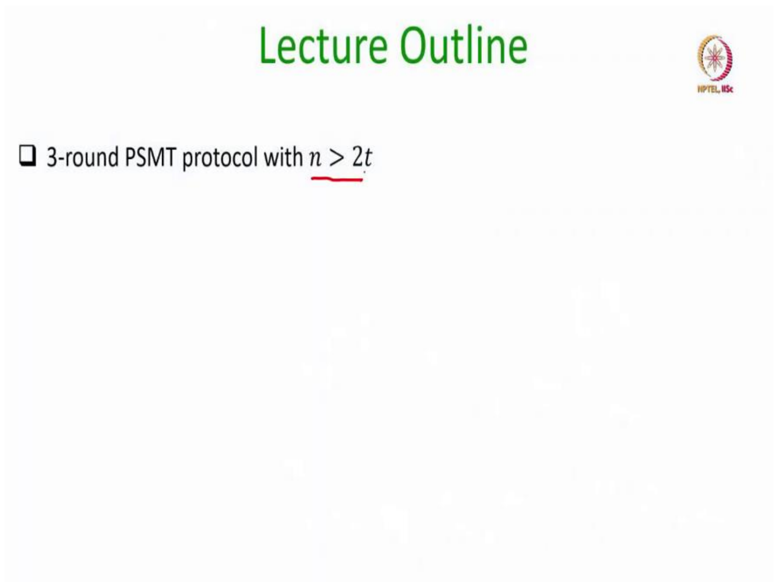
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

Lecture - 25
Multi-Round PSMT Protocol: II

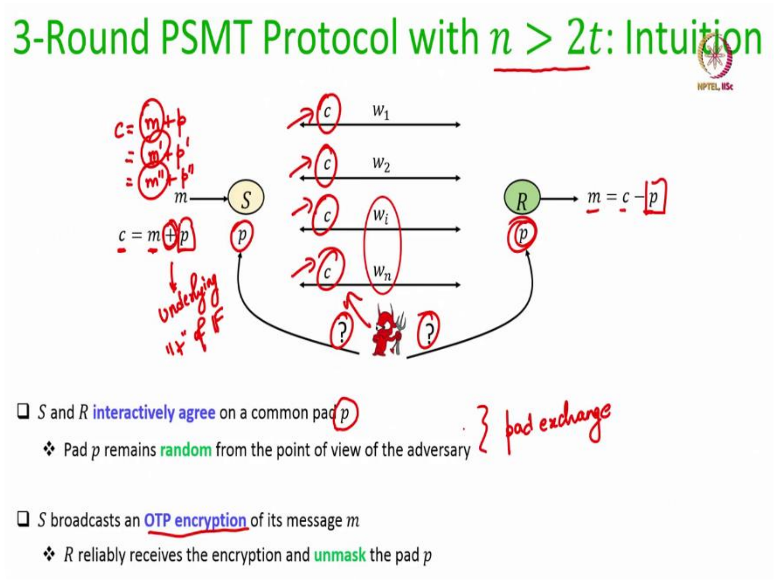
(Refer Slide Time: 00:29)



Lecture Outline

- ❑ 3-round PSMT protocol with $\underline{n > 2t}$

(Refer Slide Time: 00:33)



3-Round PSMT Protocol with $n > 2t$: Intuition

- ❑ S and R **interactively agree** on a common pad p
  - ❖ Pad p remains **random** from the point of view of the adversary

} pad exchange

- ❑ S broadcasts an OTP encryption of its message m
  - ❖ R reliably receives the encryption and **unmask** the pad p

Hello everyone, welcome to this lecture. So, in this lecture we will see a 3-round PSMT protocol with the condition $n > 2t$. So, the underlying intuition idea behind the protocol is as follows. Imagine you have at least $2t + 1$ bidirectional wires between the sender and the receiver of which at most t could be byzantine corrupt then what S and R do is that they run an interactive protocol and that is allowed because we are now taking into consideration bidirectional wires.

So, interaction in both at both the ends is possible S can talk to R, R can send feedback to S and so on and once they are done interacting both should agree upon a common pad p which should be random from the viewpoint of the adversary who would have observed a communication over t channels.

Assume such a pad exchange protocol is there right which gives you these guarantees then to securely send the message m which is the input of the sender can do the following it can compute a one-time pad encryption of its message. So, what is the one-time pad encryption of the message? It is simply the addition of the message with the pad p where the addition is the underlying field operation underlying plus of field and the resultant value will be denoted by c that one-time encryption of the cipher text will be now broadcasted to the receiver over all the channels.

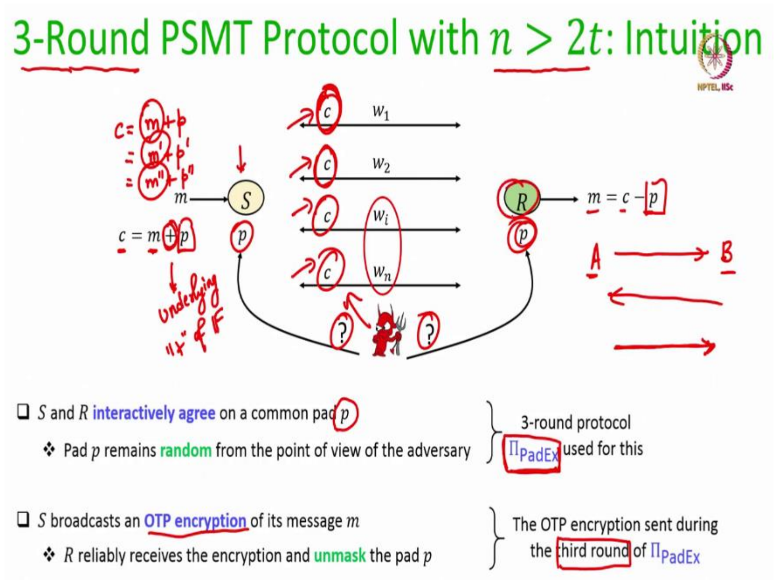
So, even if adversary changes t of the copies of this one-time pad encryption the majority of the wires will be honest they will deliver the actual OTP encryption. So, once receiver recovers c by taking majority it can unmask the same pad p. So, adding the mask p can be considered as masking operation and subtracting can be considered as the unmasking operation.

So, once receiver receives the encryption c, it can unmask the pad p which is also available with the receiver because of this pad exchange protocol and then it can recover back the message and then why the privacy will be achieved? The privacy will be achieved because even though the adversary will be seeing the cipher text c because it is broadcasted it will not be knowing the exact pad p.

And since the pad p is random from the viewpoint of the adversary, the value c is also random for the adversary; that means, it could be the case that $c = m + p$ or it could be the case that $c = m' + p'$ or it could be the case that $c = m'' + p''$ and so on.

So, every candidate message from the message space could have been encrypted in this cipher text c. Since the exact value of the pad p is not known to the adversary the underlying message will be completely random from the viewpoint of the adversary. So, that is the intuition behind this 3-round PSMT protocol.

(Refer Slide Time: 04:09)



3-Round PSMT Protocol with $n > 2t$: Intuition

- ❑ S and R **interactively agree** on a common pad p
  - ❖ Pad p remains **random** from the point of view of the adversary

} 3-round protocol Π_{PadEx} used for this

- ❑ S broadcasts an **OTP encryption** of its message m
  - ❖ R reliably receives the encryption and **unmask** the pad p

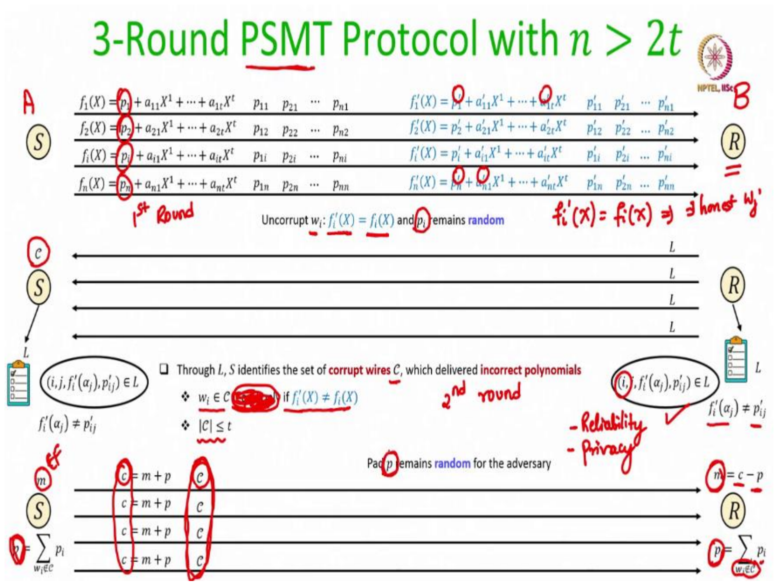
} The OTP encryption sent during the third round of Π_{PadEx}

Now, to do this pad exchange we will use the pad exchange protocol which we had discussed in the last lecture that was a 3-round pad exchange protocol. But you might be wondering that if the pad exchange requires 3-round then when exactly is the message communicated it is the when exactly is the OTP encryption communicated is it during the fifth round or the fourth round or the subsequent round well that is not the case because overall, we require the PSMT protocol to be at 3-round protocol.

So, what we will do is, we will run this pad exchange protocol in parallel with the OTP encryption communication where the OTP encryption will be communicated by the sender during the third round of the pad exchange protocol. So, recall that the pad exchange protocol was a 3-round protocol where A sends polynomials to B during round 1 and then during round 2, B sends back the conflict list and then during the third round, A sends the identity of the faulty wires which have delivered wrong polynomials to B.

What I am proposing here is that during the third round, when the identity of the faulty wires are communicated to B, sender will also club in the OTP encryption and at the end of the third round receiver, who will be playing the role of party B in the pad exchange protocol, will be learning the identity of the faulty wires which have delivered wrong polynomials. Based on that receiver will be able to recover back the pad which has been used by A which in this case will be the sender and once the pad is known then as I have discussed here receiver will be able to unmask the pad from the OTP encryption.

(Refer Slide Time: 06:15)



3-Round PSMT Protocol with $n > 2t$

- Uncorrupt w_i: $f_i'(X) = f_i(X)$ and p_i remains random
- Through L, S identifies the set of corrupt wires $\mathcal{C}$, which delivered incorrect polynomials
  - $w_i \in \mathcal{C}$ [illegible] if $f_i'(X) \neq f_i(X)$
  - $|\mathcal{C}| \leq t$
- Pad p remains random for the adversary

A
B
S
R
$f_1(X) = p_1 + a_{11}X^1 + \cdots + a_{1t}X^t$ p_{11} p_{21} $\cdots$ p_{n1}
$f_2(X) = p_2 + a_{21}X^1 + \cdots + a_{2t}X^t$ p_{12} p_{22} $\cdots$ p_{n2}
$f_i(X) = p_i + a_{i1}X^1 + \cdots + a_{it}X^t$ p_{1i} p_{2i} $\cdots$ p_{ni}
$f_n(X) = p_n + a_{n1}X^1 + \cdots + a_{nt}X^t$ p_{1n} p_{2n} $\cdots$ p_{nn}
$f_1'(X) = p_1' + a_{11}'X^1 + \cdots + a_{1t}'X^t$ p_{11}' p_{21}' $\cdots$ p_{n1}'
$f_2'(X) = p_2' + a_{21}'X^1 + \cdots + a_{2t}'X^t$ p_{12}' p_{22}' $\cdots$ p_{n2}'
$f_i'(X) = p_i' + a_{i1}'X^1 + \cdots + a_{it}'X^t$ p_{1i}' p_{2i}' $\cdots$ p_{ni}'
$f_n'(X) = p_n' + a_{n1}'X^1 + \cdots + a_{nt}'X^t$ p_{1n}' p_{2n}' $\cdots$ p_{nn}'
1st Round
$f_i'(x) = f_i(x) \Rightarrow \exists$ honest w_j'
c
L
$(i, j, f_i'(\alpha_j), p_{ij}') \in L$
$f_i'(\alpha_j) \neq p_{ij}'$
2nd round
$(i, f_i'(\alpha_j), p_{ij}') \in L$
$f_i'(\alpha_j) \neq p_{ij}'$
- Reliability
- Privacy
m
$c = m + p$
c
$m = c - p$
$p = \sum_{w_i \notin \mathcal{C}} p_i$
NPTEL, IISc

So, let us see the full details here how we combine these two things together the pad exchange protocol with the OTP encryption broadcast during the third round. So, during the first round, sender will run the first round of pad exchange protocol. So, sender will be playing the role of A as per the pad exchange protocol and receiver will be playing the role of B as per the pad exchange protocol.

So, sender will be sending n random t degree polynomials whose constant terms are also random values and the values of these polynomials at n points $\alpha_1 \ldots \alpha_n$. Now, up to t wires may deliver incorrect polynomials. So, we do not know which polynomials are delivered correctly which polynomials are delivered incorrectly.

So, that is why we are using this prime notation to represent the polynomials received by the ith party and in the same way receiver will receive the supposedly values of these polynomials at $\alpha_1 \ldots \alpha_n$ again we do not know whether these values have been delivered correctly or incorrectly.

So, what we know is that if the wire w_i is uncorrupted then the ith polynomial which has been delivered to the receiver will be same as will be the same as the polynomial sent by the sender and its constant term will be random because for such polynomials adversary will be only learning up to t points.

Now, based on the values which are received by the receiver a receiver will prepare a conflict list and it will broadcast the conflict list whenever there is a conflict between the ith delivered polynomial with respect to its value at the jth point, when there will be a conflict between the ith wire and the jth wire, and all such conflicts are delivered to the sender irrespective of the corruption happening over t wires because they are broadcasted.

So, at the end of the round 2 sender also will have the list of all such conflicts and through such conflicts sender will be able to identify all the corrupt wires $\mathcal{C}$ which have delivered incorrect polynomials during round 1 to the receiver. This is because whenever there is a wire w_i right which delivers the incorrect which delivers an incorrect polynomial corresponding to that there will be at least one on is w_j such that this conflict will be present.

And when sender receives those conflicts and compares those conflicting values with whatever values it has sent during round 1, it will come to the conclusion that it is the wire w_i which is the corrupt wire and it would include such wires w_i in this set $\mathcal{C}$.

So, if the ith polynomial has been delivered incorrectly then the ith wire will be present in the set $\mathcal{C}$ and the cardinality of this set $\mathcal{C}$ is at most t because there can be at most t corrupt wires in the system. So, this is the first round from sender to receiver, this was the second round from receiver to sender now there is a third round where sender has a message m belonging to the field $\mathbb{F}$ which is the sender's input for the PSMT protocol.

Now, what sender does is the following, it computes the following pad p which is the summation of the constant term of all the polynomials which were delivered over the wires outside the set $\mathcal{C}$; that means, those polynomials have been delivered in the same way correctly in the same fashion to the receiver; that means, their constant terms are same both for the sender as well as for the receiver.

And now what sender does is the following it broadcasts the identity of the faulty set of wires right namely the wires which have delivered incorrect polynomials and the OTP encryption. What the receiver will do is that since the list of faulty wires which have delivered the wrong polynomials is broadcasted, receiver also will be able to recover it properly by taking the majority even if up to t wires change the contents of this $\mathcal{C}$ still by taking the majority receiver will be able to recover back the set $\mathcal{C}$.

So, it can also exclude though the polynomials which were delivered over those wires and focus only on the constant term of the remaining polynomials sum them and get the pad p and the receiver's version of the pad p will be same as the sender's version of the pad p. Because all the wires which are not present in the set $\mathcal{C}$ for those wires $f_i(X)$ will be same as $f_i'(X)$.

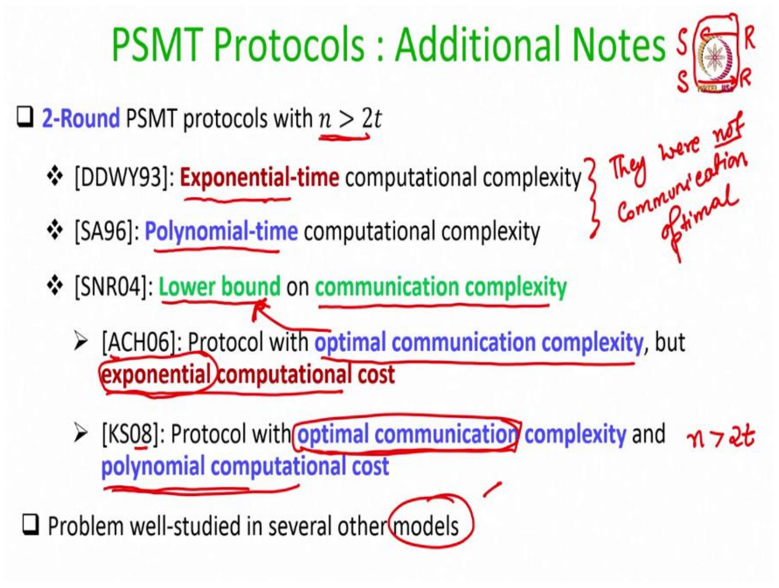
Now once it has the pad p anyhow it is also receiving the OTP encryption how because it can take the version of the OTP encryption which is received over majority of the wires because it is broadcasted. So, even if up to t wires deliver incorrupt even if up to t wires deliver incorrect OTP encryption receiver will be to able to recover back the right version and then it can unmask the pad p from the OTP encryption to get back the message.

Now, why this. So, let us now argue that why this protocol is a PSMT protocol. So, recall that PSMT requires two properties first is the reliability namely receiver should be able to output sender's message and that follows from the fact that both sender and receiver will be computing the same pad p.

And the second property that we require from PSMT is that of privacy which demands that senders view which demands that adversaries view adversary who is controlling over t channel its view should be independent of senders message; that means, whatever information is learnt by the adversary sitting over t wires that should not help the adversary to infer whether the sender message was m or m' and that follows from the fact that the pad p is random for the for the adversary.

And why the pad p is random for the adversary? That we have proved already in the context of pad exchange protocol.

(Refer Slide Time: 14:37)


PSMT Protocols : Additional Notes

- 2-Round PSMT protocols with $n > 2t$
  - [DDWY93]: Exponential-time computational complexity
  - [SA96]: Polynomial-time computational complexity
  - [SNR04]: Lower bound on communication complexity
    - [ACH06]: Protocol with optimal communication complexity, but exponential computational cost
    - [KS08]: Protocol with optimal communication complexity and polynomial computational cost
- Problem well-studied in several other models

So, that completes the description of our 3-round PSMT protocol now you might be wondering that can we reduce the number of rounds to 2 and the answer is yes. Indeed, we can have a 2-round PSMT protocol with $n > 2t$ condition and in that 2-round PSMT protocol the first round will be from the receiver to sender and then the second round will be from sender to receiver.

Because if the first round is from sender to receiver, then the second round is totally useless and then we must go to the third round to do the secure communication. So, if at all the PSMT protocol has to be completed in 2 rounds the first round has to be from the receiver to sender and the second round will be from the sender to receiver. So, in the original paper by Dolev et al an exponential time 2-round protocol has been proposed that is exponential time because it requires exponential amount of computation complexity.

Then later on, in an interesting work of Sayeed Abu-Amara et al, they gave a polynomial time protocol which is a polynomial time 2-round protocol with the condition $n > 2t$. In a very interesting work by Srinathan et al lower bounds are derived for the communication complexity of PSMT protocols namely they derive bounds on what is the minimum communication required by any PSMT protocol that PSMT protocol need not be based on polynomials.

It can be based on other primitives irrespective of what primitives are used what is the amount of communication that any PSMT protocol has to do; that means, how much communication sender and the receiver has to necessarily do in any PSMT protocols those lower bounds were derived in this work by Srinathan et al and it was in 2006 that Aggrawal et al gave a 2-round PSMT protocol with $n > 2t$ condition which matches the lower bound of Srinathan et al namely that protocol has optimal communication complexity.

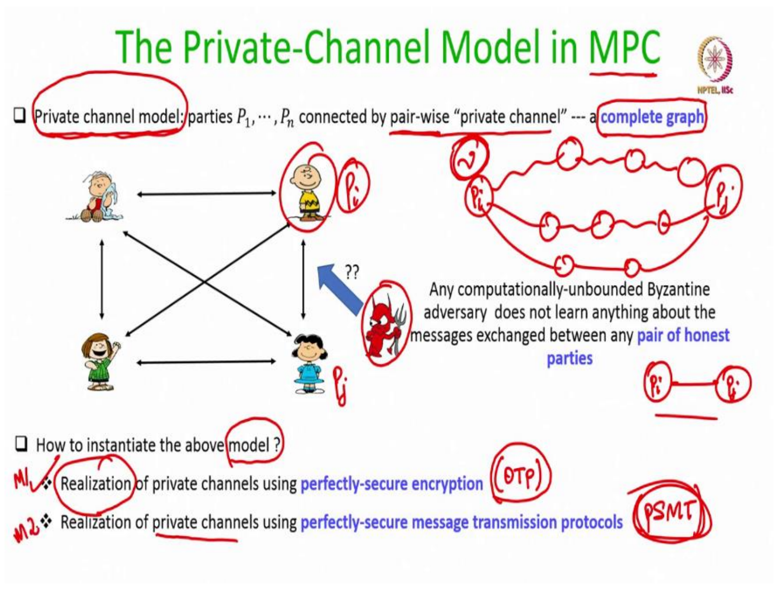
Optimal in the sense that it exactly satisfies the lower bound as given by Srinathan et al; however, the downside of that protocol was that it requires exponential amount of computation. So, the earlier protocols the protocol of Dolev et al and Sayeed Abu Amara et al they are they were not communication optimal even though they were 2-round protocols, they were not the protocol require more communication compared to the optimal communication.

It was the protocol of Aggrawal et al which is the first 2-round protocol with optimal communication optimal communication complexity, but the downside was that the amount of computation required is exponential and then finally, in 2008 in a very beautiful work Kurosawa et al gave a 2-round protocol with polynomial computation cost and which also requires optimal amount of communication.

So, this protocol is optimal in every aspect it is designed with exactly the condition $n > 2t$ which is the necessary condition for multi round PSMT, it requires polynomial amount of computation, and it has the optimal amount of communication.

So, this PSMT problem is a very well studied problem, and it has been studied in several other models, models means depending upon whether the adversary is threshold or a non threshold adversary, whether the adversary is static or dynamic, whether the underlying communication graph models asynchronous communication or asynchronous communication and so on.

(Refer Slide Time: 19:04)


NPTEL, IISc
P_i
P_j
??
P_i
P_j
P_i
P_j
M1
M2
OTP
PSMT

The Private-Channel Model in MPC

- Private channel model: parties $P_1, \cdots, P_n$ connected by pair-wise "private channel" --- a complete graph

Any computationally-unbounded Byzantine adversary does not learn anything about the messages exchanged between any pair of honest parties

- How to instantiate the above model ?
  - ❖ Realization of private channels using perfectly-secure encryption
  - ❖ Realization of private channels using perfectly-secure message transmission protocols

So, based on the discussion on PSMT, we can conclude the following whenever we say a private channel model in the context of MPC. So, remember our goal is to eventually solve the MPC problem where we have a set of mutually distrusting parties and they want to do secure they want to compute some function of their inputs without disclosing their inputs and in the MPC protocols we often talk about a private channel model where we make the assumption that the parties are connected by pair wise private channel; that means, the underlying communication graph is a complete graph.

And; that means, if the P_i and P_j are honest and they are talking with each other over this channel, then even there is an adversary who is computationally unbounded should not learn anything about the messages exchanged between the parties P_i and P_j that is the assumption in the private channel model. Now how do we instantiate the above model?

Because in practice in real life the underlying communication graph need not be a complete graph; that means, party P_i might be in one continent and P_j might be in another continent and there may not be a direct channel a secure direct channel between P_i and P_j over which they can talk.

So, the so called secure channel between P_i and P_j can be realized or can be emulated by either using perfectly secure encryption namely say one time pad encryptions where P_i and P_j will already have beforehand huge collection of one time pads and whenever as part of the MPC protocol P_i is supposed to send any message to P_j those pads can be used to compute the OTP encryptions and sent to party P_j through the intermediate networks.

That could be one option to instantiate the above secured channel or we can instantiate the above secure channels between the pair of parties using a PSMT protocol; that means, even though there is no direct secure channel between P_i and P_j and say P_i and P_j are a part of a huge network where there are several intermediate nodes between P_i and P_j along those paths ok.

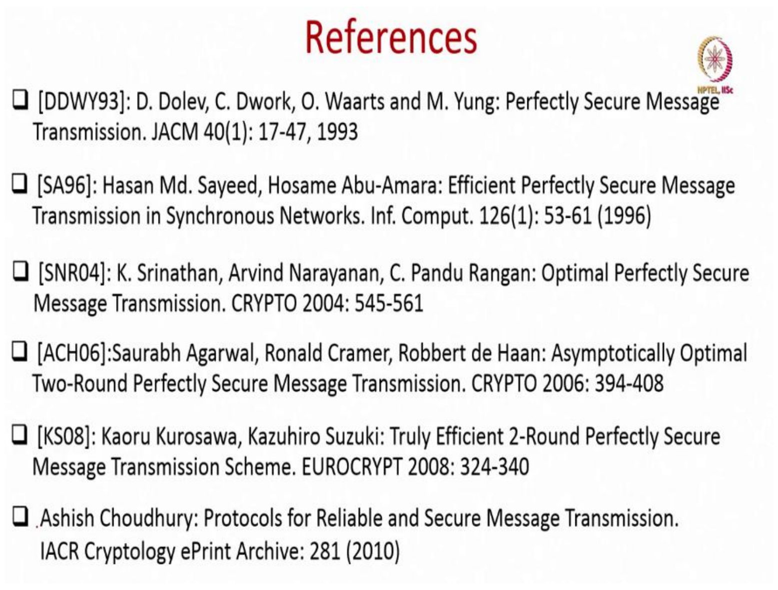
Then whenever as part of the MPC protocol, P_i is supposed to send some value say v to P_j instead of sending the value v in clear along this intermediate nodes party P_i can use a PSMT protocol with input v and send that message v to P_j and that will have the same effect as if there is a direct channel between P_i and P_j.

So, either you need to ensure that before the MPC protocol starts P_i and P_j already have agreed upon a sufficient collection of one time pads which can be used to do the actual encryptions of the values which have to be communicated as part of the MPC protocol or at the runtime whatever values have to be communicated from the ith party to the jth party as part of the MPC protocol.

Those values are communicated using a PSMT protocol by either asking P_i to play the role of the sender or P_j the role of the sender depending upon whether P_i has to send a message to P_j or whether P_j has to send a message to P_i. But irrespective of the case now from now onwards whenever we say that we are in the private channel model, we will assume that the underlying communication graph is a complete graph where there is the dedicated pair wise secure channel private channel between every pair of parties and then assuming such a network, we will design MPC protocols.

To emulate that complete graph either you can use mechanism number 1 or you can use mechanism number 2.

(Refer Slide Time: 23:46)



References

- ❑ [DDWY93]: D. Dolev, C. Dwork, O. Waarts and M. Yung: Perfectly Secure Message Transmission. JACM 40(1): 17-47, 1993
- ❑ [SA96]: Hasan Md. Sayeed, Hosame Abu-Amara: Efficient Perfectly Secure Message Transmission in Synchronous Networks. Inf. Comput. 126(1): 53-61 (1996)
- ❑ [SNR04]: K. Srinathan, Arvind Narayanan, C. Pandu Rangan: Optimal Perfectly Secure Message Transmission. CRYPTO 2004: 545-561
- ❑ [ACH06]:Saurabh Agarwal, Ronald Cramer, Robbert de Haan: Asymptotically Optimal Two-Round Perfectly Secure Message Transmission. CRYPTO 2006: 394-408
- ❑ [KS08]: Kaoru Kurosawa, Kazuhiro Suzuki: Truly Efficient 2-Round Perfectly Secure Message Transmission Scheme. EUROCRYPT 2008: 324-340
- ❑ Ashish Choudhury: Protocols for Reliable and Secure Message Transmission. IACR Cryptology ePrint Archive: 281 (2010)

So, as I said whole area of perfectly secure message transmission is a very active area of is well which not be a very active area of research right now, but it is a very fundamental problem in a secure distributed computing and there are several fundamental results related to this area. So, these are some of the references which you can use to know more about the perfectly secure message transmission problem.

Thank you.