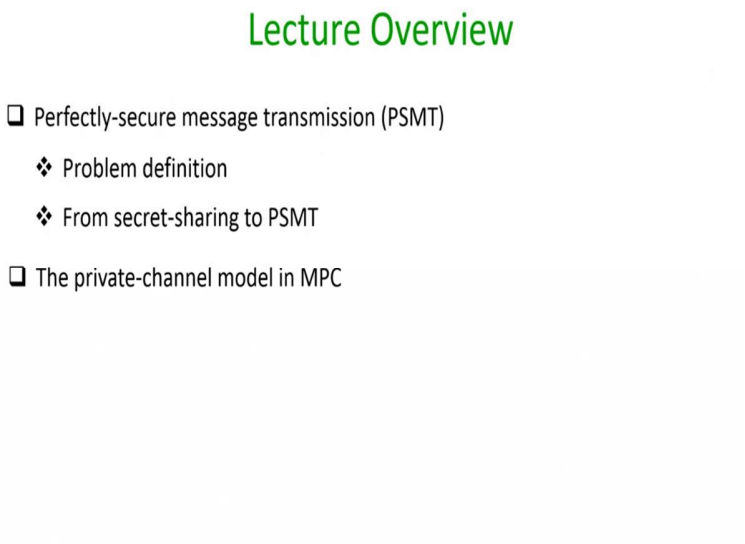
Secure Computation - Part I
Prof. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology, Bangalore

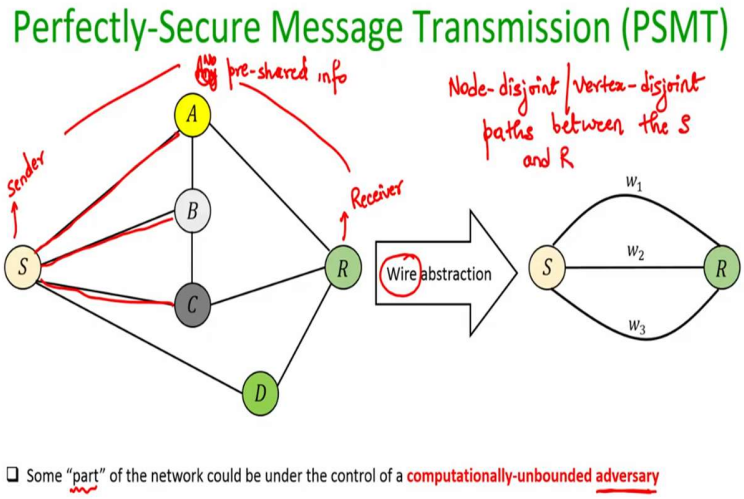
Module - 3
Lecture - 15
Perfectly-Secure Message Transmission

(Refer Slide Time: 00:33)



Hello everyone. Welcome to this lecture. So, the plan for this lecture is as follows: In this lecture, we will see the problem of perfectly-secure message transmission, which is also called as PSMT. We will see the problem definition and we will see that how we can solve this perfectly-secure message transmission problem using the help of secret-sharing. And then, we will conclude with what we call as the private-channel model in the multi-party computation.

(Refer Slide Time: 01:00)



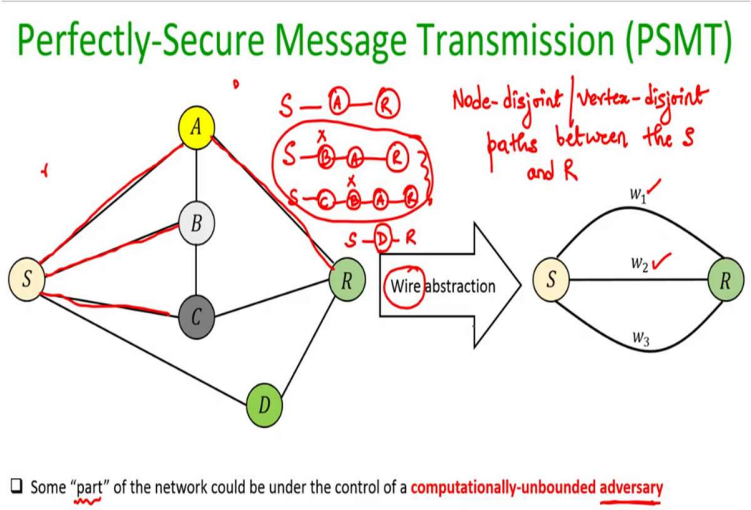
So, let us start with the problem definition of perfectly-secure message transmission or PSMT in short. So, the setting is as follows: You have 2 nodes or parties in the network, a sender S and a receiver R, and they do not have any pre-shared information, and they are connected in a network via intermediate nodes. So, you have several paths between the sender and receiver.

So, you have 1 path through the node A; you have another path through the node B and then followed by node A; you have a path through the node C; and then you have a path through the node D. And some part of the network could be under the control of a computationally unbounded adversary. So, I have written here, part, in quote-unquote. Of course, sender and receiver, they are not allowed to be under the control of the adversary; but some part of the network, that means, some of the intermediate nodes between S and R, could be under the control of an adversary who is computationally unbounded.

And they are controlled by the adversary, but they will not be deviating from any protocol instructions. That means, if some instructions are given to the intermediate nodes, even if they are under the control of the adversary, they will follow the protocol instructions; but whatever communication is happening through those intermediate nodes, that communication will be forwarded to the adversary.

So, what we do here is, we will abstract the underlying network by what we call as wire abstraction. And by wire abstraction I mean, I consider node-disjoint paths also called as vertex-disjoint paths between the sender and receiver. So, what does the vertex-disjoint or node-disjoint paths mean? Let us try to understand that in the context of this example itself.

(Refer Slide Time: 04:00)



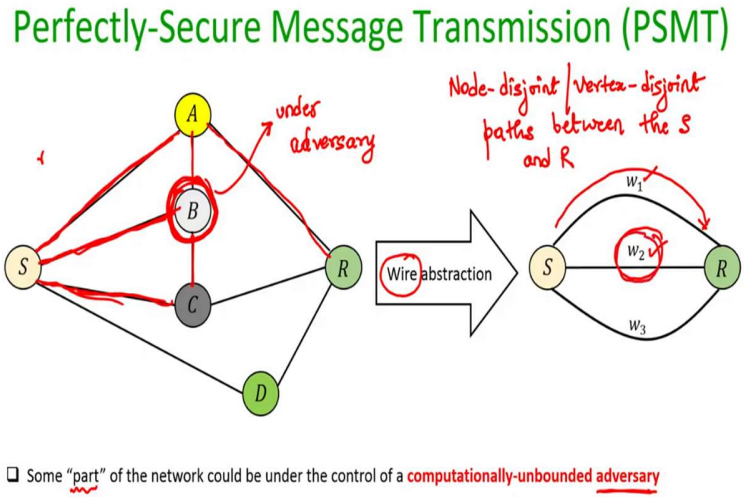
So, you can consider this path S and then A and then R as 1 path. And then, even though now you have 2 other paths, 1 path going from S to B and B to A and then to R, and another path going from S to C and C to B and then B to A and then A to R; they are not node-disjoint. So, if I consider the path S to B and then B to A and then from A to R, and another path consisting of S to C and then from C to B and then from B to A and then to R; these 2 paths, they are not node-disjoint, because there is a common node, namely the node B which is present in both these paths.

So, together these 2 paths will be abstracted as a single wire w_2, between S and R. And by wire, I mean channel; do not consider it as a physical wire, that is just a name. Similarly, the path S, A and R is abstracted as channel or wire w_1. And now you have a third node-disjoint path namely S to D and then D to R. So, there is no intermediate node here in this path, which is common in any of the remaining paths which we have already considered.

So, overall, I can abstract this network, the intermediate nodes present between S and R as wires between or channels between S and R, the disjoint channels. And the reason for abstracting this

communication as wires is the following: If any communication is happening through the node A, from S to R, that can be abstracted as if that communication is happening from S to R.

(Refer Slide Time: 05:59)

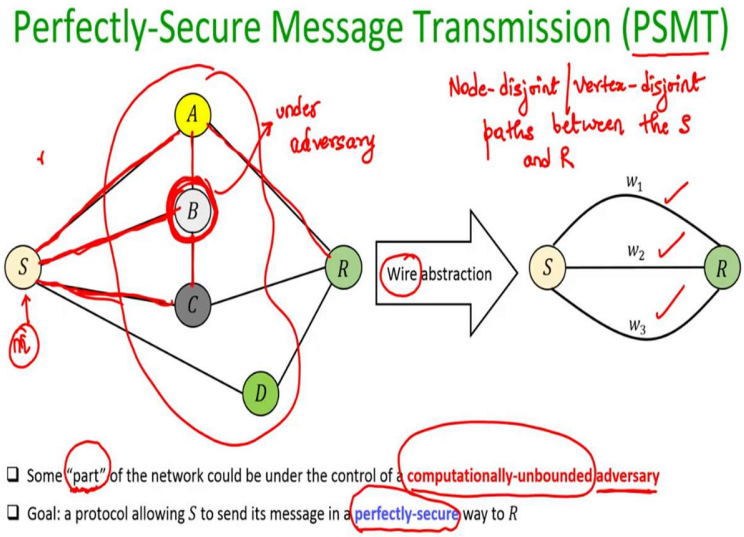


On the other hand, if some communication is happening from S to R, which involves the intermediate node B, and if the node B is under the control of the adversary; that means, even if the communication would have happened from S to C and then C to B and then B to A, that communication also will be under the control of the adversary; because, the node B is a common node along the path S to B, B to A, A to R, as well as along the path S to C, C to B, B to A and A to R.

That means, if this single node B gets corrupt; corrupt in the sense, if it gets compromised and get controlled by the adversary, then the adversary can actually see the communication happening through 2 of the paths between S to R. Namely, the communication happening over S to B, B to A, A to R, as well as S to C, C to B, B to A, A to R; so, that is abstracted as if there is a single wire between S and R, w 2.

And if it gets controlled by the adversary, then whatever communication is happening over this channel or the wire, will be learnt by the adversary. So, that is a way we do the wire abstraction.

(Refer Slide Time: 07:26)



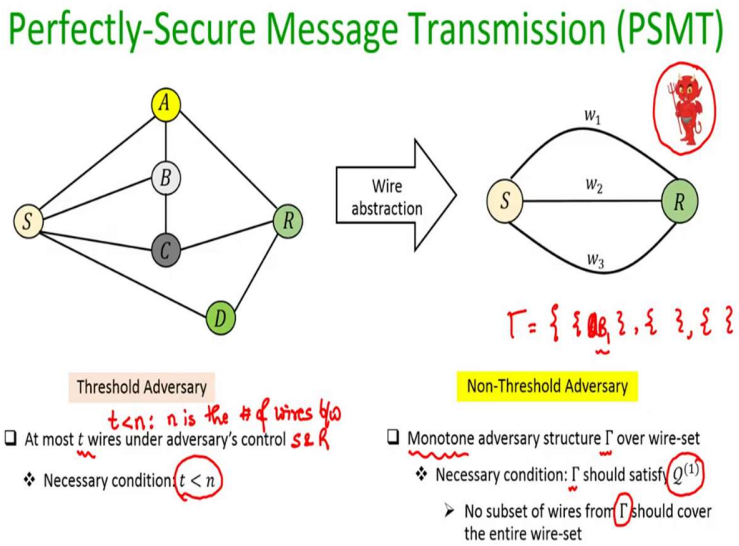
Now, what is the goal in the perfectly-secure message transmission problem? The goal is the following: Sender will have some message m, which is known only to the sender. It could be a single bit, it could be a bit string of certain length, whatever. The goal is to derive or devise a mechanism which allows the sender to communicate this message m via these intermediate nodes to the receiver R, in such a way that even if some part, some of these intermediate nodes or some of these wires get controlled by a computationally unbounded adversary, the message should remain private.

Of course, since we are assuming here that the adversary cannot cause the controlled node to deviate from the protocol instructions, the message will be delivered correctly, that is not an issue; but the goal here is to ensure the privacy of the message. And why we are calling it perfectly-secure message transmission, because here we are trying to demand security against an adversary who is computationally unbounded.

I stress here that the assumption here is that only a part of the network will be under the control of the adversary, excluding S and R. That means, it cannot be the case that all the intermediate nodes get controlled by the adversary. That is not possible. Because, if that would have been the case, then that is equivalent to saying that all the wires, w_1, w_2, w_3, all of them are under the control of the adversary, then how can it be possible for the sender to communicate anything privately to the

receiver. That means, there should be at least 1 wire or some of the wires which are not under the control of the adversary.

(Refer Slide Time: 09:15)



So, now, let us see that how exactly we model, that adversary controls a subset of the wires. So, we assume that we have a computationally unbounded adversary who can sit over some of the wires. And the adversary can be characterized as either a threshold adversary or a non-threshold adversary. So, in the threshold adversary model, we assume that if there are n wires between S and R, then at most t of those wires can be under the adversary's control, where t is strictly less than n and n is the number of wires between S and R.

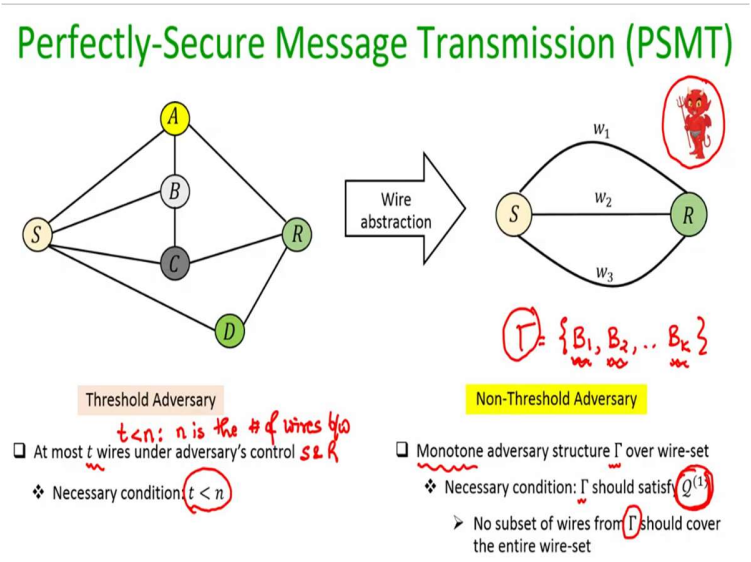
Whereas in a non-threshold adversary, we do not model the adversary by a threshold value t, but rather we will be given an adversary structure which will consist of several potential subsets of wires and any one of those subsets can be under the control of the adversary when the protocol gets executed. And this adversary structure will be a monotone adversary structure.

That means, Γ will consist of all the maximal subsets of wires which can potentially get corrupt by the adversary. So, if there is a subset say B_1, then, by monotone I mean that any proper subset of wires of B_1 will also be considered as a potentially corrupt set of wires which can get controlled by the adversary; same as we had done for the case of secret-sharing.

So, it is easy to see that if our adversary is modelled as a threshold adversary, then of course, a necessary condition to solve the perfectly-secure message transmission problem is that, there should be at least 1 wire which is not under the control of the adversary, namely, t should be strictly less than n. Because, if all the n wires are under the control of the adversary, there is no mechanism, no way by which S can communicate its message in a perfectly private way to the receiver.

Whereas, the necessary condition for solving the PSMT problem against the non-threshold adversary is that, my adversary structure should satisfy what we call as $Q^{(1)}$ condition. So, what does this $Q^{(1)}$ condition mean? This $Q^{(1)}$ condition mean that, if you take any subset of potentially corrupt wires from your adversary structure, that should not be the entire set of n wires between S and R.

(Refer Slide Time: 12:21)



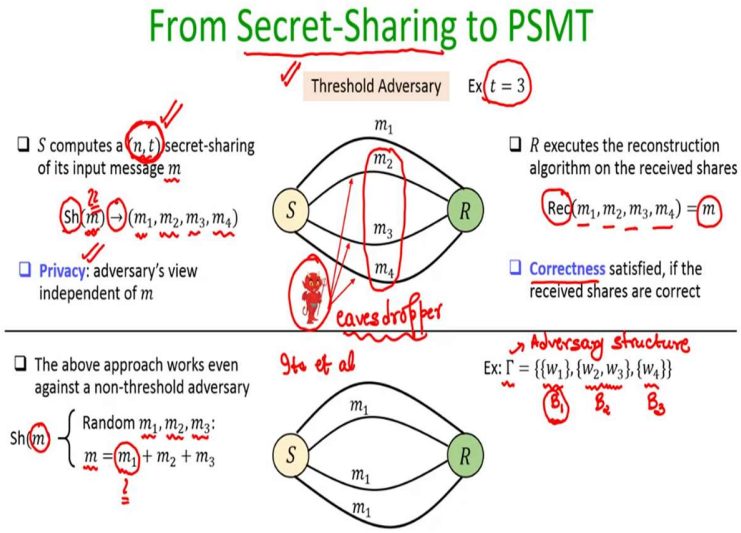
That means, if I consider that, say my Gamma consists of subsets B_1, B_2, and like that there are say B_k potential unauthorised subset of wires over an adversary can sit, either the wires in B_1 or the wires in B_2 or the wires in B_k, then, when I say that my adversary structure Gamma satisfies the $Q^{(1)}$ condition, by that I mean that B_1 is not the entire set of wires, or B_2 is not the entire set of wires, or B_k is not the entire set of wires, and so on.

And it is easy to see that this $Q^{(1)}$ condition is a strict generalisation of the t less than n condition. So, t less than n condition means that, you take any subset of t wires, that does not cover the entire

set of n wires, because t is strictly less than n. That condition is strictly generalised to the $Q^{(1)}$ condition, because now the cardinality of this potentially corrupt subsets of wires could be different.

So, B_1's cardinality could be different, B_2's cardinality could be different, and so on. Because, in the non-ratio model, there is no restriction that the size of each of this maximal potential corrupt subset of wires should be same.

(Refer Slide Time: 13:46)



So, we have seen the problem definition of perfectly-secure message transmission. Now, let us see that how we can solve this problem. And we will see here that, if you are given a secret-sharing scheme, say in the threshold model, then you can get a PSMT protocol in the threshold model; if you are given a secret-sharing protocol in against the non-threshold adversary, then you can design a PSMT protocol against the non-threshold adversary.

So, let us see first for the case of threshold adversary. And for the sake of demonstration, I am assuming that t = 3. That means, there should be 4 or more number of wires between S and R. So, now imagine sender has the message m which it wants to privately communicate to the receiver. And to do that, what sender does is the following: It computes a vector of shares according to a n, t secret-sharing scheme, assuming that its secret is the message m.

So, the message m which sender wants to communicate privately to the receiver is treated as the input for the secret-sharing algorithm. And we assume that we are given 1 such secret-sharing algorithm. It could be any n, t secret-sharing algorithm. We are treating the n, t secret-sharing scheme as a black box here. We are not interested in the underlying detail. It could be your Ito's scheme, it could be Benaloh's scheme, it could be Shamir secret-sharing scheme, it could be any n, t secret-sharing scheme.

Now, the secret-sharing scheme will generate n shares for the message m. And since the sharing algorithm will be randomised, that is why I am using this arrow notation to denote the output of the sharing algorithm; I am not using the assignment operator. And now, what sender does is the following: It sends the first share over the first wire, it sends the second share over the second wire, it sends the ith share over the ith wire, and the nth share over the nth wire; as simple as that.

So, intuitively, you can imagine this whole process as the message m is kind of divided into packets. The shares of the message here can be in interpreted as various packets, which overall when combined, gives you the message m. But these packets have the property that, if any t of these packets get compromised, gets intercepted, it does not reveal anything about the underlying message.

That is how you can interpret the division of the message into the shares m_1, m_2, m_3, m_4 and so on. Now, let us see whether the privacy condition is satisfied or not here. So, what will be adversary's view here? Imagine there is a threshold adversary and this threshold adversary can eavesdrop the communication over any 3 out of the 4 wires. It could be, say the first 3 wires, the last 3 wires and so on.

Say for instance, it eavesdrops the last 3 wires. So, it will learn the shares m_2, m_3, m_4. But will learning m_2, m_3, m_4 reveal anything to this adversary? The answer is no. Because, as per the property of this n, t secret-sharing scheme, these shares m_1, m_2, m_3, m_4, have the property that, if any subset of 3 shares are considered, then they are probability distribution is independent of the underlying secret which is used in the secret-sharing algorithm.

And the secret which is used in the secret-sharing algorithm is nothing but the message itself. And this holds not only for the last 3 wires, the adversary can choose any 3 wires; it could be the first 3 wires, or it could be the first, second or fourth wire and so on. The property of this n, t secret-sharing scheme ensures that the probability distribution of any 3 shares out of these n shares is independent of the message. And hence, the privacy property is satisfied.

Now, how can the receiver get back the message? So, receiver will receive the shares m_1 over the first wire, m_2 over the second wire, m_3 over the third wire, and m_4 over the fourth wire; and it will know the reconstruction algorithm of the underlying secret-sharing scheme. It can use the reconstruction algorithm and get back the message m. And correctness here will be satisfied.

By correctness I mean, receiver will be able to recover back the sender's message correctly without any error, if the shares m_1, m_2, m_3 and m_4 are communicated as it is. That means, they are, their contents, their values are not changed. That means, even if adversary is sitting over the second, third and fourth wire, it has not changed any of the bits of m_2 or m_3 and m_4. And that is guaranteed, because, as per my assumption, I am assuming here that adversary is an eavesdropper here.

You might be wondering what if my adversary is not eavesdropper, it can tamper the contents over the channels which are under its control. Well, there are mechanisms to deal with that as well. But since the focus of this course is semi-honest adversaries or eavesdropper, we will be just focusing our attention on eavesdropper; namely, adversary who does not alter the contents of the messages communicated over the channels under its control.

So, this construction will work against the threshold adversary. Now, let us see whether the above approach of computing shares for the sender's message and communicating over the individual channels, work even against a non-threshold adversary as well. And the answer is yes. So, let me consider a non-threshold adversary, where I consider this adversary structure. And this adversary structure represents here that, during the execution of a PSMT protocol, the adversary can either control the wire w_1, or it can control together the second and third wire, or it can control only the fourth wire.

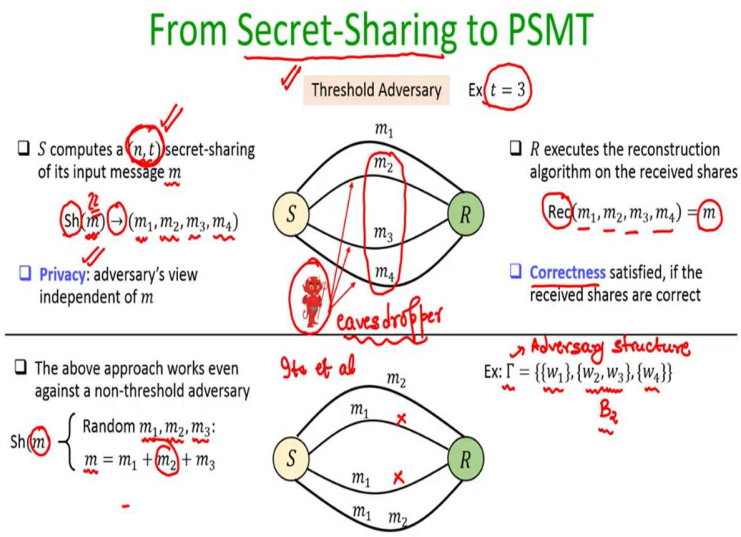
So, now you can see, the cardinality of different subsets in this adversary structure are different. It is not the case that the cardinality of all the subsets in this adversary structure is the same. So, now, let us see whether sender can securely communicate the message here. So, we will run the non-threshold secret-sharing scheme by Ito et al. And if you recall what we do in the secret-sharing scheme of Ito et al., we find out the number of subsets in the adversary structure.

So, we have 3 subsets here. This is 1 bad subset, another bad subset and another bad subset. And the idea behind the Ito's secret-sharing scheme is that, we divide the secret into so many pieces that for each potential bad set in the adversary structure, there should be some piece which should not be available with the parties in that bad set. So, since we have 3 bad sets here, the secret or the sender's message is divided into 3 shares m_1, m_2, m_3, which are random, and which will have the property that they sum up to the message m.

So, here I am assuming that the message m is an element of the group, and the group has a plus operation. Now, the first share of the message, m 1 needs to be communicated to the receiver. Through which of the channels it will be communicated? So, again, m_1 is communicated over all the channels except the channels in the bad subset B_1. So, the bad subset B_1 has wire 1.

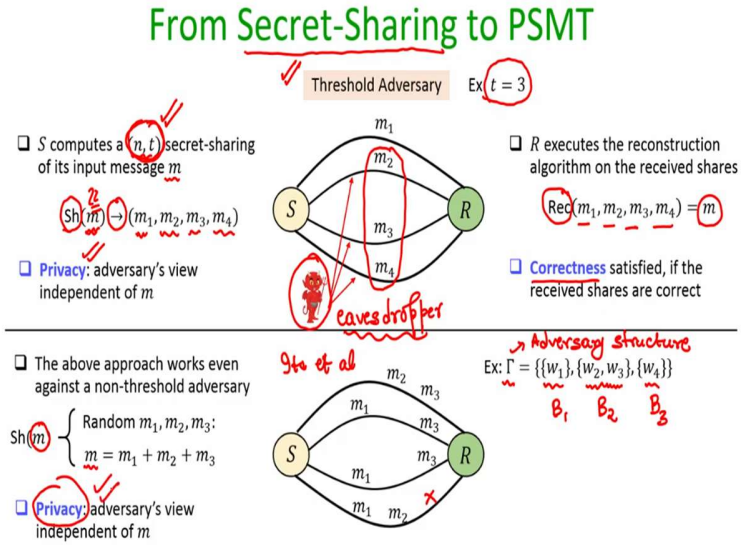
So, wire 1 will not be used to communicate the first piece of the message, but through the remaining wires it will be communicated. And the idea here is that, if during the execution of the protocol, if indeed adversary controls the wires in B_1, it will not be getting the piece m_1. And since m_1 will be missing for the adversary, from its viewpoint, even if it gets the remaining shares of the message, it could be any message which sender has communicated to the receiver.

(Refer Slide Time: 23:01)



In the same way, the piece m_2 will be communicated to the receiver over all the channels except the channels in B_2. So, the B_2 channels, B_2 subset has the channels w_2 and w_3. So, w_2 and w_3 will not be used to communicate m_2; but over the remaining channels, m_2 will be communicated. And again the idea here is that, if actually it is the set of wires in B_2 which gets corrupt by the adversary during the execution of the protocol, then m_2 will be missing, and hence the adversary who is controlling the set of wires in B_2 fails to learn the message m.

(Refer Slide Time: 23:49)

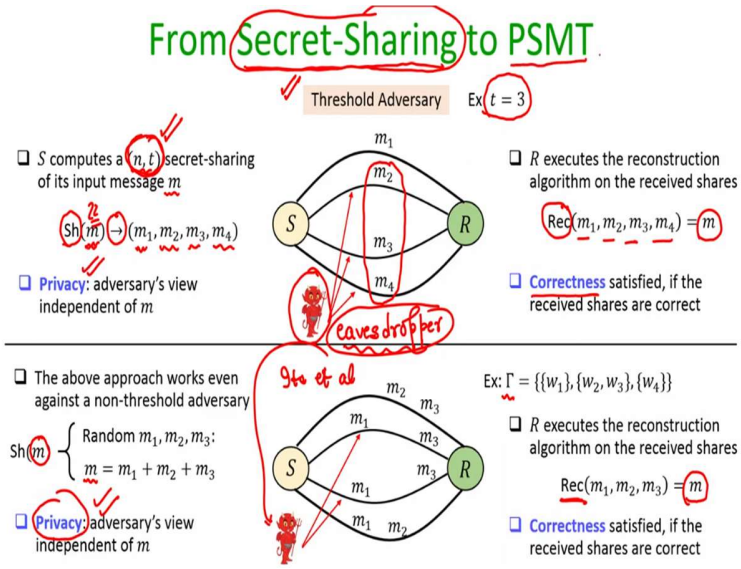


Whereas, the third piece m_3 will be communicated through all the wires except the wires in B_3. So, B_3 has wire w_4; so, wire 4 will not be used to communicate m_3, but m_3 will be communicated over the remaining wires. Again, privacy is very easy to argue here. Namely, it does not matter

which subset of wires from the adversary structure adversary controls. If it is B_1, then m_1 is missing; if it is B_2, then m_2 is missing; and if it is B_3, then m_3 is missing.

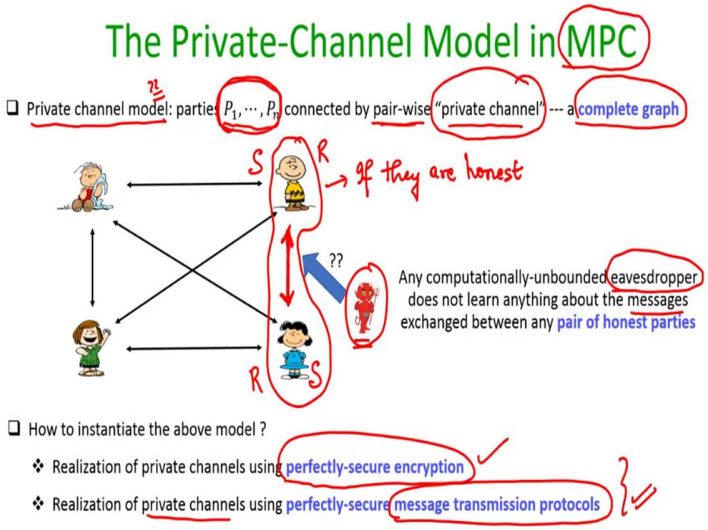
That means, irrespective of which bad subset adversary controls, there is at least 1 share of m which will be missing; and hence, it could be any message m which sender has split and communicated via various channels to the receiver R. So, privacy is guaranteed.

(Refer Slide Time: 24:45)



Now, what about the correctness? Will receiver be able to recover back the message? Yes. Since we are assuming an eavesdropper here, it will be just eavesdropping the contents of the channels, it will not be altering the contents of the channels. And the pieces m_1, m_2, m_3 will be arrived as it is, to the receiving end. And now, the receiver can apply the reconstruction algorithm of the underlying secret-sharing scheme, and it will recover back the message m. So, now you can see that if you have a perfectly-secure secret-sharing scheme, then that can be directly translated to obtain a perfectly-secure message transmission protocol.

(Refer Slide Time: 25:39)



So, that brings us to what we call as the private-channel model in secure multi-party computation protocols, which we will be designing later. So, very often, we use this term private-channel model in MPC. We will say that, we make the assumption that we are in the private-channel model. And by private-channel model, we mean that the parties p_1 to p_n who wants to perform secure computation, they are connected by pairwise private channels.

That means, we abstract the underlying network as a complete graph. And when I say pairwise private channels, that means, if there are 2 parties; suppose these 2 parties are honest and honest means that they are not under the control of the adversary; then, by pairwise private channel, I mean here that whatever communication is going to happen over this channel between these 2 parties, even if there is a computationally unbounded eavesdropper, passive adversary who is monitoring the communication, cannot learn anything about the underlying messages which are communicated by these 2 honest parties.

That is what I mean by the pairwise private-channel model. Of course, if 1 of these 2 parties is the corrupt party, namely, under the control of the adversary, adversary will fully learn what exactly are the underlying messages which are communicated, because, either it will be the sender or the receiver of this channel. But if both the parties, namely the sender and the receiver at the receiving end of this channel are not under the control of the adversary, then a pairwise private channel

means here that, even a computationally unbounded eavesdropper cannot make out anything regarding the communication happening between these pair of parties.

So, now the question is that, how exactly we instantiate this model in practice; how we ensure that between every pair of parties in the system, there indeed exists a private channel; because these parties are finally going to be connected by internet, or they are part of a big network. The parties might be disturbed, located across various parts of the globe; or even if they are within the same city, they might be at different places and so on.

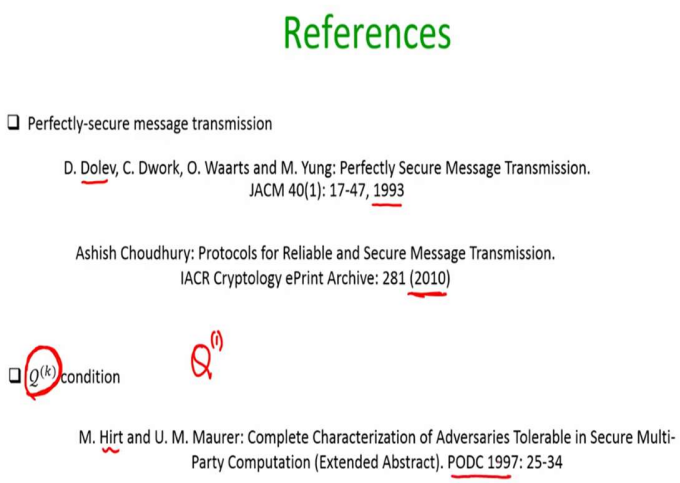
So, how can we ensure that indeed there is a mechanism to do secure communication between every pair of honest parties? So, there are 2 ways to realise the pairwise private channels. One way is that, we use some perfectly-secure encryption scheme, like, say one-time pad. Or, we realise the private channel between these 2 parties by treating them as sender and receiver, and sender and receiver respectively, and executing a perfectly-secure message transmission protocol.

That means, it might be the case that, even though they are not directly connected in the network, they are connected via intermediate nodes. If it is ensured that there are more than $t + 1$ number of node-disjoint channels between these 2 parties, and even if t of them are under the control of this computationally unbounded adversary, we get the effect of this direct pairwise private channel between these 2 parties.

So, that is another way to realise a direct pairwise private channel between any pair of parties in the system. So, the point is that, we will abstract out the underlying network and we will make the assumption that the n parties, they have mechanism to do pairwise private communication. Given this setup, our goal will be to design a protocol for allowing the parties to perform secure computation.

How exactly you instantiate the pairwise private channel? Well, you have one of these, you have either option number 1, use perfectly-secure encryption scheme, or you have the option number 2, use a perfectly-secure message transmission protocol.

(Refer Slide Time: 30:18)



So, with that, I end today's lecture. So, these are the references. So, perfectly-secure message transmission in itself is a very interesting well studied problem in the secure distributed computing community. The problem was introduced by the seminal work of Dolev et al. way back in 1993. And if you want to know more about perfectly-secure message transmission, then you can refer to this PhD thesis.

The $Q^{(1)}$ condition; so, we have used the $Q^{(1)}$ condition against the non-threshold adversary model in our perfectly-secure message transmission protocol, but in general, we can have a condition called $Q^{(k)}$ condition. The $Q^{(k)}$ condition demands that you take union of any potential k subsets from the adversary structure that should not be the entire universal set. Universal set could be either the set of parties or it could be the set of wires. So, this notion of $Q^{(k)}$ condition was introduced by Hirt et al. in this paper in 1997. With that, I end this lecture. Thank you.