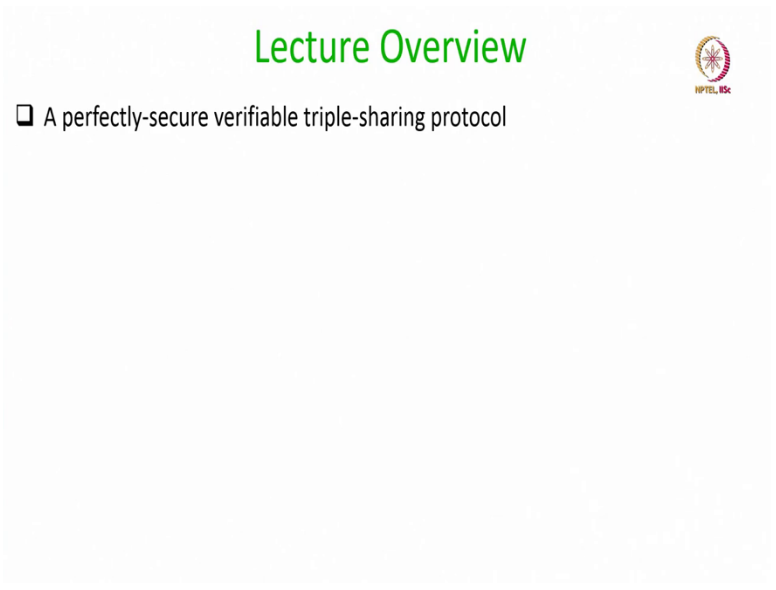
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 52
Perfectly-Secure Verifiable Triple-Sharing Protocol

Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:26)

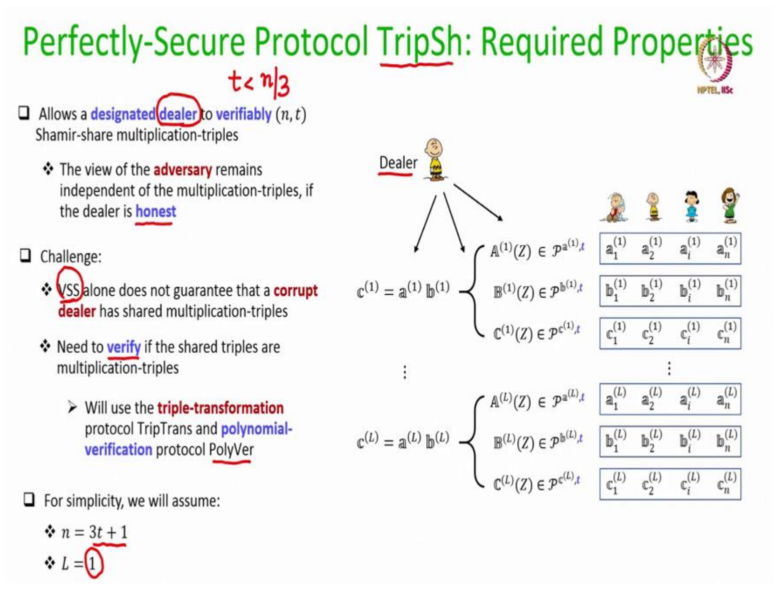


Lecture Overview

- A perfectly-secure verifiable triple-sharing protocol

So, in this lecture we will now see the exact instantiation of the components for our pre-processing phase protocol. So, we will start with the instantiation of the Perfectly-Secure Verifiable Triple Sharing Protocol.

(Refer Slide Time: 00:41)



Perfectly-Secure Protocol TripSh: Required Properties

$t < n/3$

- Allows a designated dealer to verifiably (n, t) Shamir-share multiplication-triples
  - The view of the adversary remains independent of the multiplication-triples, if the dealer is honest
- Challenge:
  - VSS alone does not guarantee that a corrupt dealer has shared multiplication-triples
  - Need to verify if the shared triples are multiplication-triples
    - Will use the triple-transformation protocol TripTrans and polynomial-verification protocol PolyVer
- For simplicity, we will assume:
  - $n = 3t + 1$
  - $L = 1$

NPTEL, IISc
Dealer
$\mathbb{c}^{(1)} = \mathbb{a}^{(1)}\mathbb{b}^{(1)}$
$\mathbb{A}^{(1)}(Z) \in \mathcal{P}^{\mathbb{a}^{(1)},t}$
$\mathbb{B}^{(1)}(Z) \in \mathcal{P}^{\mathbb{b}^{(1)},t}$
$\mathbb{C}^{(1)}(Z) \in \mathcal{P}^{\mathbb{c}^{(1)},t}$
$\mathbb{a}_1^{(1)}$ $\mathbb{a}_2^{(1)}$ $\mathbb{a}_i^{(1)}$ $\mathbb{a}_n^{(1)}$
$\mathbb{b}_1^{(1)}$ $\mathbb{b}_2^{(1)}$ $\mathbb{b}_i^{(1)}$ $\mathbb{b}_n^{(1)}$
$\mathbb{c}_1^{(1)}$ $\mathbb{c}_2^{(1)}$ $\mathbb{c}_i^{(1)}$ $\mathbb{c}_n^{(1)}$
$\mathbb{c}^{(L)} = \mathbb{a}^{(L)}\mathbb{b}^{(L)}$
$\mathbb{A}^{(L)}(Z) \in \mathcal{P}^{\mathbb{a}^{(L)},t}$
$\mathbb{B}^{(L)}(Z) \in \mathcal{P}^{\mathbb{b}^{(L)},t}$
$\mathbb{C}^{(L)}(Z) \in \mathcal{P}^{\mathbb{c}^{(L)},t}$
$\mathbb{a}_1^{(L)}$ $\mathbb{a}_2^{(L)}$ $\mathbb{a}_i^{(L)}$ $\mathbb{a}_n^{(L)}$
$\mathbb{b}_1^{(L)}$ $\mathbb{b}_2^{(L)}$ $\mathbb{b}_i^{(L)}$ $\mathbb{b}_n^{(L)}$
$\mathbb{c}_1^{(L)}$ $\mathbb{c}_2^{(L)}$ $\mathbb{c}_i^{(L)}$ $\mathbb{c}_n^{(L)}$

So, just to recap what exactly we want from the triple sharing protocol TripSh. So, it is a special type of VSS protocol, where there will be a designated dealer, whose inputs will be now multiplication-triples. So, we want the dealer to secret share L number of multiplication-triplets, ok, where L is some parameter.

And, we want the privacy and the commitment properties here. The privacy property demands that, if the dealer is honest, then the view of the adversary should be independent of the secret shared multiplication-triples; that means adversary will get t shares for each of the triples for each of the multiplication-triples. But the exact value of those multiplication-triples should be unknown to the adversary.

It could be as if any multiplication triplet from the field has been secret shared by the dealer. So, that is the first requirement. And, we want the commitment property here as well, which demands that even if the dealer is corrupt, it should be guaranteed that it has secret shared only multiplication-triplets and not arbitrary triplets.

That means it should be guaranteed that the c component of each of the triplets which dealer has secret shared is indeed equal to the product of the corresponding a and b components, right. So, that is a challenge here, that is a challenge in this triple secret sharing protocol; because just using the polynomial based verifiable secret sharing is not sufficient here.

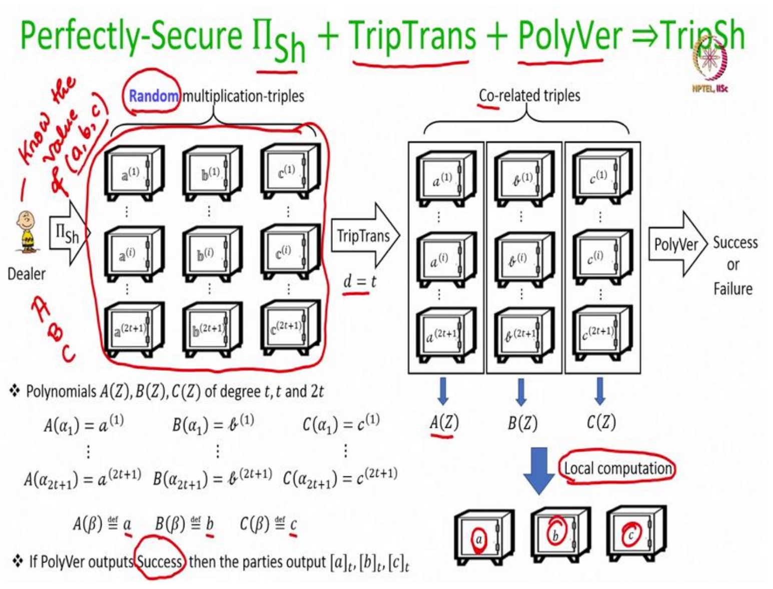
Because using the polynomial based VSS, it will be guaranteed that dealer has secret shared triplets of values using t degree polynomials. But, it may not secret share multiplication-triplets. So, on top of verifiable secret sharing, we need to add a layer of verification to verify that indeed the secret shared triplets are multiplication-triplets. And, the challenge here is that, this additional layer of verification should respect the privacy of dealer's triplets if the dealer is honest.

So, intuitively what we are going to do here is that, we are going to use the triple-transformation protocol coupled with the polynomial verification protocol which we had discussed in our earlier lectures on top of the VSS to verify whether dealer has secret shared multiplication-triplets or not.

For simplicity, for the purpose of explanation, I will assume n=3t+1, which is the smallest value of n satisfying the condition, t < n/3. Remember, this is the optimal resilience for any generic perfectly secure multi-party computation protocol. And, also, for the sake of demonstration, I will explain the protocol assuming that dealer wants to verifiably secret share, only one multiplication-triplet.

Of course, if dealer wants to secret share more than one multiplication-triplet the protocol steps can be executed in parallel.

(Refer Slide Time: 04:10)



Perfectly-Secure Π_{Sh} + TripTrans + PolyVer ⇒TripSh

Know the value of (a, b, c)
Random multiplication-triples
Co-related triples
Dealer
Π_{Sh}
A B C
$a^{(1)}$ $b^{(1)}$ $c^{(1)}$
$a^{(i)}$ $b^{(i)}$ $c^{(i)}$
$a^{(2t+1)}$ $b^{(2t+1)}$ $c^{(2t+1)}$
TripTrans
$d = t$
PolyVer
Success or Failure
$A(Z)$ $B(Z)$ $C(Z)$
Local computation
a b c
NPTEL, IISc

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

$$A(\alpha_1) = a^{(1)} \quad B(\alpha_1) = b^{(1)} \quad C(\alpha_1) = c^{(1)}$$
$$\vdots$$
$$A(\alpha_{2t+1}) = a^{(2t+1)} \quad B(\alpha_{2t+1}) = b^{(2t+1)} \quad C(\alpha_{2t+1}) = c^{(2t+1)}$$
$$A(\beta) \stackrel{\text{def}}{=} a \quad B(\beta) \stackrel{\text{def}}{=} b \quad C(\beta) \stackrel{\text{def}}{=} c$$

- If PolyVer outputs Success then the parties output $[a]_t, [b]_t, [c]_t$

So, here is the way we can combine any polynomial-based VSS with the triple-transformation protocol and the polynomial verification protocol to get our triple sharing protocol. To begin with dealer will secret share a bunch of random multiplication-triplets. So, remember the goal of the dealer is to secret share one multiplication-triplets at the end of the protocol.

But we also want a verification process to be incorporated to enable the parties to verify whether dealer has behaved correctly or not. Dealer actually secret shares a bunch of multiplication-triplets; out of this bunch of multiplication-triplets after performing verification etcetera, we will be left with only one secret shared multiplication-triplet which will be considered as the secret shared triplet on the behalf of the dealer, ok.

So, in this specific case dealer has secret shared 2t + 1 number of arbitrary and random multiplication-triplets. If the dealer is honest, then each of the triplets is indeed a multiplication-triplet, but if the dealer is corrupt that need not be the case. So, we have to now incorporate our verification mechanism, once the dealer has secret shared its triplets.

So, once the dealer has secret shared its triplets, its done; it cannot now change its mind. Now, we have to go; we means, the parties have to go and now they have to verify whether dealer has indeed secret shared multiplication-triplets. So, what we first do is the following.

The parties execute the triple-transformation protocol on the triplets shared by the dealer by setting d to be 2t, ok. So, recall that in the triple-transformation protocol there is a set of secret shared triplets, which may or may not be multiplication-triplets and they are transformed into another bunch of secret shared triplets which are correlated, in the sense there will be an A polynomial, there will be a B polynomial, there will be a C polynomial; where the C polynomial will be the product of the A and B polynomials if all the input triplets are multiplication-triplets. So, that is what we are going to do here. So, we apply that triple-transformation protocol here, we mean, the parties execute a triple-transformation protocol here on the set of secret shared triplets of the dealer. As a result of that, they will now have 2t + 1 number of secret shared triplets.

And, they are correlated. They are correlated in the sense, that there will be a t degree A polynomial; there will be a t degree B polynomial and there will be a 2 t degree C polynomial, ok. Such that, the a components of all the transformed triplets lie on the A polynomial; the b components of all the transformed triplets lie on the B polynomial and the c component of c components of all the transformed triples triplets lie on the C polynomial, ok.

Now, let us define another point on this defined A, B and C polynomials namely, the A polynomial at beta we defined to be a; the B polynomial defined at beta let us call it b and the C polynomial defined at beta a C polynomial evaluated at beta let us call it c. Notice that, the dealer will know the value of this triplet a, b, c. Why it will know the value of this triplet a, b, c? Because, it will know the polynomials A, B and C.

Because, this A, B and C polynomials they are computed deterministically as a function of all the triplets shared by the dealer during this triple-transformation protocol. So, since all the triplets shared by the dealer are known to the dealer and it knows the steps of the triple-transformation we can say that dealer is aware of this triplet a, b, c.

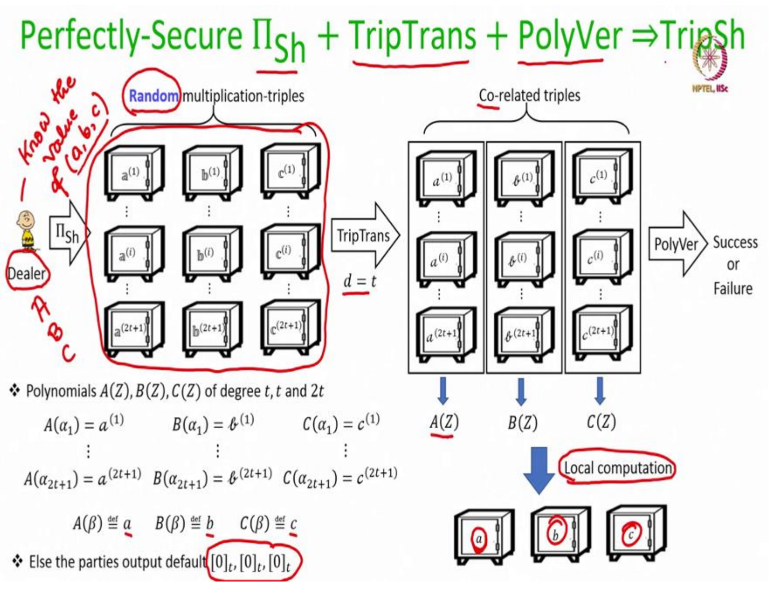
Now, once the triplets are transformed into the correlated triplets, we have this A, B and C polynomials. And, now the goal is to check whether the C polynomial is the product of A and B polynomials. And, for that, the parties execute an instance of polynomial verification protocol PolyVer, the output will be either success or failure for the honest parties.

If the output of the polynomial verification protocol is success, then the parties output a secret sharing of this triplet a, b, c. And, they can do that because, a secret sharing of a, b, c can be computed as a linear function of the secret shared triplets which are obtained at the end of triple-transformation; because the a component of this triplet a, b, c is nothing but, a point on the A polynomial.

So, the value a is nothing but a Lagrange interpolation function of the a-component of all the transformed triplets which is a linear function. So, since a-component of all the transformed triplets are secret shared, by applying the same Lagrange interpolation function on the secret sharing of the a-component of all the transformed triplets, the parties will get a secret sharing of the a-component of the triplet a, b, c.

Similarly, they can obtain a secret sharing of the b-component of this triplet a, b, c and the secret sharing of the c-component of this triplet a, b, c.

(Refer Slide Time: 10:26)



Perfectly-Secure Π_{Sh} + TripTrans + PolyVer ⇒ TripSh

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

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

⋮

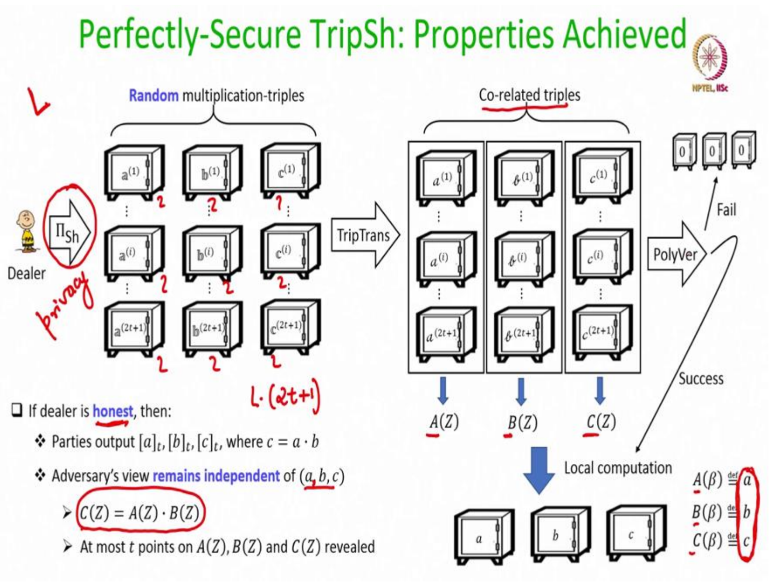
$A(\alpha_{2t+1}) = a^{(2t+1)}$ $B(\alpha_{2t+1}) = b^{(2t+1)}$ $C(\alpha_{2t+1}) = c^{(2t+1)}$

$A(\beta) \overset{\text{def}}{\equiv} a$ $B(\beta) \overset{\text{def}}{\equiv} b$ $C(\beta) \overset{\text{def}}{\equiv} c$

❖ Else the parties output default $[0]_t, [0]_t, [0]_t$

Whereas if the, output of the polynomial verification is failure and, looking ahead, if the output of the polynomial verification is failure, then the dealer is corrupt; we are going to claim that later. Then the parties safely take a default sharing of the triplet 0 0 0 on the behalf of the dealer. That means, the parties set the output assuming that dealer wanted to secret share the triplet 0 0 0. So, that is the triple sharing protocol.

(Refer Slide Time: 11:00)



Perfectly-Secure TripSh: Properties Achieved

Random multiplication-triples
Co-related triples
Dealer
Π_{Sh}
privacy
TripTrans
PolyVer
Fail
Success
$L \cdot (2t+1)$
$A(Z)$
$B(Z)$
$C(Z)$
Local computation
$A(\beta) \overset{\text{def}}{\equiv} a$
$B(\beta) \overset{\text{def}}{\equiv} b$
$C(\beta) \overset{\text{def}}{\equiv} c$

❑ If dealer is honest, then:

❖ Parties output $[a]_t, [b]_t, [c]_t$, where $c = a \cdot b$

❖ Adversary's view remains independent of (a, b, c)

➢ $C(Z) = A(Z) \cdot B(Z)$

➢ At most t points on $A(Z), B(Z)$ and $C(Z)$ revealed

Now, let us prove each of the properties which we desire from this triple sharing protocol. So, let us first prove a bunch of properties if the dealer is honest. So, remember, this is a dealer specific protocol and we have a bunch of requirements for an honest dealer and we have another bunch of requirements for a corrupt dealer. So, let us prove the requirements for an honest dealer. So, if the dealer is honest, then the claim is that the parties do not output the default sharing of 0.

But rather they output the secret shared a, b, c triplet where the triplet a, b, c will be a multiplication-triplet such that, the adversary's view will be independent of this triplet a, b, c; that means, adversary will have at most t shares for this triplet a, b, c which are going to be randomly distributed over the field.

And this comes from the fact that, since all the triplets shared by an honest dealer will be multiplication-triplet then, from the property of triple-transformation it follows that all the correlated triplets will also be multiplication-triplets; that means, the C polynomial will be the product of A and B polynomials, right. And, during the triple-transformation, the adversary will not learn anything about the A, B and C polynomials.

Because, none of the triplets shared by the dealer will be known; that comes from the privacy property of the verifiable secret sharing, right. So, the privacy property of the verifiable secret sharing guarantees that since the dealer is honest, the adversary will not know any of the secret shared triplets, the input triplets.

And as a result of that, since none of the triplets shared by the dealer is known to the adversary, the privacy property of triple-transformation guarantees that none of the transformed triplets is also known to the adversary; that means, they are randomly distributed, adversary will just have t shares for each of the transform triplets. But, during the polynomial verification process, at most t of these transformed triplets will be learned by the adversary during the designated verification by t corrupt verifiers.

And that is all; that means, throughout the protocol adversary will learn at most t points on these A, B and C polynomials. But, the degree of the A polynomial, degree of the B polynomial is t; that means, there is one degree of freedom from the viewpoint of the adversary, ok; that means, definitely there is at least one triplet among these correlated triplets which is not known to the adversary, ok, at the end of the polynomial verification process.

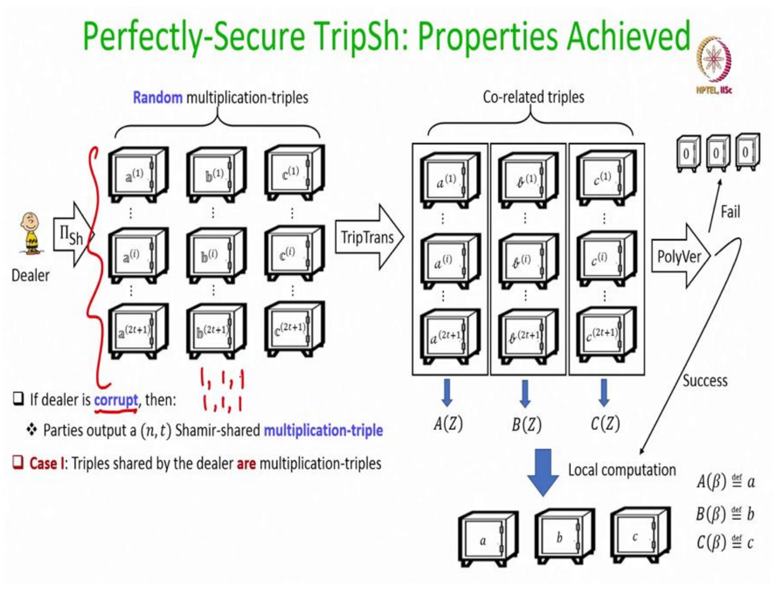
And as a result of that, the value of the A, B and C polynomials at beta is also randomly distributed for the adversary. Namely, for every candidate triplet among these correlated triplets, which is unknown to the adversary at the end of polynomial verification there is a corresponding a, b, c triplet which will be consistent with the view of the adversary, whatever shares adversary has received.

And, why this triplet a, b, c will be a multiplication-triplet? Because that triplet constitute distinct points on the A, B and C polynomials and we know that for an honest dealer the C polynomial is the product of A and B polynomials, ok. So, we have proved that if the dealer is honest, then at the end of the protocol there is one secret shared triplet known to the dealer which is a multiplication-triplet, which is random for the adversary and which is secret shared triplet.

So, you can see that at the end of the protocol it is only 1 output which is the final output, namely the secret shared triplet a, b, c even though dealer started the protocol with a bunch of 2 t plus 1 triplets. So, dealer has secret shared 2 t plus 1 triplets; all of them are transformed, some of them are verified, some means all of them are verified in the verification process, the privacy of some of the triplets is lost. That is why at the end of the protocol its only one triplet, which is considered to be secret shared on the behalf of the dealer.

That means, if dealer wants to secret shared L number of multiplication-triplets, then it has to do this process L number of times; that means, it will secret share actually L times 2t+1 number of multiplication-triplets, then there will be L batches of triple-transformation protocol and L batches of polynomial verification process and so on.

(Refer Slide Time: 16:19)


Perfectly-Secure TripSh: Properties Achieved

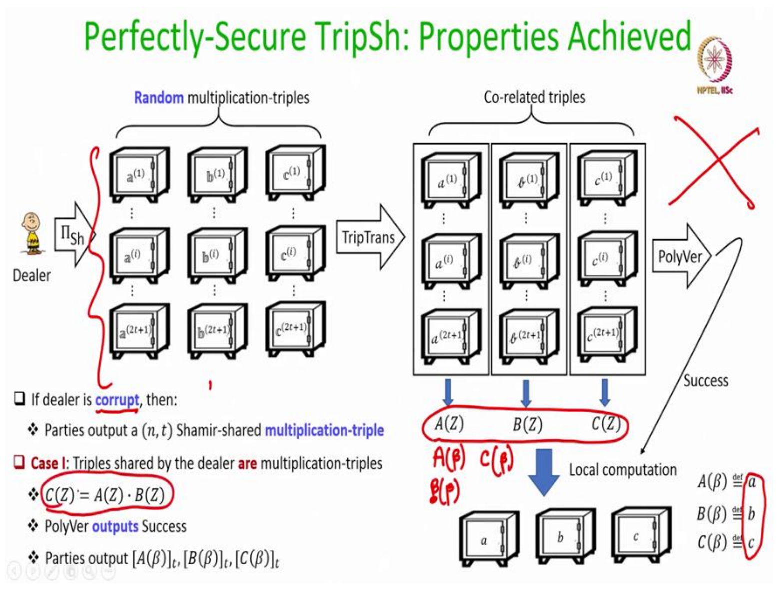
- If dealer is corrupt, then:
  - Parties output a (n, t) Shamir-shared multiplication-triple
- Case I: Triples shared by the dealer are multiplication-triples

Random multiplication-triples
Co-related triples
Dealer
Π_{Sh}
TripTrans
PolyVer
Fail
Success
$A(Z)$
$B(Z)$
$C(Z)$
Local computation
$A(\beta) \overset{\text{def}}{=} a$
$B(\beta) \overset{\text{def}}{=} b$
$C(\beta) \overset{\text{def}}{=} c$
NPTEL, IISc

Now, let us claim the properties achieved by this triple sharing protocol for a corrupt dealer. We want to claim here that even if the dealer is corrupt at the end of the protocol there will be a secret shared multiplication-triplet which will be taken as the output on the behalf of the dealer.

And now there are two possible cases depending upon whether the polynomial verification protocol fails or it provides the output success, which further depends upon whether the triplets shared by the dealer using the verifiable secret sharing are multiplication-triples or not. So, if the triplets shared by the dealer are multiplication-triplets; that means, even though dealer is corrupt, but it is careful in the sense that during the VSS instances while secret sharing the triplets it has secret shared multiplication-triplets. They need not be random; because, it is a corrupted dealer. So, for instance it may always secret share 1 1 1, all the triplets it might secret share as 1 1 1, that is fine. Our goal will be to ensure that at the end of the protocol the value a, b, c which is considered as the output for a corrupted dealer turns out to be a multiplication-triplet.

(Refer Slide Time: 17:44)



Perfectly-Secure TripSh: Properties Achieved

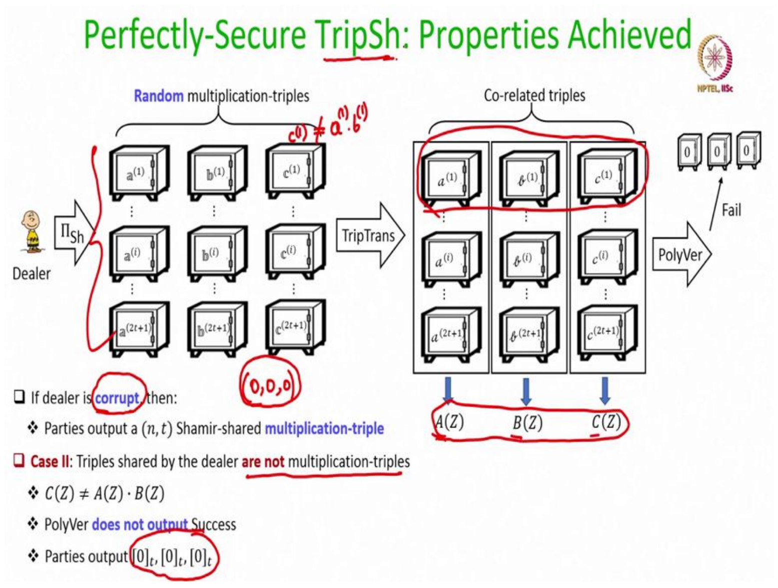
- If dealer is corrupt, then:
  - Parties output a (n, t) Shamir-shared **multiplication-triple**
- **Case I**: Triples shared by the dealer **are** multiplication-triples
  - $C(Z) = A(Z) \cdot B(Z)$
  - PolyVer **outputs** Success
  - Parties output $[A(\beta)]_t, [B(\beta)]_t, [C(\beta)]_t$

NPTEL, IISc
Random multiplication-triples
Co-related triples
Dealer
Π_{Sh}
$a^{(1)}$ $b^{(1)}$ $c^{(1)}$
$a^{(i)}$ $b^{(i)}$ $c^{(i)}$
$a^{(2t+1)}$ $b^{(2t+1)}$ $c^{(2t+1)}$
TripTrans
PolyVer
Success
$A(Z)$ $B(Z)$ $C(Z)$
A(β) C(β) B(β)
Local computation
a b c
$A(\beta) \overset{\text{def}}{=} a$
$B(\beta) \overset{\text{def}}{=} b$
$C(\beta) \overset{\text{def}}{=} c$

So, if all the triples shared by the dealer are multiplication-triples, then the triple-transformation protocol guarantees that the C polynomial is the product of the A and B polynomials. And as a result of that, the polynomial verification protocol will output success because there will be no genuine complaints against the dealer. And, if the output of the polynomial verification is success, then no one will take this route, no one will output a default secret sharing of 0 the triplet 0 0 0 on the behalf of the dealer.

Everyone will take the secret sharing of the values a, b, c. And, a, b, c will constitute a multiplication-triplet because a is nothing but, the A polynomial evaluated at beta; b is nothing but, the B polynomial evaluated at beta and the value c is nothing but, the C polynomial evaluated at beta and we have already argued that we are in the case where the C polynomial is the product of A and B polynomials.

(Refer Slide Time: 18:49)



Perfectly-Secure TripSh: Properties Achieved

Random multiplication-triples
Co-related triples
$c^{(1)} \neq a^{(1)} \cdot b^{(1)}$
Dealer
Π_{Sh}
TripTrans
PolyVer
Fail
$a^{(1)}$ $b^{(1)}$ $c^{(1)}$
$a^{(i)}$ $b^{(i)}$ $c^{(i)}$
$a^{(2t+1)}$ $b^{(2t+1)}$ $c^{(2t+1)}$
$(0,0,0)$
$A(Z)$ $B(Z)$ $C(Z)$
0 0 0
NPTEL, IISc

- If dealer is corrupt, then:
  - Parties output a (n, t) Shamir-shared multiplication-triple
- Case II: Triples shared by the dealer are not multiplication-triples
  - $C(Z) \neq A(Z) \cdot B(Z)$
  - PolyVer does not output Success
  - Parties output $[0]_t, [0]_t, [0]_t$

The other case could be that, a corrupt dealer has not secret shared multiplication-triples ok. That means, during the VSS instances it is supposed to secret shared multiplication-triples but, suppose, it has not secret shared multiplication-triples. Say, for instance, the first triplet shared by the dealer is not multiplication triple; that means, c1 is not equal to a1 into b1.

If that is the case, then we know that at the end of the triple-transformation protocol, the C polynomial will not be equal to the product of A and B polynomial; because triple-transformation protocol will then guarantee that the first transformed triplet is not a multiplication-triplet.

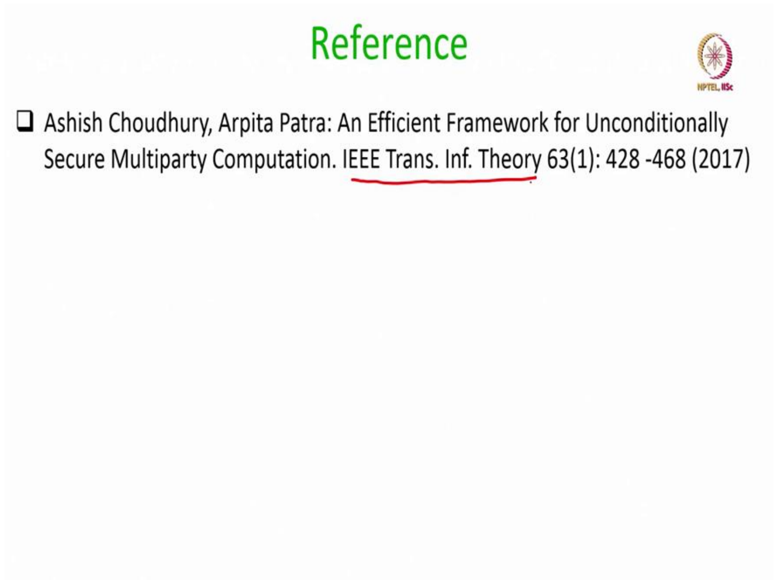
But, the a-component, b-component and c-component of all the transformed triplets will be guaranteed to be lying on A, B and C polynomials respectively. And, since the first transformed triplet is not a multiplication-triplet; that means, the C polynomial at alpha 1 is not equal to the B polynomial at alpha 1 times the A polynomial at alpha 1.

Now, since, the A, B and C polynomials further go through the polynomial verification process, and in this case the C polynomial is not equal to the product of A and B polynomials, the output of the polynomial verification protocol will be failure, ok, it would not be this route. As a result, the parties will go and follow the route under failure condition and under the failure condition, the parties are supposed to output a default secret sharing of the triplet 0 0 0.

Namely, the parties everyone will output, the shares 0 0 0 on the behalf of the dealer. And it is easy to see that 0 0 0 indeed constitute, constitutes a multiplication-triplets. Well, it is not random; because everyone will know that dealer as secret shared 0 0 0. And, everyone will know the value of the triplet.

But that is fine because, we are not claiming anything regarding the random or non-random nature of the triplet shared on the behalf of a corrupt dealer, ok. So, that shows that all the claimed properties of the triple sharing protocol are achieved in a perfectly secure way even in the presence of a computationally unbounded adversary and in an error free fashion.

(Refer Slide Time: 21:30)



Reference

- 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. Again, the reference for this lecture is this IEEE Transaction Paper.

Thank you.