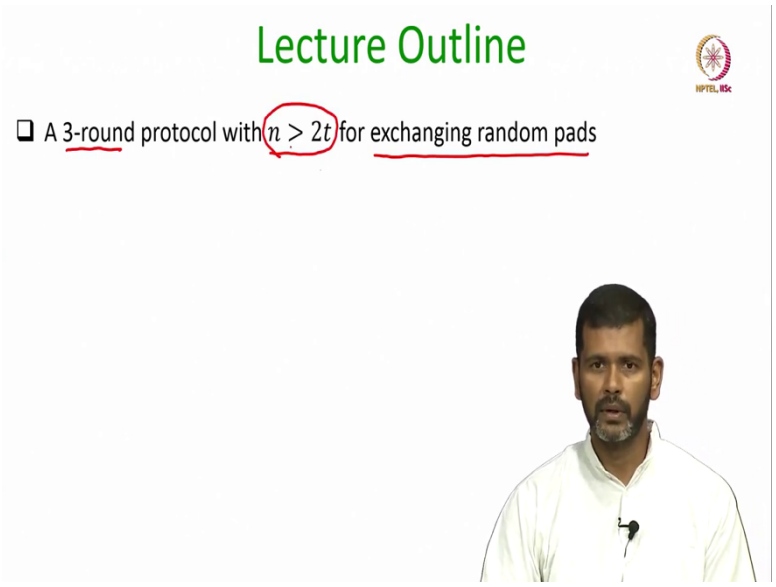
**Secure Computation: Part II**
**Prof. Ashish Choudhury**
**Department of Computer Science and Engineering**
**Indian Institute of Science, Bengaluru**

**Lecture - 24**
**Multi-Round PSMT Protocol-I**

Hello everyone. Welcome to this lecture. So, in the last lecture, we discussed a one-round PSMT protocol, we will now focus our attention on Multi-Round PSMT Protocols.

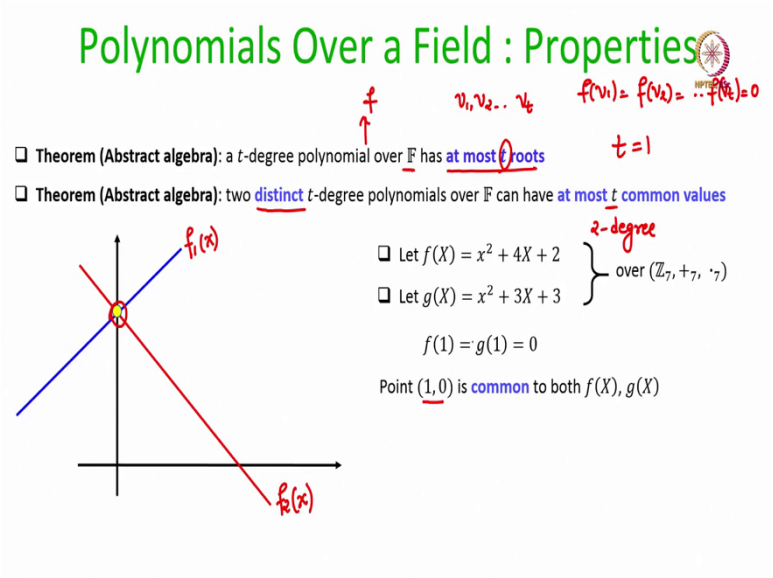
(Refer Slide Time: 00:40)




So, we will have two lectures on multi-round PSMT protocols. So, in this lecture, we will discuss a protocol which allows two parties to exchange random pads and this will be a 3-round protocol with the condition n > 2t.

In the next lecture, we will see how we can utilize this protocol to get a multi-round PSMT protocol.

(Refer Slide Time: 01:02)



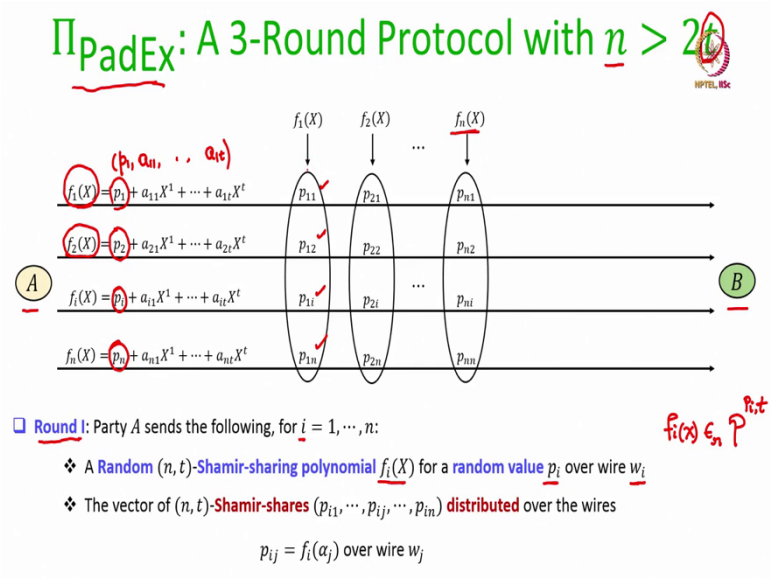

So, before going into the protocol for exchanging random pads, let us see some more properties of polynomials over a field. So, I will be just going through the properties. We will not be spending time on the proof for these properties. If you want to see the exact proofs for these properties you are referred to any standard text on abstract algebra or to my NPTEL course on discrete mathematics where I do have a module on abstract algebra.

So, the first property is that if you take any t-degree polynomial over a field then it can have at most t roots. That means, there could be at most t values, t elements, $v_1$, $v_2$, $v_t$, such that if my polynomial is f, then $f(v_1) = f(v_2) = \cdots = f(v_t) = 0$. There can be at most t such values. There cannot be more than t such values.

And based on this result we can derive another result, which states that if you take any two different t-degree polynomials then they can have at most t common values, ok. So, pictorially for instance if I take t is equal to 1, then a polynomial of degree 1 is nothing but a straight line.

So, if I take 2 straight lines which correspond to 2 different polynomials of degree 1, namely you have say a polynomial $f_1(X)$ and another straight line represented by the polynomial $f_2(X)$, they will have only one common point lying on both of them. And this was for t is equal to 1, but in general, we can prove this result for any t, ok.

(Refer Slide Time: 04:06)




Now, based on these properties let us see the description of the pad exchange protocol. The goal here is that we have two entities, two parties A and B, ok and they are connected by n bidirectional wires, where n > 2t. And among those n bidirectional wires at most t wires could get byzantine corrupt. Which t by wires will be byzantine corrupt it will not be known.

But the value t is publicly known, the parameter t is publicly known. And looking ahead in the context of our PSMT protocol, S will be playing the role of node A and receiver or R will be playing the role of node B, ok. But this is an abstract protocol where any of the two parties S and R could have played the role of A and B, respectively.

The goal is basically to allow the parties A and B to interact for 3-rounds such that at the end of the interaction both A and B have a common pad, common random value from a field which is known only to both A and B, which will be then later used in our PSMT protocol.

So, to achieve this goal during the first round party A computes and sends the following over the different wires to the party B or the node B. So, since this is the 3-round protocol there will be round 1, round 2 and round 3. So, this is what happens during round 1.

So, over the ith wire $w_i$ party A does the following. It picks a random value $p_i$ from the field and corresponding to that it picks a random t-degree Shamir-Sharing polynomial $f_i(X)$. So, $f_i(X)$ is a random polynomial from the set of all polynomials whose constant term is $p_i$ and

whose degree is t, ok. So, it sends one polynomial over each wire. So, remember this is for $i = 1, \cdots, n$, there are n such wires.

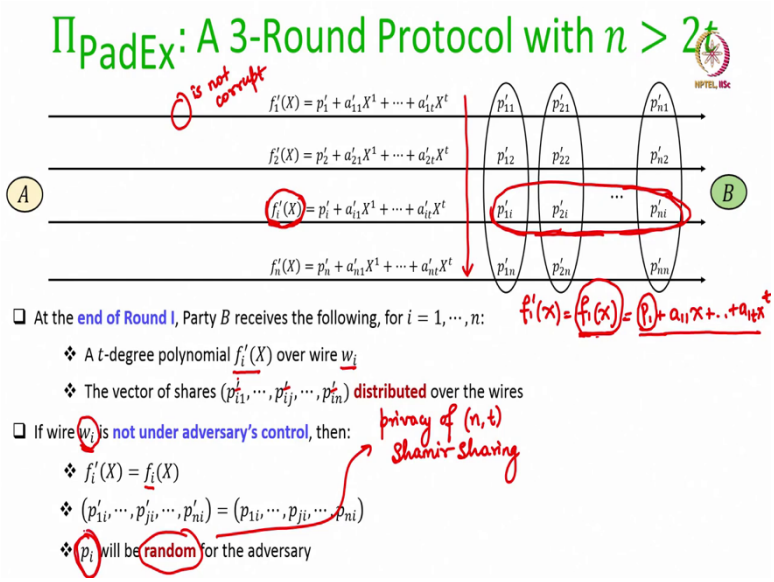
So, over the first wire, A sends the first polynomial. And when I say it sends the polynomial, it is equivalent to saying that it sends the coefficients of those polynomials, ok. Over the second wire, A sends the second random polynomial, and you see all these values $p_1, p_2, p_n$, they are randomly picked by node A. Over the ith wire, it sends the ith polynomial, and over the nth wire, it sends the nth polynomial.

Apart from these polynomials, it also computes a vector of Shamir-shares for the value $p_i$ and that vector of Shamir-shares is distributed over the n wires. Specifically, the polynomial $f_1(X)$ is evaluated at $\alpha_1, \alpha_2, \cdots, \alpha_n$ and those resultant values are sent over the wires $w_1, \cdots, w_n$ respectively.

So, you see the first polynomial is sent over the first wire, but its n values are distributed and sent across n wires, 1 value over each wire. In the same way, the second polynomial is evaluated at alpha 1, alpha 2, alpha n, and the resultant values are sent across the n channels to node B. And like that the nth polynomial is evaluated at alpha 1, alpha 2, alpha i, alpha n and sent to node B. That is round 1.

So, to summarize in round 1, n polynomials are communicated and the value of those n polynomials at n distinct points are also communicated, ok.

(Refer Slide Time: 08:53)




Now, at the end of Round 1, party B will receive the following. It will receive some t-degree polynomial over the wire w i, ok.

So, I am using a notation with prime to denote the polynomials which are received by the node B because if the wire w i is corrupt then the received polynomial could be different. Whereas, if the wire w i is not corrupt then the received polynomial will be the same as sent by the sender. But node B will not be knowing whether the received value is the one which sender sent or whether it is a changed polynomial.

So, that is why to differentiate it from the polynomial which A would have communicated, I am bringing this prime notation here, so ok. So, over each wire B would receive a t-degree polynomial and it would receive a vector of shares across the n wires, n such vectors of shares. And again I use this prime notations to denote the vector of shares because if the wire w i is corrupt, then it can deliver incorrect information.

So, what we can say here is the following. If the ith wire is not under the adversary's control, then the delivered polynomial is the same as sent by the node A or the party A. That means, if the ith channel is not under adversary's control, then this polynomial $f_i'(X)$ will be same as the senders polynomial $f_i(X)$, and the ith component of all the vectors of shares will be the same as sent by the sender.

But if the ith wire is not, but if the ith wire is under adversary's control, then there is no guarantee whether the delivered polynomial is the same as sent by the node A. And at the same time it will not be guaranteed that the ith component of all the vectors of shares as delivered to the node B is the same as sent by the node A.

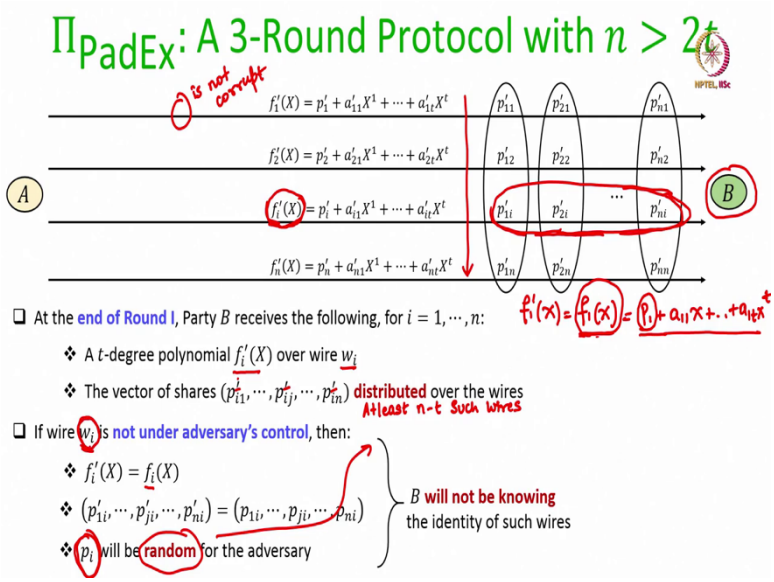
Also, if the wire wi is not under the adversary's control, then the constant term of the ith polynomial will be random for the adversary. This comes from the privacy property of n, t Shamir-Sharing. Say for instance, if the w1 wire is not corrupt, in that case $f_1^{'}(X)$ is same as $f_1(X)$ and its form will be $p_1 + a_1X + \cdots + a_tX^t$.

So, since the first wire is not under the adversary's control, the adversary will not be learning this polynomial, but corresponding to this polynomial adversary would have learnt the value of this polynomial at t points, because those values are distributed across n wires among those n wires t could be under adversary's control. So, even though the adversary has not seen the full polynomial $f_1(X)$, it will see the t values of these polynomials. But this polynomial $f_1(X)$ is a random polynomial, it is chosen randomly.

So, now we can invoke the privacy property of n t Shamir secret sharing which guarantees that if the underlying Shamir sharing polynomial is chosen randomly and if its degree is t, then the constant term of the polynomial it remains random from the view point of the adversary, ok. Namely, adversary will not be able to figure out what exactly is the constant term of the underlying sharing polynomial, and that will ensure that the value p 1 is random for the adversary.

So, the claim here is that whichever wires are not under the adversary's control, the constant term of those polynomials will be random from the viewpoint of the adversary.

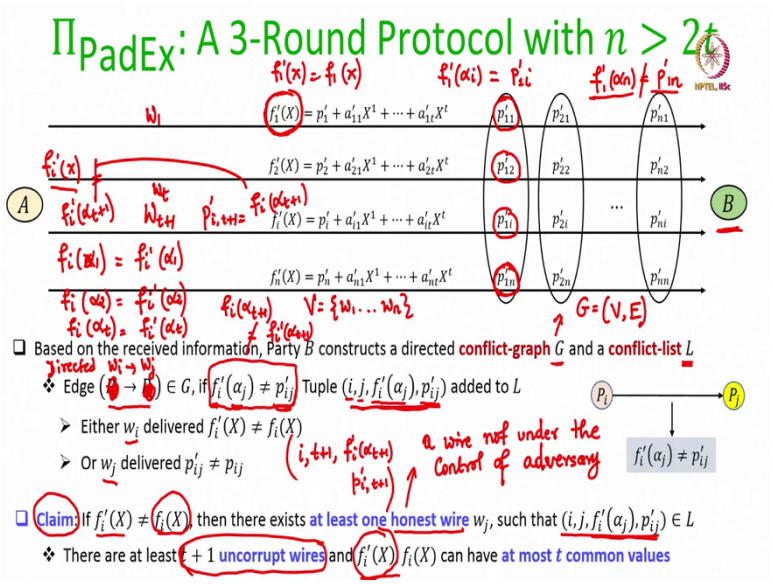
(Refer Slide Time: 13:59)




And how many such honest wires or un-corrupt wires wi will be there? At least n - t such wires because there are at least n - t wires, which can be free from adversary's control.

However, the party B or the node B will not be knowing the exact identity of such clean wires w i because it only knows the number of corrupt wires. However, it may not know the exact identity of the corrupt wires, right. So, that finishes round 1.

(Refer Slide Time: 14:36)



Now, based on whatever information node B has received, party B has received at the end of round 1 it prepares a conflict-graph G and corresponding to that conflict-graph G it computes a conflict list L. So, let us see what exactly is the conflict-graph.

It will be a graph where the vertex set is the set of wires, ok. So, there are n vertices in the conflict-graph.

And there will be a directed edge, I stress a directed edge from the node wi to the node w j if the value of the ith polynomial when evaluated at alpha j does not match the value $p'_{ij}$. So, what we are trying to do here? Ideally, say for instance if I take the polynomial $f'_1(X)$, then if $f'_1(X)$, is delivered without any error; that means, adversary has not changed the polynomial $f'_1(X)$,. That means, $f'_1(X) = f_1(X)$, suppose if this is the condition then this polynomial when evaluated at alpha 1 should match the value $p'_{11}$. Similarly, $f'_1(X)$ polynomial evaluated at alpha 2 should match the value $p'_{12}$. And like that, $f'_1(X)$ polynomial evaluated at alpha n should match the value $p'_{1n}$.

Suppose, it turns out that say $f'_1(X)$ polynomial evaluated at alpha n is not equal to $p'_{1n}$, then it either means that this first polynomial has been delivered incorrectly or its nth value has been delivered incorrectly to the node B. What exactly is the case; node B will not be knowing because it does not know what exactly are the corrupt wires, right. What it can conclude only is that the wire one is corrupt or wire w n is corrupt.

So, basically through the conflict-graph the party B is trying to capture all such mutual conflicts between pair of wires. So, whenever ith received polynomial, does not match the supposedly jth point on that polynomial, node B goes and add a directed edge as a conflict from the wire w i to the wire w j. And corresponding to that conflict, it adds a tuple, a conflicting tuple to a conflict list.

And that conflicting tuple represents the indices of the conflicting wires and the two pieces of conflicting values. Namely, the ith received polynomial evaluated at alpha j and the supposed

value $p'_{ij}$ which should ideally lie on that which should match that value, but they are not matching, ok.

And whenever such a tuple conflicting tuple is added to the conflict list L, then as I said earlier B can conclude that either wi has delivered an incorrect polynomial or the wire w j has delivered an incorrect point. But what exactly is the case it will not be able to find out, ok.

Now, before we proceed, an important claim here is that if at all the ith wire has delivered an incorrect polynomial; that means, the ith wire was corrupt, adversary has changed at least one of the coefficients of the polynomial, so that the received ith polynomial is different from the sent ith polynomial. Then, there will be at least one honest wire.

And what do we mean by honest wire? A wire not under the control of the adversary. So there will be, at least one such wire w j, such that there will be a conflicting tuple present in the conflict list where the wire w j is in conflict with wire w i. Namely, this tuple will be present in the list L. And this comes from the fact that you have at least t + 1 un-corrupt or honest wires, these wires will deliver the value of the polynomial $f_i(X)$ correctly, they will not be delivering the value of $f'_i(X)$, ok. And two different t-degree polynomials can have at most t common values.

So, among those t + 1 un-corrupt wires, the values which are delivered over those wires they may lie on $f_i(X)$ as well as $f'_i(X)$. So, they may not get in conflict with the wire wi. But there will be at least one honest wire among these t + 1 wires whose delivered value will be in conflict with the corresponding point on the received polynomial $f'_i(X)$.

So, for instance, if I take the first t+1 wires as the un-corrupt wires. Then, say the ith polynomial is changed, then it could so happen that the value of the actual ith polynomial at $\alpha_1$ is the same as the value of the modified ith polynomial at $\alpha_1$.

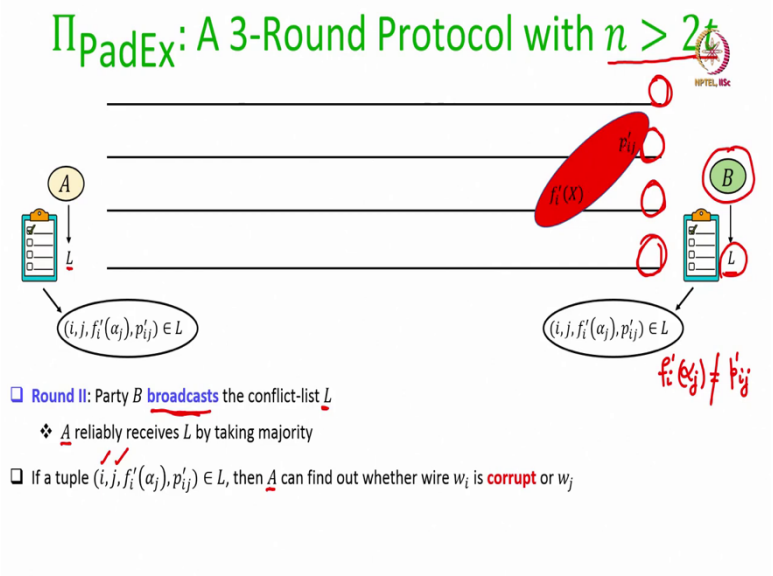
In that case, w1 will not be in conflict with wi. And it may so happen that the actual ith polynomial at $\alpha_2$ gives you the same value as the changed ith polynomial at $\alpha_2$, fine. And like that it could so happen that the actual ith polynomial evaluated at $\alpha_t$ gives you the same value

as the changed ith polynomial at $\alpha_t$, fine, in that case wire wt also will not be in conflict with wire wi.

But it cannot be the case that the value of the actual ith polynomial at $\alpha_{t+1}$ is the same as the value of the changed ith polynomial at $\alpha_{t+1}$. No, it will not be the case because we know that two different t-degree polynomials can have at most t common values.

That means, the value $p'_{i,t+1}$, which is actually the value of the original ith polynomial at $\alpha_{t+1}$, because I am assuming that the $(t+1)$th wire is an un-corrupt wire; this value will conflict with the value of the received ith polynomial at $\alpha_{t+1}$. And that is why there will be a tuple of the form $(i, t+1, f'_i(\alpha_{t+1}), p'_{i,t+1})$. That is a very important claim and that constitutes the heart of this 3-round pad exchange protocol.

(Refer Slide Time: 24:33)




Now, once B has prepared the conflict list it broadcasts it. And when I say broadcasts it, I mean to say that it sends the copy of L identically over all the n wires. So, do not confuse this term broadcast with the reliable broadcast protocol, which we had discussed earlier, ok. Here the broadcast means sending the same value, sending the same content over all the n wires to the other party.

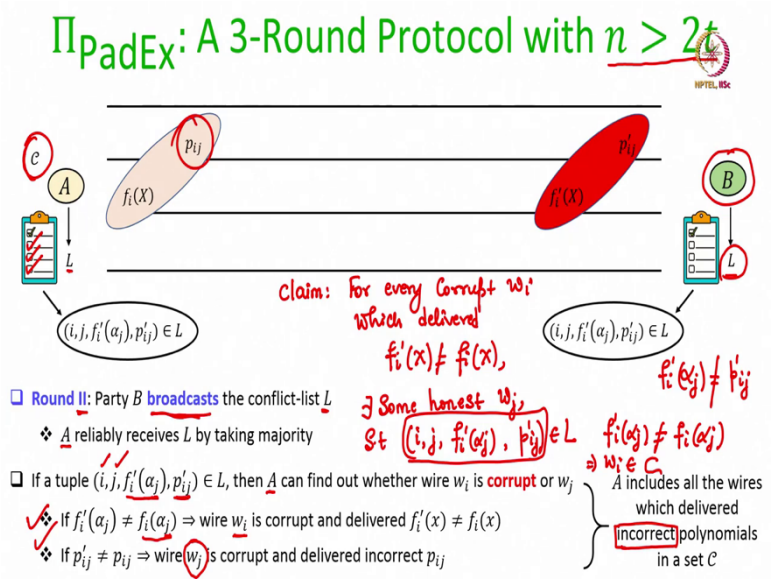
Now, since it is broadcasted and n is greater than 2t and at most t wires may get corrupt, those t corrupt wires may deliver incorrect copies of L to A. But still majority of the wires will be honest, they will be un-corrupt, and they will be sending, they will be delivering the actual conflict list L to the party A. So, party A at the end of round 2 will be able to get back the Bs version of the conflict list by taking the majority vote. That means, at the end of round 2, both A and B have agreed upon the conflict list.

And now for each conflict, each conflicting tuple which is present in this conflict list L, A can find out whether the wire wi is corrupt or whether that wire wj is corrupt, ok. So, from the view point of party B, there was a conflict between the ith polynomial and its value which has been delivered over the jth wire, that is why that conflict was present.

That means, the value of the ith received polynomial at $\alpha_j$ was not matching with the value $p'_{ij}$ delivered over the jth wire, that is why that conflict was present. But B was unable to figure out whether it is the polynomial, which has been delivered incorrectly or whether it is the value which has been delivered incorrectly, right.

But, now A can find out what exactly is the case. A would have received the same conflicting tuple, and now A can go and do the following.

(Refer Slide Time: 27:02)



It can take the first conflicting value in this tuple, re-compute the value of the ith polynomial that it has sent earlier at alpha j and if the two values mismatch then it can infer that it is the wire w i which has delivered an incorrect polynomial. And in the same way, it can go and take the second conflicting value in that tuple and compare it with the value that he had sent during round 1, and if a mismatch occurs, then it can conclude that it is the jth wire which has delivered an incorrect value during round 1.

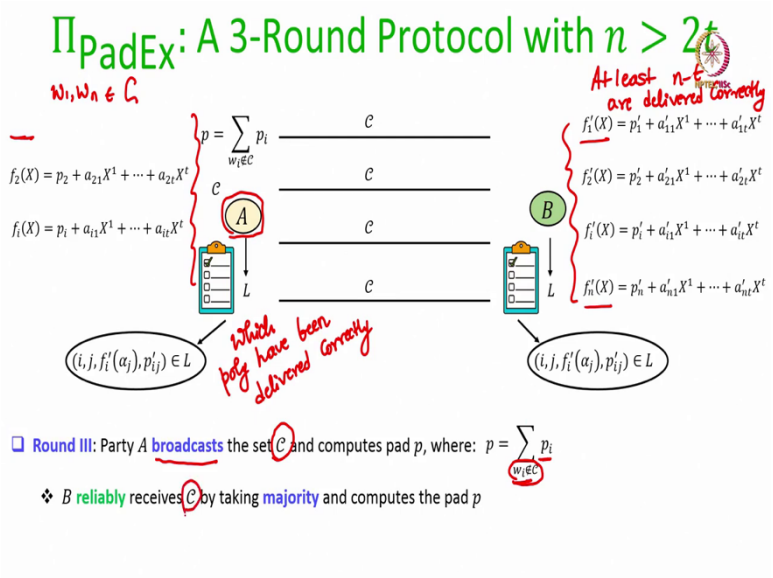
So, node A, party A can will carry out this computation for each of the list, sorry for each of the conflicting tuple present in the conflict list and from that it can find out, which of the two wires in each of the conflicting tuple is the corrupt wire, ok. Now, after A has done this task for all the conflicting tuple, it will have the set of all the wires $C$, which would have delivered incorrect polynomials during round 1 to the party B.

Why so? Because remember we just claimed earlier that for every corrupt wire wi which delivered an incorrect polynomial there will be some honest wire wj such that a conflicting tuple of the form $(i,\ j, f_i^{'}(\alpha_j), p_{i,j}^{'})$. will be present in the list L.

Now, when A would have received the list L, and when A would have performed this check for this conflicting tuple, it will find that it is $f_i^{'}(X)$ polynomial which has been delivered incorrectly. And hence, it will conclude that wire wi should be added to the set $C$, ok.

So, that means, at the end of round 2, once B communicates back its list of conflicts, the party A will be able to identify, which wires have delivered the correct polynomials and which wires have delivered incorrect polynomials. However, B does not know that information at the moment, it is available only with the party A. Only party A has found that set of wires, which have delivered incorrect polynomials, ok.

(Refer Slide Time: 30:29)




Now, remember party A also has the full set of polynomials which it has selected and communicated to the party B during round 1. So, the scenario right now is the following.

Party A has a set of n polynomials, party B has a set of n polynomials. Among those n polynomials at least n minus t are delivered correctly to B, but B does not know which polynomials have been delivered correctly which are delivered incorrectly. But A knows at the end of round 2, which polynomials have been delivered correctly. A has now that information.

So, what A does is during round 3, party A provides the information regarding the wires which have delivered incorrect polynomials by broadcasting the set $C$, ok. And then it simply ignores the polynomials which have been delivered over those wires to B because A does not know what version of the corresponding polynomials B have.

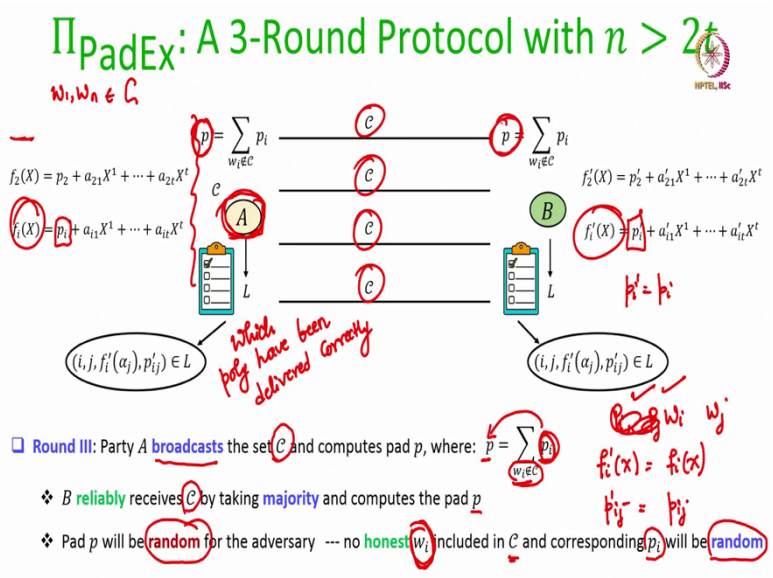
Say for instance, if wire 1, so I am taking the case when say w1 and w n are present in the set $C$. That means, the first polynomial has been delivered incorrectly and the nth polynomial is delivered incorrectly. Party A has absolutely no way of finding out what was the version of the first polynomial that B has received and what is the version of the nth polynomial that B has received. So, there is no point in working with them, so party A ignores them.

And now based on the remaining polynomials, which party A knows have been delivered correctly to the party B, a pad is computed by the party A and that pad is basically the the

summation of the constant term of all the polynomials which were delivered over the wires which are not in the set $C$, ok.

So, during round 3, the identity of the corrupt wires which have changed the polynomials during round 1 is communicated. And since it is broadcasted even if the adversary changes the contents of the set $C$ over t wires, majority of the wires will deliver the right set $C$.

(Refer Slide Time: 33:47)




So, B can recover the set $C$ by taking the majority. And once it receives the set $C$, it can also ignore the same polynomials, which have been ignored by party A. And the remaining polynomials are now in consideration. Their constant terms are taken, summed up and that is the pad p.

And now there is a common pad p which is known both to party A and party B. Why it is common? Because both of them will have the same set $C$. And all the polynomials in set $C$ are ignored by both A as well as B, and the polynomials which are delivered over the wires, not in set $C$, are the same at A's end as well as B's end. So, their constant term also will be the same and hence their summation will also be the same.

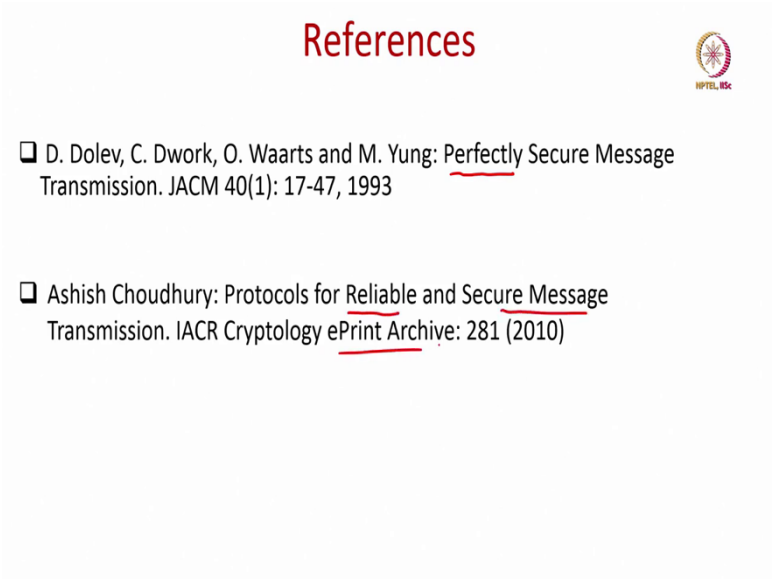
Now, why it will be random for the adversary? Because no honest wire would have been included in this set $C$, because no honest wire will be in conflict with any other honest wire. That means, all the honest wires are actually outside the set fancy C and that is why they are corresponding p i's are contributing to this final pad p. But, what can we say about the p i's

corresponding to the honest wires w i, which have been delivered over the honest wire w i? So, remember we have argued that they are random from the view point of the adversary.

Namely, if the ith wire was honest then we have already argued that adversary does not have any information about p i and there are multiple such p is which are summed up and giving the final sum p. So, at since at least one such unknown p i is there in the sum p, the value p will be random for the adversary, ok.

So, this is the 3-round pad exchange protocol which allows the two parties A and B to interact over the channels publicly. So, that even if up to t channels are under the adversary's control and where the adversary is computationally unbounded, it will not be able to stop A and B from agreeing on a random pad, ok which will be unknown for the adversary.

(Refer Slide Time: 37:06)




So, the 3-round protocol that I have discussed is primarily taken from this paper. And if you want to know more about PRMT and PSMT protocols you can refer to this thesis.

Thank you.