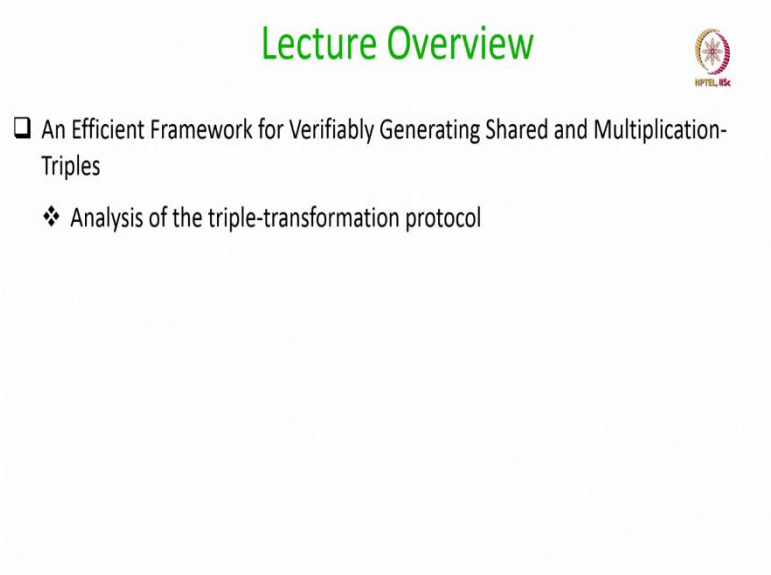
Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 50
Generating Random Multiplication-Triples: III

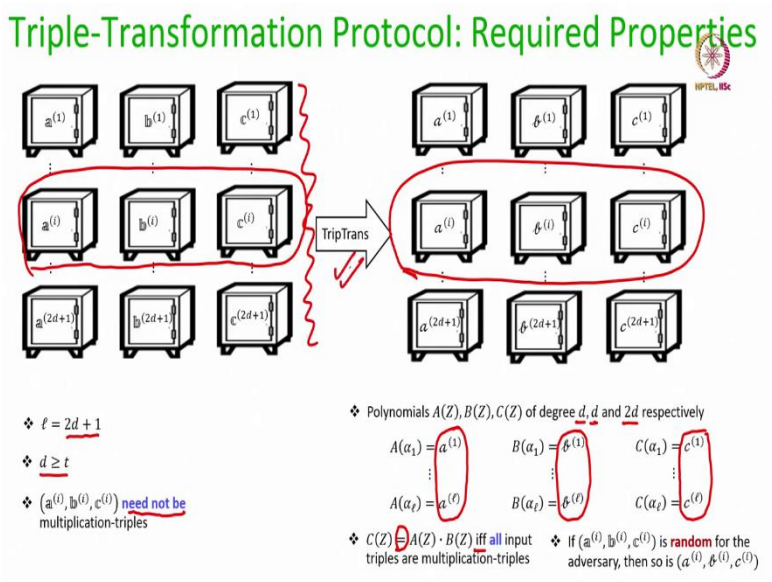
Hello everyone. Welcome to this lecture.

(Refer Slide Time: 00:25)



So, in this lecture we will continue our discussion regarding the efficient framework for generating shared random multiplication triples. So, in the last lecture we had seen the triple transformation protocol, in this lecture we will complete the analysis of the triple transformation protocol.

(Refer Slide Time: 00:41)



So, just to quickly recap what are the required properties from the triple transformation protocol, the input will be a set of $2d + 1$ number of secret shared triples. And these triples need not be multiplication triples and there may not be any relationship among these triples and all these triples are triples shared as per the Shamir secret sharing scheme, where the degree of the sharing is t.

And here d will be at least t and the output for the triple transformation protocol will be another bunch of $2d + 1$ secret share triples, which are now correlated. And the correlation is the following; so first of all we want that the a components of all the output triples should lie on a d-degree polynomial, all the b components of the output triples should lie on a d-degree polynomial and the c component of all the output triples should lie on a $2d$ degree polynomial.

So, I stress that the a component, b component and c components of the output triples they are themselves secret shared, but we want that there the values of those triples they should lie on polynomials A, B and C of degree d, d and $2d$ respectively, that is the first correlation we require from the triple transformation protocol.

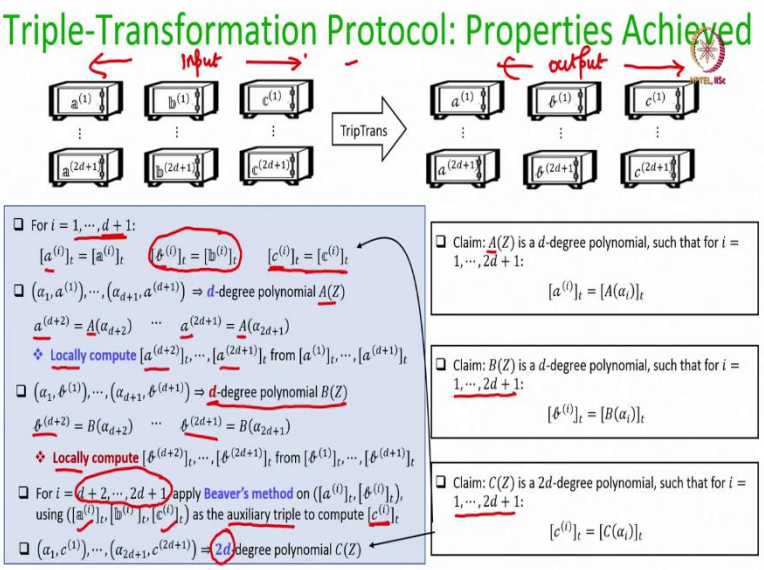
The second correlation that we require is that the C polynomial should be equal to the product of the A and B polynomials if and only if all the input triples are multiplication triples. And this is this should hold in an if and only if fashion, namely if the C polynomial

is the product of A polynomial and B polynomial, then that implies that all the input triples were multiplication triples and vice versa.

That is if all the input triples for multiplication triples, then the C polynomial is the product of A polynomial and B polynomial, which implicitly means that all the output triples are also multiplication triples. And the third property from the security point of view is that if the ith input triple was random for the adversary, then the ith output triple should also be random for the adversary.

Of course, adversary will have t shares for the for each of the triples, both for the input triples as well as for the ith triple, but the property here demands that that if the ith input triple was random, namely the value of the triple was unknown to the adversary, then the value of the ith output triple also should be unknown to the adversary. So, these are the requirements from the triple transformation protocol.

(Refer Slide Time: 04:04)



And in the last lecture we had seen a protocol for triple transformation. So, let me quickly go through the steps of the triple transformation protocol. So, what we do is that we are given $2d + 1$ number of secret shared triples. So, this is the input and the input triples are divided into two groups, the first group consisting of $d + 1$ number of triples.

So, sorry pardon here, this the input is only this bit in the picture and this is the output. So, the idea behind the protocol is that we divide the input triples into two groups, the first

group consisting of the first $d + 1$ secret shared triples and then the second group have the has the remaining d secret shared triples.

Now, using the 1st $d + 1$ secret shared triples we do the following, we define the A and B polynomials. So, we let the A polynomial to be the unique $d + 1$ degree, unique d-degree polynomial passing through the a components of the first $d + 1$ input triples. And then that automatically defines the, that automatically said sets the first $d + 1$ output triples as well. So, the first $d + 1$ output triples set to be the first $d + 1$ input triples.

And then we compute new points on this defined A polynomial in a secret shared fashion because that involves computing a linear function. So, we compute the next set of points on the A polynomial and they constitute the remaining d output triples. So, since these points can be computed as a linear function of the first $d + 1$ points on the A polynomial, all of which are secret shared, it follows from the linearity property of secret sharing, that the parties can now compute a secret sharing of the a components of the remaining output triples.

The same thing we do even for the B polynomial, we take the b components of the first $d + 1$ triples and using those as distinct points we define a d-degree polynomial and then we compute the next set of points on that defined B polynomials. And this will help the parties to locally compute the secret shared b components of the output triples. And now what we do for the C polynomial is the following.

We take the leftover d number of input secret shared triples and use them as the auxiliary triples in the Beaver's method to compute the c component of the last d output triples. And now we have $2d + 1$ number of c components for the output triples, using them we define the C polynomial. So, that was the protocol. And now we want to prove that it achieves all the properties that we had desired for.

So, the first claim here is that the A polynomial that we have defined is a d-degree polynomial such that the a components of all the output triples lie on this A polynomial. And this simply follows from the steps of the protocol. How exactly we have computed the a component of the output triples? Well, the first $d + 1$ at the a component of the first $d + 1$ output triples are same as the a component of the input triples.

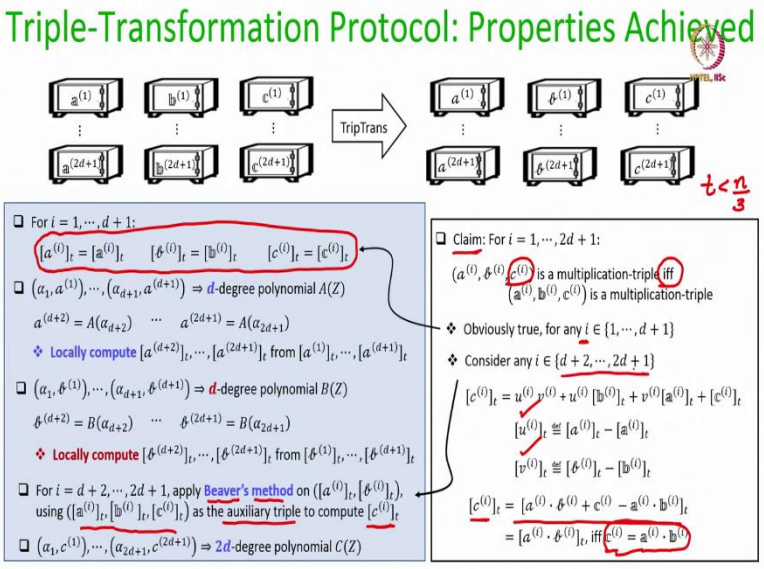
And the a component of the last d output triples are distinct points on the A polynomial. So, that comes from the steps of the protocol the proof is straightforward. The same claim we can make for the B polynomial, the claim here is that the b component of all the output triples lie on a d-degree B polynomial and this again comes from the steps of the protocol.

The b component of the first $d+1$ output triples are same as the b components of the input triples using them we define the B polynomial and this B polynomial will be a unique polynomial because using $d+1$ distinct points we can only define a single d-degree polynomial.

And then the remaining points on the B polynomial constitute the b component of the last d output triples. And then we have the claim that the c component of all the output triples lie on a $2d$ degree C polynomial and this also follows using a similar argument because of the steps of the protocol. Namely, the c component of the first $d+1$ input triples are said to be the c component of the first $d+1$ output triples.

And now we are applying the Beaver's method to compute the c components of the last d triples and now using the c component of all the output triples which are $2d+1$ in number, we are defining the C polynomials. So, using $2d+1$ distinct points we can define a unique $2d$ degree polynomial and that polynomial in this case is the C polynomial. So, all these three claims hold.

(Refer Slide Time: 10:25)



Now, let us prove some other properties. So, we next claim here that for each of the triples namely for i equal to $2d + 1$ if the ith input triple was a multiplication triple, then the ith output triple is a multiplication triple and vice versa. And this holds for any i in the range 1 to $2d + 1$. So, let us take some arbitrary i in the range 1 to $2d + 1$.

So, the claim is obviously true if that i is in the range 1 to $d + 1$, because the first $d + 1$ output triples are said to be the first $d + 1$ input triples. So, if the first $d + 1$ input triples are multiplication triples, then that automatically implies that the first $d + 1$ output triples are also multiplication triple.

So, the claim is obviously true, if my i is either 1 or 2 or $d + 1$, but my i could be different from 1 to $d + 1$, because remember the way I have computed the output triples is that I do some set of actions for the first $d + 1$ output triples and a different set of actions for the last d output triples. So, what I have argued here is that if I focus on the first $d + 1$ output triples, there will be multiplication triples, if and only if the first $d + 1$ input triples are multiplication triples.

Now, consider an i in the range $d + 2$ to $2d + 1$, namely a triple which is different from the first $d + 1$ output triples. My claim I would like to claim that even that output triple will be a multiplication triple if and only if the corresponding input triple was a multiplication triple.

So, for that we have to show that the c_i which we have computed in the c component of the output triple is the product of the a and b component, if and only if for the ith input triple the c component was the product of the a component of the input triple and the b component of the input triple.

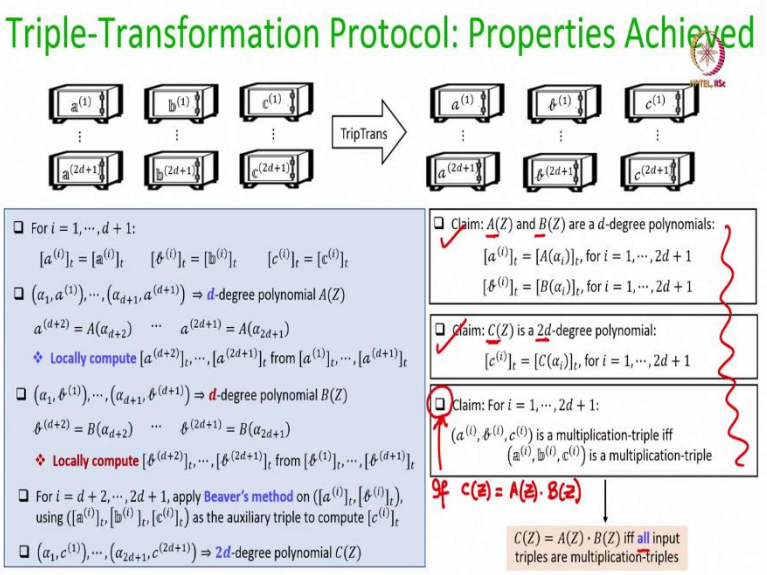
Now, for that we have to see how exactly we have computed as the c component of the ith output triple. We have computed the c component of the ith output triple by applying the Beaver's method, on the a component of the ith output triple and the b component of the ith input triple. And for applying the Beaver's method we have used the ith input triple as the auxiliary triple. Now, as part of the Beaver's method what exactly are the computations which are involved for computing the c component of the ith output triple?

Well, we would have computed an intermediate value u and intermediate value v and would have publicly reconstructed them. The values u and v would have been robustly

reconstructed even if up to t parties produce incorrect shares, because we are in the setting of $t < \frac{n}{3}$. And that is why the error correction will work properly so; that means, everyone will reconstruct the right value of u_i and v_i.

And then as per the Beaver's method, the c component of the ith output triple would be set to this value. Now, this value will be same as the product of the a component and b component of the ith output triple, if and only if the c component of the ith input triple was the product of the a component and b components of the ith input triple. And that shows that this claim is now true, even for an output triple where that output triple is in the range $d + 2$ to $2d + 1$.

(Refer Slide Time: 15:02)



So, we have proved a bunch of claims now. We have proved the claims regarding the degrees of the A, B and C polynomial and we have proved that the ith output triple is a multiplication triple if and only if the ith input triple was the multiplication triple. Now, as a corollary of all these claims, as an implication of all these claims we can claim that the C polynomial is the product of the A and B polynomial if and only if all the input triples are multiplication triples.

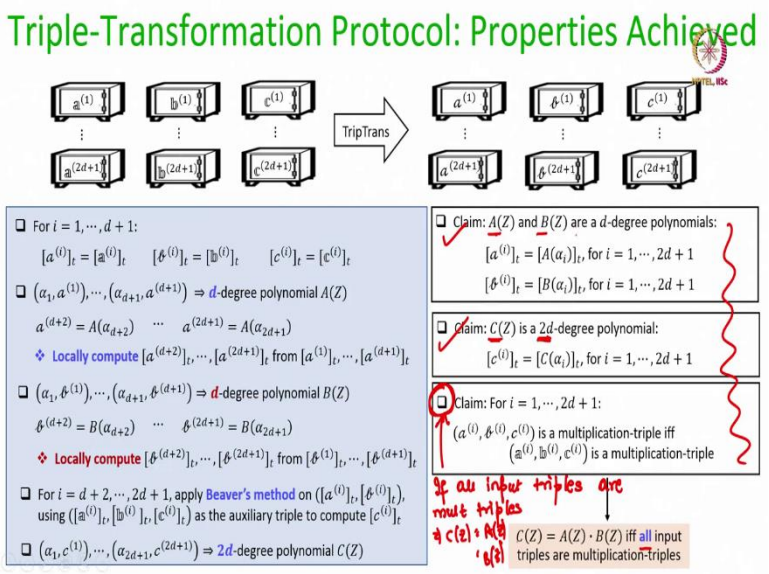
This is because the C polynomial is a $2d$ degree polynomial, A polynomial is a d-degree polynomial and B polynomial is a d-degree polynomial. So, let us prove this implication. So, the there it is since it is an if and only if statement, we have to prove two things. So,

the first thing we must prove is that if the C polynomial is the product of the A and B polynomials, then all the input triples are multiplication triples and that comes from this claim.

Because if the C polynomial is the product of A and B polynomials; that means, the c components of all the output triples is equal to the product of the corresponding a component and b component of the output triples because the points because the a, b and c components of the output triples, they lie on the A, B and C polynomials which followed from the first two claims.

And now if the c component of every output triple is the product of the corresponding a and b components, then because of this last claim it automatically follows that even for the input triples the c component is the product of the a and b component. So, that shows the implication in one direction.

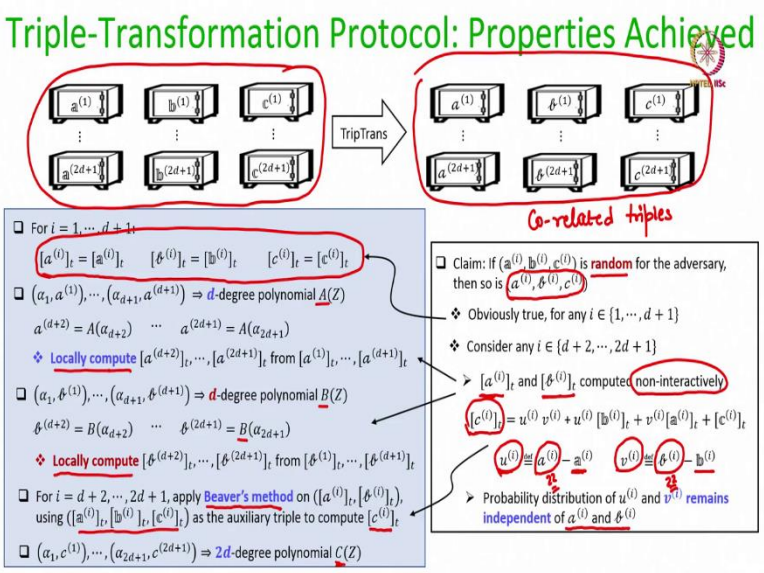
(Refer Slide Time: 17:16)



The other direction thing that we have to prove is that if all input triples are multiplication triples, then we want to claim that C polynomial is the product of A polynomial into B polynomial. And this again comes from the fact that if all the input triples are multiplication triples, then from this third claim this last claim all the output triples are multiplication triples.

And the output triples are nothing but the points on the A, B and C polynomials and C polynomial has degree $2d$ and A and B polynomials have degree d, which automatically implies that the C polynomial will be the product of the A and B polynomials.

(Refer Slide Time: 18:08)



Now, let us prove the last property which we require from the triple transformation protocol. The property that we require is that, if the ith input triple was random for the adversary, random in the sense it does not know the value of a component, b component and c component except that the c component is the product of a and b component.

That means that triple could be any triple over the field, then at the end of the triple transformation protocol even the ith output triple is also randomly distributed for the adversary. Adversary will not know the exact value of the ith output triple, it could be any triple over the field corresponding to which it may have at most t shares. So, again to prove the claim, we have to focus on the index i and there could be two cases depending upon whether the ith triple is one of the triples among the first $d + 1$ output triples or whether the ith triple is one of the triples among the last d output triples.

So, if this i is in the range 1 to $d + 1$, then the claim holds trivially because of the way we have defined the first $d + 1$ output triples. The first $d + 1$ output triples are said to be the first $d + 1$ input triples. So, if any of the triples among the first $d + 1$ input triples is random for the adversary, then that automatically implies that the same triple is when viewed as an output triple is also random for the adversary.

Because for computing the first $d+1$ output triples, we have not done any computation. They are said to be the first $d+1$ input triples only. Whereas, consider the case where i is in this range $d+2$ to $2d+1$. So, in this case what can we say about the a component of the output ith output triple and the b component of the ith output triple?

Well, they are computed non interactively. Why are they computed non interactively? Because they are basically locally computed as a distinct point on the defined A polynomial and as a distinct point on the defined B polynomial and this does not require any interaction among the parties.

So, if there is no interaction happening among the parties; that means, adversary has learned nothing about the resultant a component of the ith output triple and the b component of the ith output triple. But for computing the c component of the ith output triple interaction is involved, because the c component of the ith output triple is computed by applying the Beaver's method where the ith input triple is used as the auxiliary triple.

So, now, let us go into the details of what exactly or the computation involved for computing the c component of the ith output triple. For computing the c component of the ith output triple the parties would have reconstructed two public values, namely the value u_i and value v_i by exchanging shares and applying the Reed Solomon error correction.

But as per our assumption, the ith input triple is random for the adversary because that is the hypothesis of the claim statement. So, if the a component of the ith input triple and the b component of the ith input triple are random for the adversary, then even though the adversary would have learnt the values u_i and v_i it cannot figure out what exactly is the value of the a component of the ith output triple and the b component of the ith output triple.

Because u_i and v_i will serve as the one-time pad encryption of the a component and b components of the ith output triples respectively, where the pads are random, the pads here are the a componen$a$ component and the b component of the ith input triples. And that shows that the probability distribution of this u_i and v_i values which adversary would have learnt will be independent of the a and b components of the ith output triples.

And finally, the c component of the ith output triple is computed locally, once the values u_i and v_i are publicly known, hence adversary does not learn anything additional about

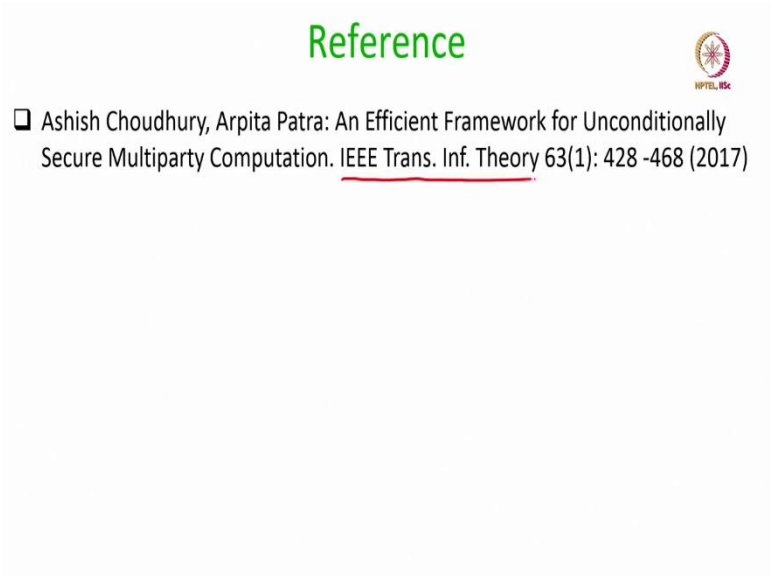
the c component of the ith output triple. It is a randomly distributed value. So, that is all about the triple transformation protocol; that means, we have now proved all the desired properties of the triple transformation protocol, namely the protocol takes a bunch of unrelated triples, which may or may not be multiplication triples.

And transforms them into another bunch of correlated triples, in a very nice way and the correlation is very nice. Namely the correlation is that the a, b and c components of the output triples now constitute distinct points on some well-defined A polynomial, B polynomial and C polynomial such that the C polynomial will be the product of A and B polynomials, if and only if all the input triples are multiplication triples. And I stress here that here all the computations are performed over secret shared values.

So, for the input triples, the values may not be known to any party, they might be secret shared and even for the output triples the values may not be known to any specific parties, but all of them are in secret shared fashion. That means, even the polynomials A, B and C are also not known to any specific party, but they are in a secret shared fashion. That means, every point on the A polynomial, B polynomial and C polynomial is available in a secret shared fashion.

And all this is happening because, we are exploiting the linearity property of secret sharing here.

(Refer Slide Time: 25:03)



So, with that I end this lecture, the reference for this lecture is this paper.

Thank you.