


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

Lecture - 51
Perfectly-Secure Protocol for Verifying Multiplicative Relationship

(Refer Slide Time: 00:26)

Lecture Overview

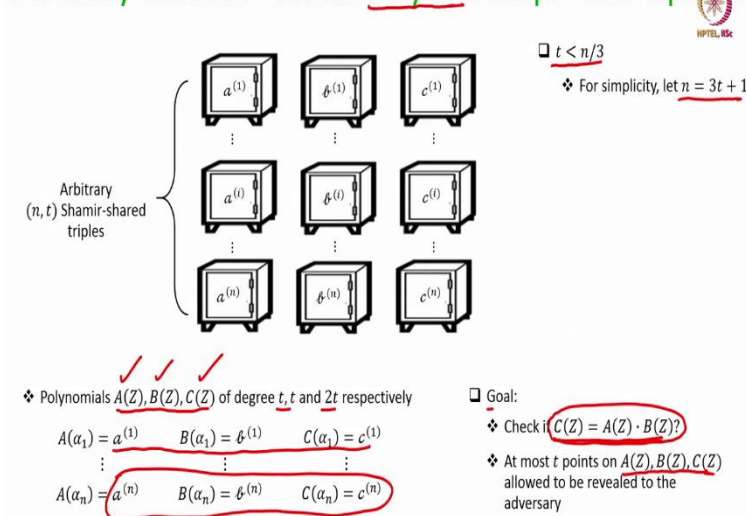


- ☐ A perfectly-secure protocol for verifying multiplicative relationship among polynomials

Hello everyone, welcome to this lecture. So, in this lecture we will see another protocol another Perfectly-Secure Protocol which will be useful later for getting our preprocessing phase protocol to generate secret shared random multiplication triples. So, this protocol is for Verifying Multiplicative Relationship among three polynomials.

(Refer Slide Time: 00:48)

Perfectly-Secure Protocol PolyVer: Required Properties



This protocol is called as the PolyVer protocol, Poly stands for polynomial and Ver stands for verification, and this is a perfectly secure protocol. Later, we will see a PolyVer protocol where there might be a negligible error involved. But this is a perfectly secure protocol where the verification happens in an absolutely error free fashion. So, what exactly is the input for this polynomial verification protocol?

We will have three polynomials an A polynomial and B polynomial and C polynomial. The degrees of A and B polynomial will be t whereas, the degree of the C polynomial will be $2t$ and there will be n points on the A, B and C polynomials which will be secret shared. What you can imagine is that you are given here n number of secret shared triples.

The a components of all the secret shared triples lie on a t degree polynomial A , the b components of all these secret shared triples lie on a t degree polynomial B and the c components of all these secret shared triples lie on a $2t$ degree polynomial C . The input triples I stress need not be multiplication triples.

So, for instance c_1 may not be equal to $a_1 \cdot b_1$ and so on that is the input that is the input for this protocol. And here the relationship among t and n will be the following, $t < \frac{n}{3}$ and for simplicity we will be focusing on the case where $n = 3t + 1$ that is the that is the smallest value of n satisfying the condition $t < \frac{n}{3}$.

Now, what is the goal of the protocol? What exactly is the output? Now, the goal here is to check whether the C polynomial is the product of A and B polynomials, now you might be wondering what the big deal in that is, why cannot we simply reconstruct all these triples right.

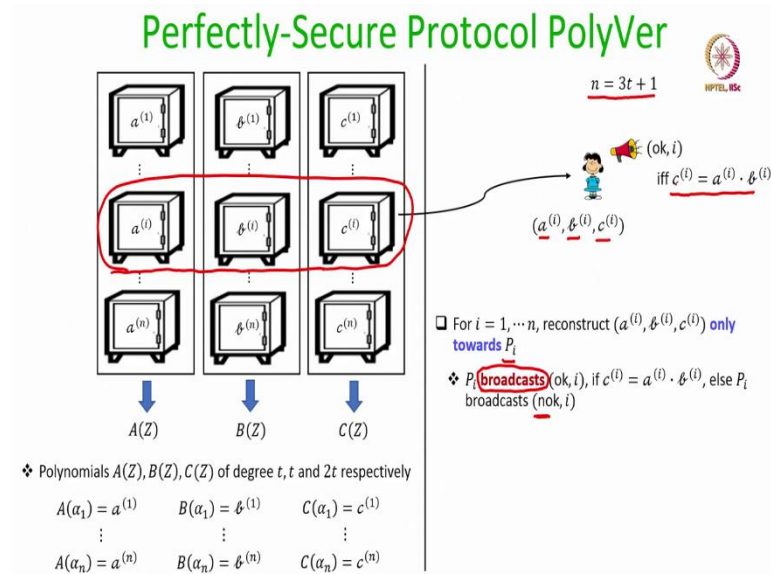
Since all these triples are secret shared, we can apply the Reed- Solomon error correction algorithm, we can ask each party to make public its shares of all the triples. And then we can apply the Reed-Solomon error correction as a result of that all the triples will be known to the parties.

And then they can check whether this property holds or not because once they know the exact value of all these triples, they can check whether $a_1 \cdot b_1 = c_1$ or not, $a_2 \cdot b_2 = c_2$ or not and then like, $a_n \cdot b_n = c_n$ or not. If all this equality holds and we can conclude that a C polynomial is indeed equal to the product of A and B polynomials. But we cannot do that because another property which we require from the polynomial verification protocol is the following.

We require that during this verification process at most t points on the A, B and C polynomials should be revealed to the adversary not more than t , this automatically implies that if the A, B and C polynomial were unknown to the adversary to begin with, then at the end of the protocol, adversary should learn at most t points on those polynomials. It cannot figure out what exactly was the entire A polynomial, B polynomial or C polynomial.

So, that is the requirement and because of this requirement whatever method I suggested earlier namely we make we make all these triples public and then check whether all the triples are multiplication triple that will not work because in that protocol adversary end up learning more than t points on the A, B and C polynomial because it learns all the n points on the A, B and C polynomial. So, that method will not work right. So, instead we have to use a different approach.

(Refer Slide Time: 05:36)



So, let us see the protocol here, the idea behind the protocol is as follows; what we will do is there are n number of secret share triples, we will designate each party to verify one of these triples. So, for instance the i -th party can be designated to verify whether the i -th input triple the i -th secret shared input triple is indeed a multiplication triple or not. For that what must we do?

We have to just let the i -th secret share triple to be reconstructed only towards the party P_i and, how we can do that? Well, we can do that by asking every party to send its share shares of these triples only to the party P_i instead of sending it to every other parties only to the party P_i and then party P_i can apply the Reed-Solomon error correction on the received shares of a_i , b_i and c_i .

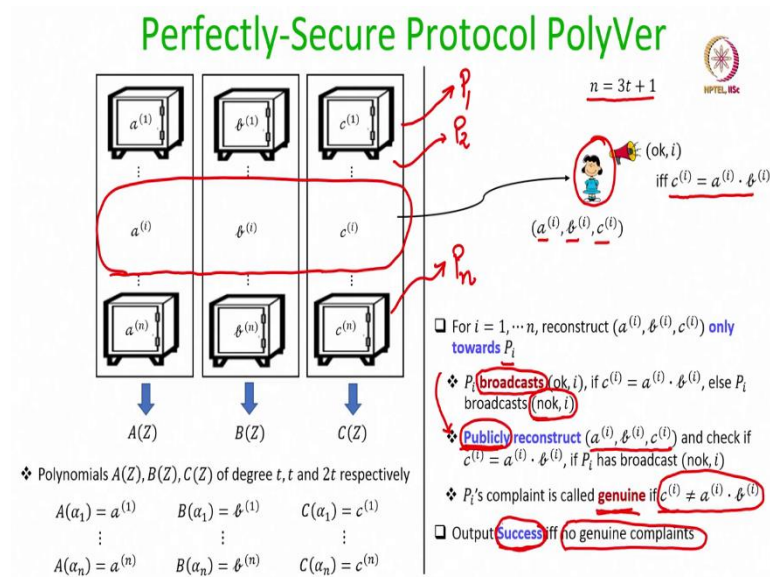
Since we are working in the setting where $n = 3t + 1$ and the degree of sharing of a_i , b_i and c_i is t , even if up to t shares for a_i , b_i and c_i are corrupt party P_i can error correct them and it can robustly reconstruct the a_i , b_i and c_i values. And then once it learns the a_i , b_i and c_i values it can verify whether this triplet (a_i, b_i, c_i) constitutes a multiplication triplet or not. If it constitutes a multiplication triplet, then P_i says that it has personally privately checked and indeed (a_i, b_i, c_i) is a multiplication triplet otherwise P_i makes public an NOK message.

So, this OK or NOK message is broadcasted - by broadcast I mean a reliable broadcast protocol is used here by P_i - to make public the OK or the NOK message. Notice that if P_i

is honest then only if the triplet is not a multiplication triplet it will broadcast an NOK message. However, if P_i is corrupt then even though the triplet (a_i, b_i, c_i) is a multiplication triplet it may unnecessarily broadcast an NOK message.

So, we do not know whether party P_i is honest or corrupt, if it is honest then indeed it will perform its steps properly that is it will broadcast OK only if the triplet is a multiplication triplet else it will broadcast an NOK message.

(Refer Slide Time: 08:38)



So, now what we have to do is if P_i broadcasts an NOK message then we do not know whether P_i is lying or whether P_i is behaving honestly; that means, whether we should consider the complaint of P_i genuine or not. If it has made a complaint by making public an NOK message then that is a complaint; that means, it is complaining against the i -th triplet and to verify whether its complaint is indeed genuine or not, what we do next is that we now publicly reconstruct the i -th triplet.

So, you see this public reconstruction of the i -th triplet is optional it is happening only when a complaint is made against the i -th triplet by P_i and only P_i can complain against i -th triplet. That means, the first triplet P_1 is designated to check, the second triplet P_2 is designated to check, the i -th triplet i -th party is designated to check and n th triplet the n th party is designated to check.

Now, the designated verifier here what they are doing is they are either making public an OK message or NOK message if any designated verifier has made public an NOK message for its triplet; that means, now it is the time to check it publicly. Because we cannot trust the designated verifier, the designated verifier could be corrupt, and it might unnecessarily complain against the i -th triplet even though it is a multiplication triplet.

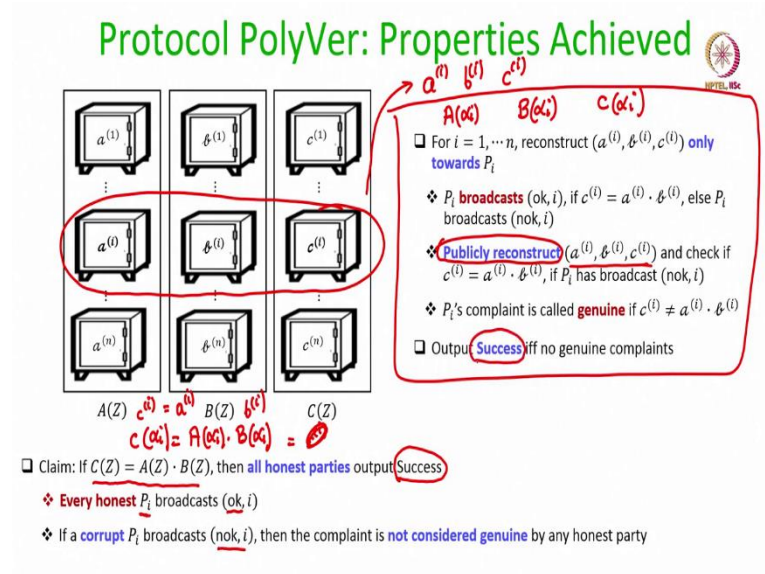
So, that is why now it is time to publicly check it and this step is optional; that means, if P_i has broadcasted an ok message for the i -th designated triplet then it is fine, no need to check it, we trust P_i , but if a complaint is made fine it is a time to publicly check it and to publicly check it the value (a_i, b_i, c_i) needs to be publicly reconstructed. So, that is why in this picture I am showing this (a_i, b_i, c_i) now in clear; that means, it will be learned by everyone.

How this (a_i, b_i, c_i) will be publicly reconstructed, well every party will be asked to send its shares of a_i , b_i , and c_i to every other party because it is now a public reconstruction and then every party will apply the Reed-Solomon error correction algorithm and they will reconstruct the value of a_i , b_i and c_i . Now, once this triplet is available publicly P_i 's complain will be called it will be termed genuine if indeed this triplet turns out to be it turns out not to be a multiplication triplet.

It turns out to be a non-multiplication triplet; that means, c_i is not the product of a_i and b_i then we will consider P_i 's complaint to be genuine otherwise we will label the P_i 's complaint as a fraud complaint. Now, what is the output of the protocol? The output of the protocol is success if and only if there are no genuine complaints against if the if and only if there are no genuine complaints.

That means, either every party has broadcasted an OK message or wherever there are NOK messages which are broadcasted those complaints turned out to be non-genuine. If this is the case then the output of the protocol is success; that means, the C polynomial is indeed the product of A and B polynomial otherwise the output is failure, that is the polynomial verification protocol right.

(Refer Slide Time: 12:30)



So, this is the these are the steps of the protocol. Now, we have to prove what are the properties achieved by the protocol, the crucial property we wanted is that if the C polynomial is not equal to the product of A and B polynomial then the output should not be success. So, we are going to prove that. So, we first prove that if indeed the C polynomial is product of A and B polynomials then all honest parties will output success, why honest parties because we do not care what the corrupt parties output.

That means, even though a corrupt party should output success because all the complaints are non-genuine it may still end up outputting failure, we cannot prevent corrupt parties from doing that. But what we want here is that if the C polynomial is indeed the product of A and B polynomials then all honest parties should output success, let us see whether that is happening here or not.

So, if the C polynomial is the product of A and B polynomials what exactly every honest party P_i is going to do. So, party P_i is designated with the task of verifying the i -th output triple. So, I am consider here I am considering here an arbitrary honest P_i that arbitrary honest P_i would have learnt the value of a_i , b_i and c_i and what exactly a_i , b_i and c_i , a_i is equal to the A polynomial at α_i , what is the b_i , b_i is the value of the B polynomial at α_i and what is c_i , c_i is the C polynomial at α_i .

What is the hypothesis of the claim? The hypothesis of the claim is that the C polynomial is the product of A and B polynomial, if C polynomial is the product of A and B polynomial. What is $C(\alpha_i)$? $C(\alpha_i)$ is the c component of the i -th triple. What is $A(\alpha_i)$?

It is the a component of the i -th triple and what is b_i ? It is the b component of the i -th triple. So, since party P_i is honest and it learns c_i , a_i , b_i and c_i is the product of a_i , b_i it will find that (a_i, b_i, c_i) is a multiplication triplet and as a result of that it will broadcast OK message. However, if P_i is corrupt it may broadcast an NOK message that is quite possible even though the triplet is a multiplication triplet.

But then if it broadcast and NOK message then the triplet will be publicly reconstructed and after publicly reconstructing it everyone will find that P_i 's complaint is not genuine because it has simply complained even though it that triplet (a_i, b_i, c_i) is a multiplication triplet. So, it will not be considered as a genuine complaint and as a result of that everyone will output success here.

(Refer Slide Time: 16:05)

Protocol PolyVer: Properties Achieved

$A(Z)$ $B(Z)$ $C(Z)$

$n = 3t + 1$

- For $i = 1, \dots, n$, reconstruct $(a^{(i)}, b^{(i)}, c^{(i)})$ **only towards P_i**
- ❖ P_i **broadcasts** (ok, i), if $c^{(i)} = a^{(i)} \cdot b^{(i)}$, else P_i broadcasts (nok, i)
- ❖ **Publicly reconstruct** $(a^{(i)}, b^{(i)}, c^{(i)})$ and check if $c^{(i)} = a^{(i)} \cdot b^{(i)}$, if P_i has broadcast (nok, i)
- ❖ P_i 's complaint is called **genuine** if $c^{(i)} \neq a^{(i)} \cdot b^{(i)}$
- Output **Success** iff no genuine complaints

□ Claim: If $C(Z) \neq A(Z) \cdot B(Z)$, then **no honest party** outputs Success

❖ $\mathcal{H} \triangleq$ set of **honest** parties

❖ For simplicity, let $\mathcal{H} = \{P_1, \dots, P_{2t+1}\}$

❖ Corresponding to some $P_i \in \mathcal{H}$:

$C(\alpha_i) \neq A(\alpha_i) \cdot B(\alpha_i)$

Handwritten notes:

$C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i)$
 $C(\alpha_2) = A(\alpha_2) \cdot B(\alpha_2)$
 \vdots
 $C(\alpha_{2t+1}) = A(\alpha_{2t+1}) \cdot B(\alpha_{2t+1})$

$\left. \begin{matrix} C(\alpha_i) = A(\alpha_i) \cdot B(\alpha_i) \\ C(\alpha_2) = A(\alpha_2) \cdot B(\alpha_2) \\ \vdots \\ C(\alpha_{2t+1}) = A(\alpha_{2t+1}) \cdot B(\alpha_{2t+1}) \end{matrix} \right\} \Rightarrow C(\alpha) = A(\alpha) \cdot B(\alpha)$

On the other hand, suppose the C polynomial is not the product of A and B polynomial then we claim that no honest party outputs success, again why an honest party? Because in this case corrupt parties may simply end up outputting success even though the C polynomial is not the product of A and B polynomials. So, for proving this claim let us focus on the set of honest parties I denote it by \mathcal{H} and for simplicity imagine that the first $2t + 1$ parties are honest.

Why $2t + 1$? Because we are considering a setting where $n = 3t + 1$ there could be up to t corrupt parties. So, we take the worst case when there are exactly t corrupt parties; that means, we are left with exactly $2t + 1$ honest parties they could be any $2t + 1$ honest parties, but for simplicity let me take those honest parties to be the first $2t + 1$ honest parties.

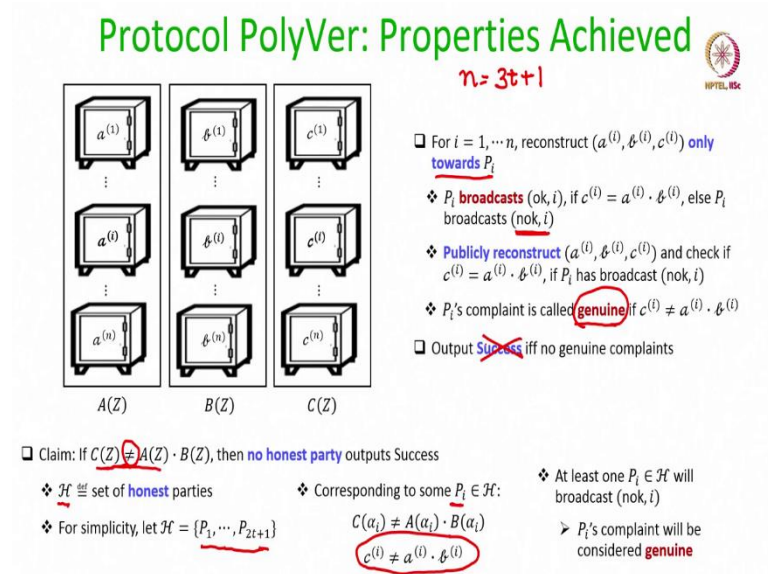
So, what is the hypothesis of the state claim here claim statement that the C polynomial is not the product of A and B polynomials. If that is the case, then the claim here is that there will be at least some honest party P_i corresponding to which the value of the C polynomial at α_i will not be equal to the value of the A polynomial at α_i times the value of the B polynomial at α_i .

This is because if this is not the case; that means, if there is no such honest P_i ; that means, the C polynomial at α_1 is equal to A polynomial at α_1 times B polynomial at α_1 and corresponding to P_2 the C polynomial at α_2 is equal to A polynomial at α_2 times B polynomial at α_2 . And like that say the C polynomial at α_{2t+1} is equal to the A polynomial at α_{2t+1} times the B polynomial at α_{2t+1} . Then together this implies that the C polynomial is equal to the product of the A and B polynomials which is a contradiction.

How we get C polynomial equal to A polynomial times B polynomial, because this equality holds for $2t + 1$ distinct points the degree of the C polynomial is $2t$ and the degree of the A and B polynomials is t and if this equality hold then the C polynomial will be equal to A polynomial times B polynomial which is a contradiction because as per the hypothesis of the claim statement the C polynomial is not equal to the product of A and B polynomials.

That means, there is definitely at least one honest party P_i in the system corresponding to which the value of the C polynomial at α_i is not equal to the value of the A polynomial at α_i times the value of the B polynomial at α_i that is guaranteed. It could be any P_i in the set \mathcal{H} it could be either P_1 or P_2 or P_3 or P_i or P_{2t+1} we do not know, but the existence of such a P_i is guaranteed.

(Refer Slide Time: 19:49)

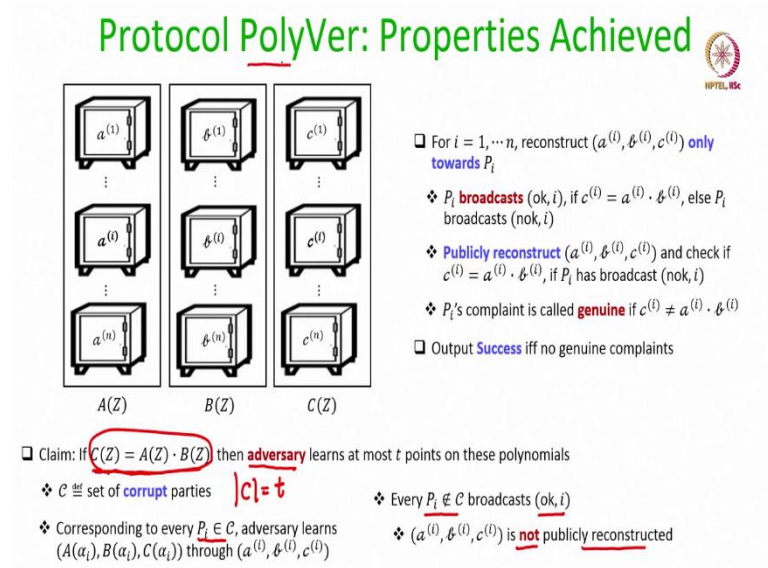


Now, if that is the case then it automatically implies that the i -th triplet here the i -th secret shared triplet is not a multiplication triplet because all these triplets constitute distinct points on the A, B and C polynomials. Now, what exactly that honest party P_i would have done in the protocol, when the party P_i would learn the i -th triplet when it is reconstructed towards the party P_i it will find that it is not a multiplication triplet and hence it will publicly complain it will broadcast an NOK message.

And as soon as it broadcast an NOK message it will be publicly reconstructed no one will prevent no adversary cannot prevent a reconstruction of this triplet robustly; that means, it cannot provide incorrect shares and hope that the triplet is reconstructed incorrectly. So, whatever is the value of the triplet it will be reconstructed correctly because everyone will be applying the Reed - Solomon error correction algorithm.

And as soon as this triplet is publicly learnt it will be known to everyone that it is not a multiplication triplet. Hence, P_i 's complaint will be considered as a genuine complaint and as a result of that no one will output success.

(Refer Slide Time: 21:22)



Now, let us prove the last property we claim here that if the C polynomial is the product of A and B polynomials then throughout the protocol adversary learn at most t points on these polynomials.

So, let \mathcal{C} be the set of corrupt parties in the system and again we take the worst case when there are exactly t corrupt parties in \mathcal{C} . So, what exactly the adversary learns here in the protocol every party is designated to first privately check one of the n secret shared triplets. So, since there are t corrupt parties; that means, there are t corresponding designated corrupt verifiers who will be verifying up to t number of secret shared triplets here.


That means, they will be learning the value of A polynomial, B polynomial and C polynomial at α_i corresponding to every corrupt P_i and there are at most t such. In fact, there are exactly two such corrupt parties because we are assuming that there are t corrupt parties in the system. So, that is one case. And now, corresponding to every honest party adversary does not learn anything about the corresponding input triplet.

So, if P_i is not a corrupt party; that means, if P_i is an honest party since the C polynomial is a product of A and B polynomial it will broadcast an OK message instead of NOK message and no one will be publicly reconstructing the i -th input triplet. That means, this i -th input triplet will not be publicly reconstructed, it will be reconstructed only by P_i and since P_i is assumed to be honest; that means, adversary does not learn what exactly is the i -th input triplet.

So, that means that if the C polynomial is the product of A and B polynomials adversary can learn at most t of this n number of secret shared triplets during the polynomial verification protocol. So, you can see now that we have achieved all the protocols and you can see that why this protocol is perfectly secure, this protocol is perfectly secure because all the properties are achieved against a computationally unbounded adversary and there is no error associated with any of the achieved properties.

(Refer Slide Time: 24:14)

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.