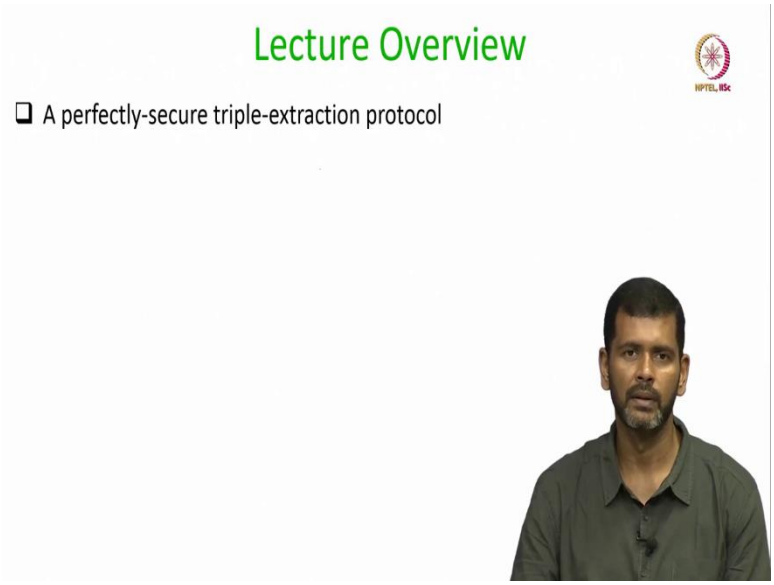
**Secure Computation: Part II**
**Prof. Ashish Choudhury**
**Department of Computer Science and Engineering**
**Indian Institute of Science, Bengaluru**

**Lecture - 53**
**Perfectly-Secure-Triple-Extraction Protocol**

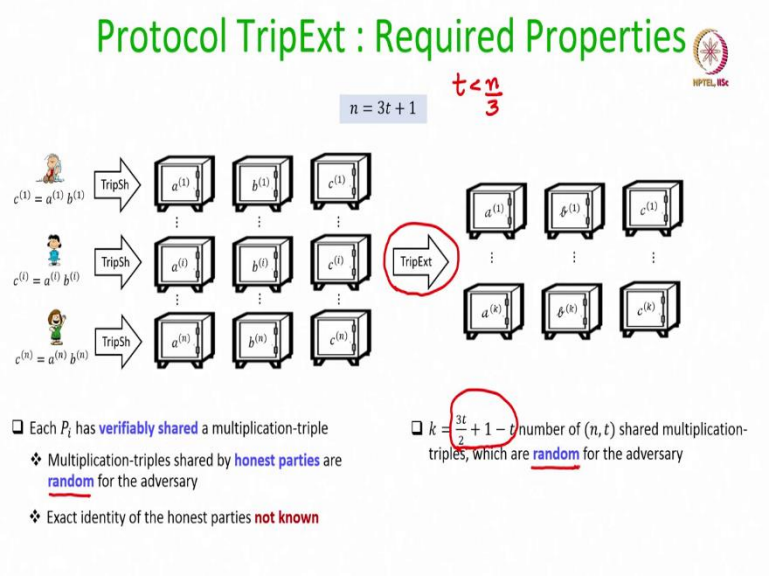
Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:27)




So, in this lecture we will see the instantiation for the second component of our framework for the pre-processing phase protocol namely the Perfectly – Secure – Triple - Extraction Protocol.

(Refer Slide Time: 00:38)




So, what are the required properties? We are in the setting where $t < \frac{n}{3}$. So, we take the case where $n = 3t + 1$that is the smallest value of $n$ satisfying the condition $t < \frac{n}{3}$. And the input scenario for this protocol will be the following it will be guaranteed that each party has verifiably shared a multiplication triplet through an instance of triple sharing protocol.

So, recall that in the last lecture we had seen that if there is a dealer then by running an instance of triple sharing protocol at the end of the triple sharing protocol it will be guaranteed that there is a multiplication triplet which is secret shared on the behalf of that dealer such that if the dealer is honest and no one knows the value of the triplet it is random. Except that adversary will have at most $t$ shares for those triplets which are randomly distributed.

Whereas, even if the dealer is corrupt for the triple sharing instance it will be guaranteed that it has secret shared of multiplication triplet namely whatever is the output of the protocol it will be secret shared multiplication triplet. So, the way this triple extraction protocol will be executed will be the following it will be guaranteed that every party has secret shared verifiable it has verifiably secret shared a multiplication triplet.

And among these $n$ parties there could be up to $t$ corrupt parties, who may not have shared random triplets. That means, those triplets are known to the adversary, but the remaining

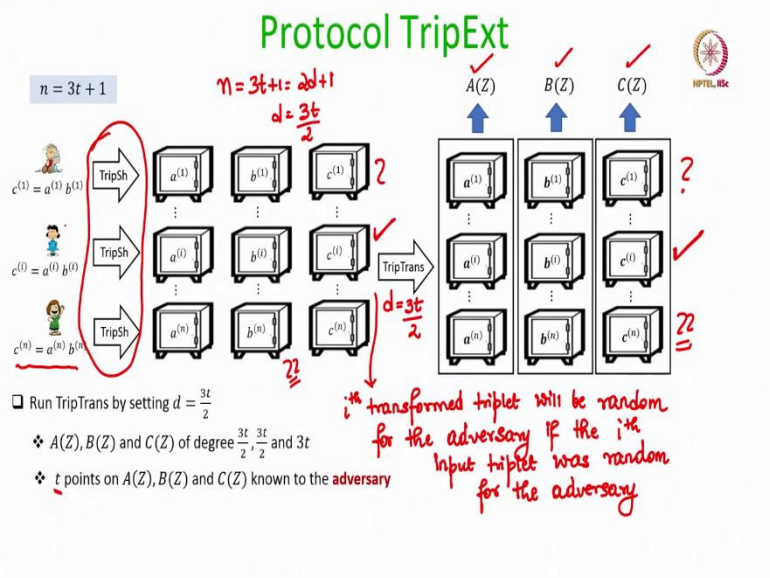
parties are honest and there are at least $n - t$ honest parties in the system whose triplets will be random for the adversary which comes from the privacy property of the triple sharing.

However, the exact identity of the honest parties will not be known we do not know whether the first party is honest or corrupt, we know it has secret shared a multiplication triplet fine. But whether the value of that multiplication triplet is known to the adversary or not we do not know depending depends upon whether the first party is honest or corrupt and no one knows the exact identity of the corrupt parties.

Now, we want to run a triple extraction protocol here which takes this bunch of random and non-random secret share multiplication triplets and output at least sorry which output at exactly $k$ number of secret shared multiplication triplets.

Where the value of $k$ is $\frac{3t}{2} + 1 - t$ such that all this output secret shared triplets are multiplication triplets plus they are guaranteed to be random for the adversary; that means, adversary is guaranteed to not know the value of those triplets it will have only $t$ shares for each of this $k$ output triplets which are randomly distributed.

(Refer Slide Time: 03:44)




That is what we want from the triple extraction protocol. And this triple extraction protocol basically involves just one instance of the triple transformation protocol that is all. So,

notice that here all the input triplets are guaranteed to be multiplication triplets because these triplets are shared by individual dealers using an instance of triple sharing protocol.

And we are in the case where $n = 3t + 1$. So, if I set $n = 3t + 1$ and treat it as $2d + 1$. Then the value of $d$ turns out to be $\frac{3t}{2}$. So, I run the triple transformation protocol by setting $d = \frac{3t}{2}$, because of the triple transformation protocol now there will be a very nice correlation among the transformed secret shared triplets. The correlation will be that there will be polynomials of degree $\frac{3t}{2}$, $\frac{3t}{2}$ and $3t$ namely $A$ polynomial, $B$ polynomial and $C$ polynomial.

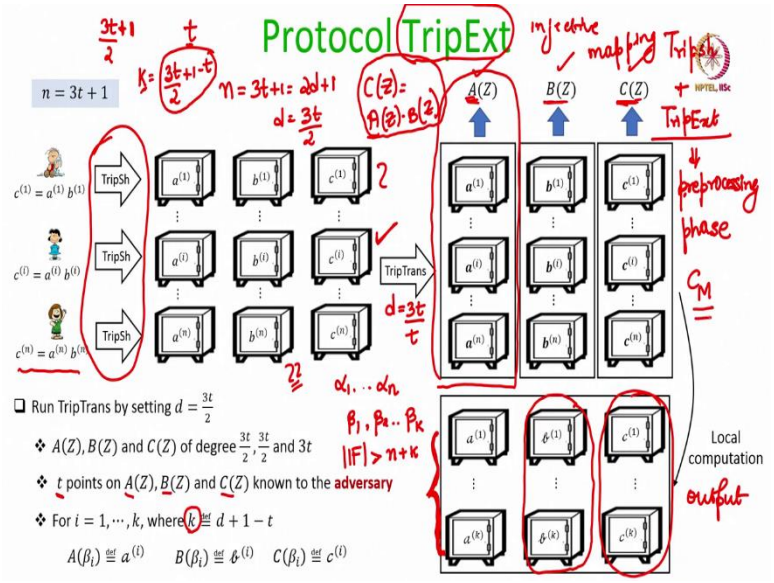
Such that the $a$ component of all the transformed triplets lie on the $A$ polynomial, the $b$ components of all the $b$ components of all the transformed triplets lie on the $B$ polynomial and the $c$ component of all the transformed triplets lie on the $C$ polynomial. Also, it will be guaranteed that at most $t$ points on the $A, B$ and $C$ polynomials are learnt during the triple transformation protocol.

Because the triple transformation protocol has the privacy property which guarantees that the $i$-th transformed triplet the $i$-th transformed triplet will be random for the adversary if the $i$-th input triplet was random for the adversary that is the privacy property of triple transformation right, for that you can refer to our earlier lecture. So, that means, if say the first party was honest the first party $P_1$ was honest.

Then during it is instance of triple sharing whatever triplet is had secret shared at the end of the first instance of triple sharing that will be random for the adversary and as a result of that the first triplet will be first transformed triplet will be random for the adversary. Whereas, suppose the $i$-th dealer was corrupt, but it has secret shared some triplet as a result of that the $i$-th transformed triplet will be known to the adversary.

In the same way suppose the $n$-th dealer was honest; that means, the value of the multiplication triplet shared by the $n$-th party is random for the adversary then the $n$-th transformed triplet will also be random for the adversary. Since these transformed triplets constitute points on the $A, B$ and $C$ polynomials it means that adversary will be knowing at most $t$ points on these triplets corresponding to the $t$ corrupt dealers in the system.

(Refer Slide Time: 07:28)




Now, we also know that since each of this input triplets or multiplication triplets it is guaranteed that the $C$ polynomial is equal to the product of $A$ polynomial into $B$ polynomial. So, what we do here is the following we our goal is to get $k$ number of output triplets those $k$ triplets are obtained by defining $k$ new points on the $A$ polynomial $B$ polynomial and $C$ polynomial.

So, we define the $A$ polynomial at $\beta_1, \ldots, \beta_k$, $B$ polynomial evaluated at $\beta_1, \ldots, \beta_k$ and $C$ polynomial evaluated at $\beta_1, \ldots, \beta_k$. So, we already have the points $\alpha_1, \ldots, \alpha_n$ which are used while defining the $A, B$ and $C$ polynomial s apart from that we also require additional $k$ number of $\beta$ points here which are different from the $\alpha$ points, and this puts an additional restriction on the field.

Now, we require the field to be at least $n + k$. Now, this new points on the $A, B$ and $C$ polynomials can be computed as a linear function on the old points on $A, B$ and $C$ polynomials by old points I mean the transformed triplets. Notice that, the transformed triplets are not available and clear to any specific party they are secret shared. So, since the new points on the $A, B$ and $C$ polynomials can be computed as a linear function of the old points on the $A, B$ and $C$ polynomial.

From the linearity property of the secret sharing, it follows that given the secret sharing of these transformed triplets the parties can locally and non-interactively compute a secret sharing of this new points on the $A, B$ and $C$ polynomials and this is taken as the output for

the protocol. So, we now want to claim that this $k$ secret shared triplets are multiplication triplets, and they are unknown to the adversary.

So, first of all why are they multiplication triplets? Because they lie on the polynomials $A, B$ and $C$ and we know in this case the $C$ polynomial is equal to the product of $A$ and $B$ polynomial because irrespective of whether the dealer of a triple transfer whether the dealer for the triple sharing instance is honest or corrupt all the triplets shared by the individual dealers are guaranteed to be multiplication triplets.

That means at the end of the first instance of triple sharing the triplet $(a_1, b_1, c_1)$ is a multiplication triplet at the end of the $i$-th instance of the triple sharing the triplet $(a_i, b_i, c_i)$ is a multiplication triplet. And at the end of the nth instance of the triple sharing the triplet $(a_n, b_n, c_n)$ is a multiplication triplet; that means, all the inputs for the triple transformation is a bunch of secret shared multiplication triplets.

As a result of that the output for the triple transformation is a bunch of secret shared multiplication triplets and since transformed triplets lie on $A, B$ and $C$ polynomial, the $C$ polynomial is the product of $A$ and $B$ polynomials. And the output triplets of this protocol are also lying on the $A, B$ and $C$ polynomials that automatically implies that all these output triplets are multiplication triplets.

Why they are random for the adversary? Well, at the end of the triple transformation among these transformed triplets, adversary will be knowing at most $k$ of them, we already argued that. And what exactly is the degree of $A$ polynomial? The degree of $A$ polynomial is $\frac{3t}{2}$. So, the degree is $\frac{3t}{2}$; that means, we need $\frac{3t}{2} + 1$ number of points to uniquely identify the $A$ polynomial among them adversary will have only knowledge about $t$ points.

Corresponding to the $t$ transformed triplets; that means, how many points are unknown for the adversary, it is $k$ which is $\frac{3t}{2} + 1 - t$ these many points in the $A$ polynomial are unknown for the adversary and there could be any unknown points because they correspond to the $a$ components of the triplets shared by the honest dealer which at the end after transformation also remain random for the adversary.

Now, whatever value those $k$ number of unknown $a$ components can take that determine the $a$ components of this $k$ output triplets; that means, there is a 1 to 1 mapping, there is

an injective mapping between the unknown $a$ components here in the $A$ polynomial which are not known to the adversary and this output $a$ components.

And since the unknown components are random, they are randomly distributed. Of course, adversary will have $t$ shares for each of them, but they are randomly distributed. It does not help the adversary to find out the exact values of those unknown components that automatically implies that even this $k$ number of output values are random for the adversary except that adversary has $t$ shares for each of them.

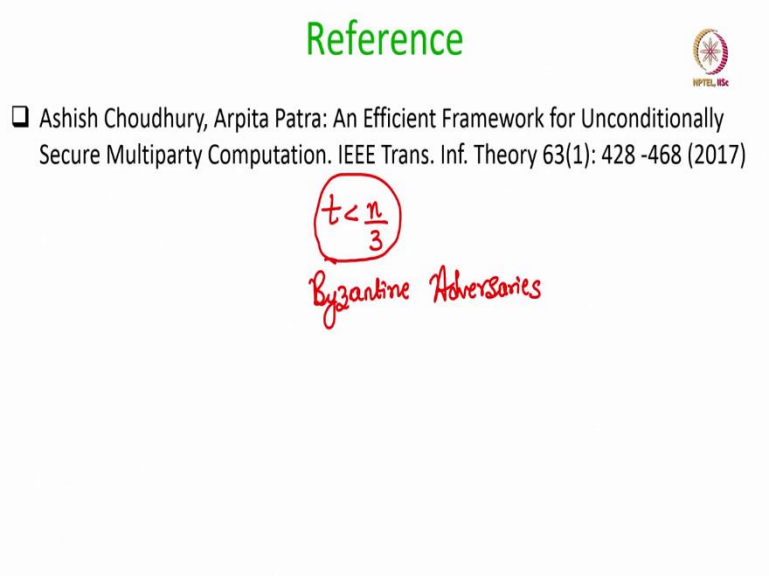
The same argument holds for the $B$ polynomial even for the $B$ polynomial its degree is $\frac{3t}{2} + 1$ adversary will be knowing at most $t$ points on this $B$ polynomial. The remaining $k$ points corresponding to the honest dealer are unknown on this $B$ polynomial for the adversary they could be any random point and as a result of that the $b$ component of this output triplets will be random for the adversary.

And same argument holds for $C$, since $A$ and $B$ polynomials are since there are $k$ number of points on the $A$ and $B$ polynomials which are random for the adversary that automatically implies that even the corresponding $k$ points on the $C$ polynomial are random for the adversary. And as a result of this the $k$ number of $c$ components here which are obtained as the output will be random for the adversary.

So, that proves the privacy property of the triple extraction protocol, and you can see that it is a perfectly secure protocol because all the properties here are achieved in an error free fashion against the computationally unbounded adversary. And now we have the instantiation for the triple sharing gadget as well as the triple extraction gadget. And in one of our earlier lectures, we had seen that if you have the triple sharing protocol and if you have the triple extraction protocol.

Then by combining them by stitching them you can obtain a pre-processing phase protocol, that part we had already discussed in the in one of our earlier lectures. That means, now you have a perfectly secure protocol for the pre-processing phase by running which the parties can generate secret sharing of $c_M$ number of random multiplication triplets, where $c_M$ is the number of multiplication gates in the circuit.

(Refer Slide Time: 16:00)




That completes our discussion on perfectly secure MPC, now we have a perfectly secure MPC with $t < \frac{n}{3}$ tolerating Byzantine adversaries. And we had already argued that $t < \frac{n}{3}$ is the necessary condition for any generic perfectly secure MPC because there are some specific tasks such as perfectly secure Byzantine agreement where the condition $t < \frac{n}{3}$ is necessary. That shows that under condition $t < \frac{n}{3}$ is necessary for any generic perfectly secure MPC computation protocol.

Thank you.