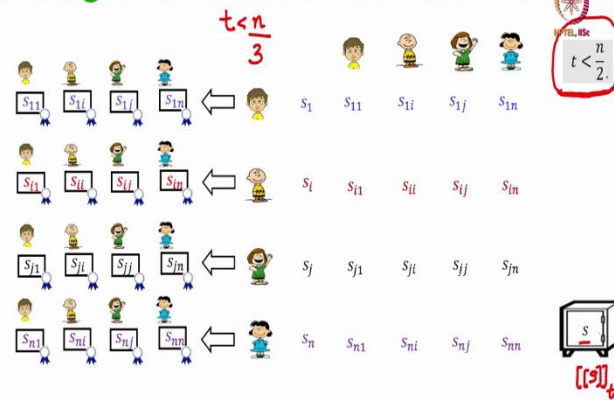
So, what will be for instance the primary shares of  $w$  for the parties, it will be the sum of the primary shares of  $s$  and  $v$  for the individual parties. Now, if I consider the  $i$ th party what will be the secondary shares for its primary share  $w_i$  it will be the sum of the

secondary shares of  $s_i$  and  $v_i$ , and now how the signature of the  $j$ th party will be computed by  $P_i$  on the  $j$ th secondary share  $w_{ij}$  it can be computed by using the linearity property of IC signature provided consistent keys would have been used as part of the IC protocol which will be guaranteed in our verifiable secret sharing protocol

So, like you had the linearity property for secret shared Shamir secret shared values you have the linearity property even for 2D secret shared value values with IC signature; that means, you can non interactively perform any computation, any linear computation on 2D secret shared data with IC signatures.

(Refer Slide Time: 10:53)

### VSS with IC Signatures: Reconstruction Protocol



Now, imagine you have a value  $s$  which is 2D secret shared with IC signatures and now we require a reconstruction protocol to publicly reconstruct  $s$  mind it. We are now in the setting  $t < \frac{n}{2}$ . So, you cannot use the perfectly secure protocols in the perfect security, in the world of perfect security to reconstruct a secret shared value  $s$  every party could have made public its share of  $s$  up to  $t$  corrupt parties could have produced incorrect shares to error correct we can use the Reed Solomon error correction property because in the perfect world we are in the setting where  $t < \frac{n}{3}$ .

But for the statistically secure protocols we will be in the setting where  $t < \frac{n}{2}$  and hence the Read Solomon error correction will not work. So, that is why the reconstruction of 2D secret shared value will be slightly different here. Our idea will be still the same we will

(Refer Slide Time: 12:29)

Diagram illustrating the reconstruction of a primary share  $s_i$  from signed secondary shares. The diagram shows a grid of shares  $s_{ij}$  and a set of shares  $s_i'$  where one share is crossed out. A red circle highlights the shares  $s_i'$  and a red oval highlights the shares  $s_i$ . A red arrow points from the shares  $s_i'$  to the shares  $s_i$ . A red box contains the text  $s_i$ .

So, our idea will be slightly changed we will not ask  $P_i$  directly to make public its primary share  $s_i$  because it can change its primary share from  $s_i$  to  $s'_i$  rather we will ask  $P_i$  to reveal all the signed second shares. So, it has to produce the secondary share  $s_{i1}$  signed by  $P_1$ , it has to produce the secondary share  $s_{i2}$  signed by  $P_2$ , it has to produce the secondary share  $s_{ij}$  signed by the  $j$ th party  $P_j$ , and it has to produce the secondary share  $s_{in}$  signed by the  $n$ th party.

Now, if any of this signed values are incorrect, incorrect in the sense that the signatures are incorrect then clearly the  $i$ th party is corrupt and it can be discarded from the system; that means, its primary share will not at all be considered, but suppose  $P_i$  is able to reveal all the signatures correctly, then we will use the revealed secondary shares  $s_{i1}, s_{i2}, \dots, s_{in}$  and see whether all of them lie on a  $t$  degree polynomial. If it is the case then using the



So, now you see that we are not asking  $P_i$  to make its primary share  $s_i$  directly public, rather we are asking him to make public the signed secondary shares which are verified for correct signature and whether they are lying on  $t$  degree polynomial, if both theses verifications pass then we interpolate them to get the primary share for  $P_i$ .

[illegible]

He should change at least one of the secondary shares say the secondary share corresponding to the first honest party or the second honest party or the third honest party or say the  $n$ th honest party, because only then the interpolated secondary shares will lead to an incorrect secondary share, because if the secondary shares corresponding to all the honest parties are revealed as it is by  $P_i$  then using them using by interpolating them the whatever primary share we recover on the behalf of  $P_i$  it is bound to be  $s_i$  only it will not be different.

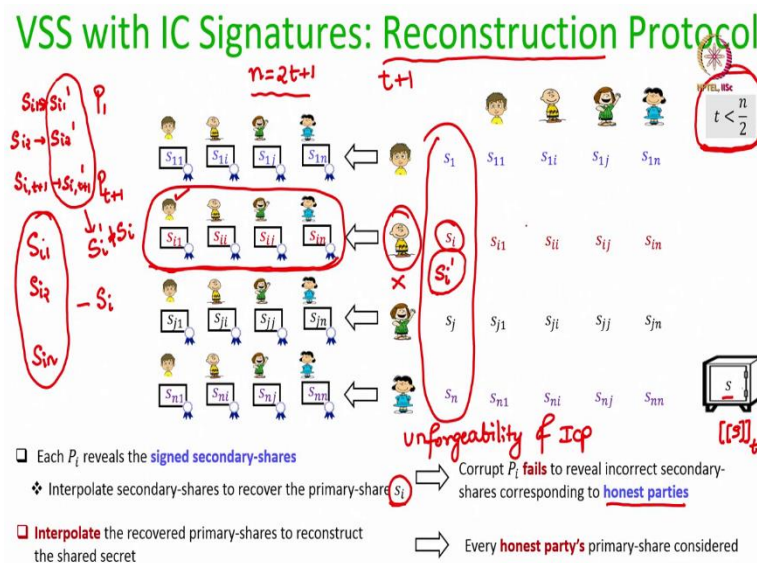


So, assuming that say for instance we are in the setting where  $n = 2t + 1$  and say the first  $t + 1$  honest parties are say the first  $t + 1$  parties are honest then either  $P_i$  should attempt to change  $s_{i1}$  to  $s'_{i1}$  or it should try to change  $s_{i2}$  to  $s'_{i2}$  or it should try to change  $s_{i(t+1)}$  to  $s'_{i(t+1)}$  and hope that all this secondary shares get accepted namely the corresponding signatures get accepted and when they are interpolated the resultant primary share will be  $s'_i$  which is different from  $s_i$ .

But now what is the probability that a corrupt  $P_i$  will be able to forge signature of at least one of the honest parties out of this  $t + 1$  honest parties on the corresponding secondary share it is very small, it is negligible which comes from the unforgeability property of your IC protocol; that means, if at all the signed values have been verified correctly.

And if all the secondary shares interpolate to a  $t$  degree polynomial, then with very high probability whatever is the primary share obtained on the behalf of the  $i$ th party it is the correct primary share, it is not an incorrect primary share.

(Refer Slide Time: 18:15)

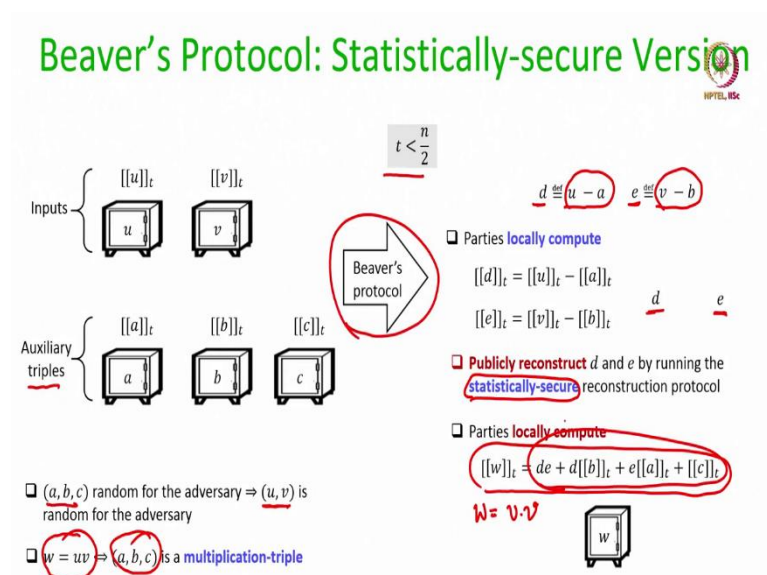


Now, once we have all the primary shares available for some parties the primary shares may not be available because that party would have been discarded if it tries to reveal an incorrect signed value, but since we are in the setting where  $n = 2t + 1$  there will be at least  $t + 1$  honest parties whose primary shares will always be finally, available we interpolate them to reconstruct a shared secret.

So, now you can see that even though we are in the setting where  $t < \frac{n}{2}$  without even applying the Reed Solomon error correction procedure we are getting we are we get a mechanism to reconstruct a secret shared value with very high probability, and you can see that what is the purpose of the secondary shares and the signatures on the secondary shares.

If we do not put signatures on the secondary shares then again  $P_i$  can simply make public any bunch of secondary shares public and it will get accepted, but as soon as we put signatures of the respective parties on the secondary shares it becomes difficult for a corrupt  $P_i$  to get an incorrect primary share being considered on its behalf.

(Refer Slide Time: 19:42)



Now, we will see a variant of Beaver's protocol which is now statistically secure this will be later useful for evaluating the multiplication gates in the circuit and this variant will work even if we are in the setting where  $t < \frac{n}{2}$  we do not require the setting of  $t < \frac{n}{3}$ .

So, the inputs here will be a pair of values  $u$  and  $v$  both of which are secret shared 2D secret shared with IC signatures and the auxiliary data will be a secret shared triplet  $a, b, c$  where the  $a, b$  and  $c$  components of the triplet are 2D secret shared with IC signature. So, the way Beaver's protocol will work in this particular setting will be the following. Let us define the value  $d$  and  $e$  to be the one-time pad encryptions of  $u$  and  $v$  with respect to the  $a$  and  $b$  components of the triplet.

Because of the linearity property of 2D secret sharing the values  $d$  and  $e$  can be computed the secret sharing of the values  $d$  and  $e$  can be computed non interactively and then the parties can run the statistically secure reconstruction protocol which we had discussed in the earlier slide to publicly reconstruct  $d$  and  $e$ .

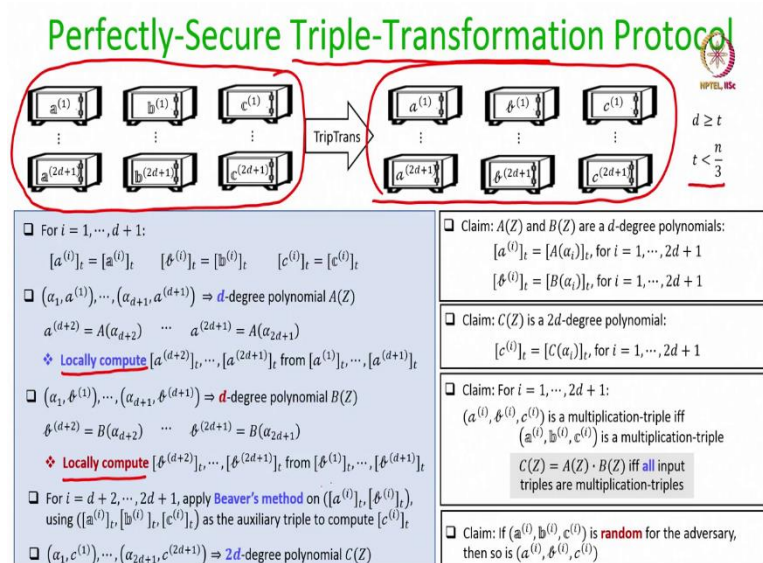
So, with very high probability the parties will be reconstructing the correct  $d$  and  $e$ , and now once  $d$  and  $e$  are publicly available the output of the Beaver's protocol can be again computed non interactively namely the parties linearly compute this function on secret shared  $a, b, c$ .

As usual we can claim the privacy property for the Beaver's protocol for this variant as well namely if the triplet  $a, b, c$  is random for the adversary, random in the sense that apart from the shares which adversary has for this triplet  $(a, b, c)$  the view of the adversary is independent of what exactly the values of  $a, b$  and  $c$ .

If that is the case then in this whole protocol no additional information about the inputs  $u$  and  $v$  is revealed that this is because even though the OTP encryptions of  $u$  and  $v$  are made public, the corresponding parts  $a$  and  $b$  are random for the adversary and the second property is that the output secret shared value  $w$  will be the product of the inputs to this protocol if and only if  $a, b, c$  is a multiplication triplet.

So, if  $a, b, c$  is a multiplication triplet then indeed the secret shared  $w$  will be equal to the product of secret shared  $u$  and  $v$  and vice versa and this simply comes by rearranging the terms here and expanding the values  $d$  and  $e$ .

(Refer Slide Time: 22:51)

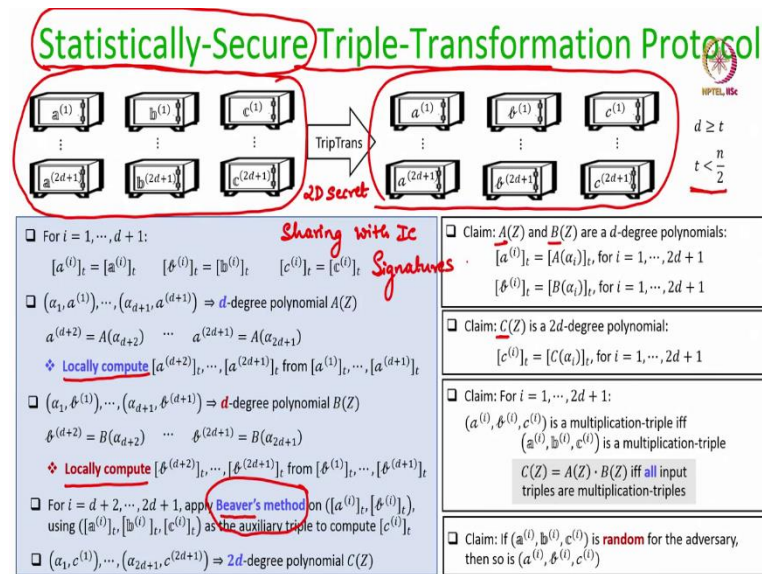


We can now so ok so, once we have a statistically secure variant of the Beaver's protocol we can go to the perfectly secure triple transformation protocol which we had discussed for the setting where  $t < \frac{n}{3}$ , and we can see that the same protocol we can run for the setting of  $t < \frac{n}{2}$  if a negligible error is allowed.

So, in the triple transformation protocol we have input a bunch of Shamir secret shared triplets and we transform them into another bunch of Shamir secret shared triplets such that even if there is no relationship among the input triplets, we end up setting up some correlation among the output triplets.

But now, our inputs will be 2D secret shared values with IC signatures. So, if you see this triple transformation protocol almost all the steps involve only performing linear functions or involving or involve performing local computation on secret shared data, the only interaction is in this step where the Beaver's method is triggered.

(Refer Slide Time: 24:22)



So, what we are going to do is we are now going to change the setting to  $t < \frac{n}{2}$ , and instead of a perfectly secure protocol we will target a statistically secure protocol and the idea will be that we can use the statistically secure variant of the Beaver's method which we had discussed earlier, rest of the proto steps of the triple transformation protocol remains the same.

Namely we define the  $A$  polynomial and  $B$  polynomial non interactively and the  $C$  polynomial is set in 2 stages we first fix few points on the  $C$  polynomial and the remaining points are obtained by applying the Beaver's method. Now, we will use the statistically secure variant of the Beaver's method and all our secret sharing will be now 2D secret sharing with IC signatures.

(Refer Slide Time: 25:29)

### Perfectly-Secure Protocol PolyVer: Required Properties

Arbitrary  $(n, t)$  Shamir-shared triples

□  $t < n/3$

❖ For simplicity, let  $n = 3t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = b^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = b^{(n)}$	$C(\alpha_n) = c^{(n)}$

□ Goal:

❖ Check if  $C(Z) = A(Z) \cdot B(Z)$ ?

❖ If  $C(Z) = A(Z) \cdot B(Z)$ , then at most  $t$  points on  $A(Z), B(Z), C(Z)$  allowed to be revealed to the adversary

So, now, once we have a triple transformation protocol we can next think about polynomial verification protocol with statistically secure properties, namely where a small error is allowed here.

So, recall we had seen a perfectly secure polynomial verification protocol for the setting  $t < \frac{n}{3}$ , where we have a triplet of polynomials  $A, B, C$  polynomials and the points on the  $A, B, C$  polynomials are secret shared, and we wanted to check the multiplicative relationship between the  $A, B, C$  polynomials.

(Refer Slide Time: 26:15)

### Perfectly-Secure Protocol PolyVer: Required Properties

Arbitrary 2D-secret-shared triples with IC signatures

□  $t < n/3$

❖ For simplicity, let  $n = 3t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = b^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = b^{(n)}$	$C(\alpha_n) = c^{(n)}$

□ Goal:

❖ Check if  $C(Z) = A(Z) \cdot B(Z)$ ?

❖ If  $C(Z) = A(Z) \cdot B(Z)$ , then at most  $t$  points on  $A(Z), B(Z), C(Z)$  allowed to be revealed to the adversary

Now, our inputs will be 2D secret shared with IC signatures.

(Refer Slide Time: 26:21)

### Perfectly-Secure Protocol PolyVer: Required Properties

Arbitrary 2D-secret-shared triples with IC signatures

□  $t < n/2$

❖ For simplicity, let  $n = 2t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = \phi^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = \phi^{(n)}$	$C(\alpha_n) = c^{(n)}$

□ Goal:

❖ Check if  $C(Z) = A(Z) \cdot B(Z)$ ?

❖ If  $C(Z) = A(Z) \cdot B(Z)$ , then at most  $t$  points on  $A(Z), B(Z), C(Z)$  allowed to be revealed to the adversary

We will be in the setting where  $t < \frac{n}{2}$ .

(Refer Slide Time: 26:25)

### Statistically-Secure Protocol PolyVer: Required Properties

Arbitrary 2D-secret-shared triples with IC signatures

□  $t < n/2$

❖ For simplicity, let  $n = 2t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = \phi^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = \phi^{(n)}$	$C(\alpha_n) = c^{(n)}$

□ Goal:

❖ Check if  $C(Z) = A(Z) \cdot B(Z)$ ?

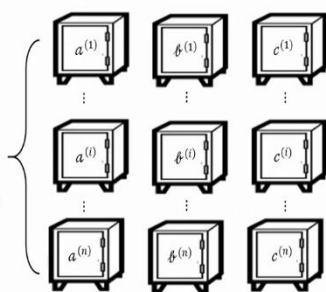
❖ If  $C(Z) = A(Z) \cdot B(Z)$ , then at most  $t$  points on  $A(Z), B(Z), C(Z)$  allowed to be revealed to the adversary

And we want to still check whether the  $C$  polynomial is equal to the product of  $A$  and  $B$  polynomials.



(Refer Slide Time: 26:33)

## Statistically-Secure Protocol PolyVer: Required Properties



Arbitrary 2D-secret-shared triples with IC signatures

$t < n/2$   
 ❖ For simplicity, let  $n = 2t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = b^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = b^{(n)}$	$C(\alpha_n) = c^{(n)}$

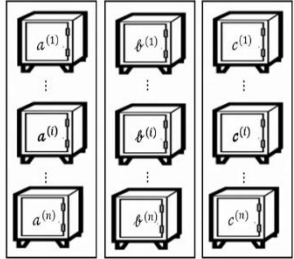
❖ Goal:

- ❖ Check if  $C(Z) = A(Z) \cdot B(Z)$ ?
- ❖ If  $C(Z) = A(Z) \cdot B(Z)$ , then at most 1 point on  $A(Z), B(Z), C(Z)$  allowed to be revealed to the adversary

But, in the process we will be letting the adversary to learn only one point on the  $A, B, C$  polynomials. For the perfect protocol we were allowed to reveal up to  $t$  points on the  $A, B, C$  polynomials, but now in the statistically secure protocol we will see that just by revealing one point on  $A, B$  and  $C$  polynomials publicly we will be able to check whether the multiplicative relationship holds with high probability or not.

(Refer Slide Time: 27:05)

## Statistically-Secure Protocol PolyVer



$n = 2t + 1$

❖ Polynomials  $A(Z), B(Z), C(Z)$  of degree  $t, t$  and  $2t$  respectively

$A(\alpha_1) = a^{(1)}$	$B(\alpha_1) = b^{(1)}$	$C(\alpha_1) = c^{(1)}$
$\vdots$	$\vdots$	$\vdots$
$A(\alpha_n) = a^{(n)}$	$B(\alpha_n) = b^{(n)}$	$C(\alpha_n) = c^{(n)}$

❖ Idea:

- ❖ Publicly check if  $C(\beta) = A(\beta) \cdot B(\beta)$
- ❖  $\beta$  a random element from  $\mathbb{F}$ , independent of  $A(Z), B(Z)$  and  $C(Z)$

*Handwritten notes:*  
 $\beta$  is a root of  $D(Z)$   
 $D(Z) = C(Z) - A(Z) \cdot B(Z)$   
 $D(\beta) = C(\beta) - A(\beta) \cdot B(\beta) = 0$   
 $D(Z) \text{ def } C(Z) - A(Z) \cdot B(Z)$

❖ Claim:

- ❖ If  $C(Z) \neq A(Z) \cdot B(Z)$ , then  $C(\beta) \neq A(\beta) \cdot B(\beta)$  except with probability at most  $\frac{2t}{|\mathbb{F}|}$

And the idea is very simple. The idea is that if indeed the  $A, B$  and  $C$  polynomials satisfy the multiplicative relationship. Then the  $C$  polynomial evaluated at any point  $\beta$ , any

random point  $\beta$  should be equal to the value of the  $A$  polynomial at  $\beta$  multiplied by the value of the  $B$  polynomial at  $\beta$ , and here  $\beta$  will be a random point which is going to be selected independent of what exactly are my  $A, B, C$  polynomials.

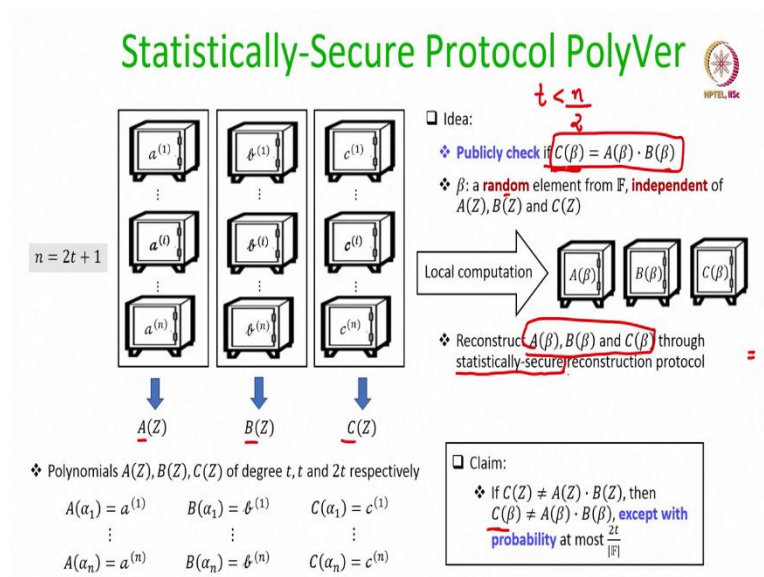
How exactly this random element  $\beta$  is going to be generated, we will discuss that later, but assume for the moment there is a mechanism through which the parties can generate publicly uniformly random value  $\beta$  which is independent of the points on the  $A, B, C$  polynomials, then just by checking this relationship they can check whether the multiplicative relationship among the  $A, B, C$  polynomials hold.

The claim here is that if the multiplicative relationship among  $A, B, C$  polynomials does not hold and even the multiplicative relationship between  $C(\beta), A(\beta)$  and  $B(\beta)$  will not hold except with probability  $\frac{2t}{|\mathbb{F}|}$ , why so, because the only way through which  $C(\beta)$  turns out to be  $A(\beta) \cdot B(\beta)$  even though the  $A, B, C$  polynomials do not satisfy the multiplicative relationship is that  $\beta$  turns out to be a root of the 2 degree polynomial say the  $D$  polynomial which is defined to be  $C(Z) - A(Z) \cdot B(Z)$ .

So, this  $D$  polynomial has degree  $2t$  because  $C$  polynomial has degree  $2t$ , and this point  $C(\beta) - A(\beta) \cdot B(\beta)$  is nothing but the point on the  $D$  polynomial and this value will be equal to 0 if and only if  $\beta$  is a root of the polynomial  $D$ , but remember the  $\beta$  value is selected randomly from the field and since the  $D$  polynomial has degree  $2t$  it can have at most  $2t$  roots.

So, the only way even though the  $C$  polynomial,  $A$  polynomial,  $B$  polynomial does not multiply the multiplicative relationship, but still  $C(\beta)$  turns out to be  $A(\beta) \cdot B(\beta)$ , is that the random point  $\beta$  turns out to be one of the roots of the polynomial  $D$  which can happen only with probability  $\frac{2t}{|\mathbb{F}|}$  because there are 2 at there could be at most  $2t$  number of roots for the polynomial  $D$ .

(Refer Slide Time: 30:40)




So, now how exactly the parties are going to check whether  $C(\beta) = A(\beta) \cdot B(\beta)$ ? Well, using the points  $A(\beta), B(\beta), C(\beta)$ . They can be computed as a linear function of the existing points on the  $A, B, C$  polynomials respectively. So, we can trigger the linearity property of 2D secret sharing with IC signatures and say that the parties can locally compute a 2D secret sharing with IC signatures for of the points  $A(\beta), B(\beta), C(\beta)$  respectively.

And then they can publicly reconstruct these points using our statistically secure reconstruction protocol with  $t < \frac{n}{2}$  which will guarantee that with very high probability the correct points  $A(\beta), B(\beta), C(\beta)$  are reconstructed, and now once you have the points  $A(\beta), B(\beta), C(\beta)$  in public we can check whether this condition holds or not.

(Refer Slide Time: 31:41)

## References



- ❑ Ronald Cramer, Ivan Damgård and Jesper Buus Nielsen: Secure Multiparty Computation and Secret Sharing. Cambridge University Press 2015, ISBN 9781107043053
- ❑ Ashish Choudhury, Arpita Patra: An Efficient Framework for Unconditionally Secure Multiparty Computation. IEEE Trans. Inf. Theory 63(1): 428-468 (2017)

So, with that I end this lecture. We had seen lots of ingredients which are going to be later useful for our statistically secure MPC namely we have introduced the data structure on which we will be performing use that we have seen the data structure which will be useful which will be used for performing computations.

In our shared circuit evaluation, we have seen how to reconstruct 2D secret shared value we have seen the linearity property, we had seen the polynomial verification protocol, and we have also seen the statistically secure variant of the triple transformation protocol. So, these are the references for this lecture.

Thank you.