


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

Lecture - 01
What is Secure Multi-Party Computation (MPC)?

Hello everyone, welcome to this lecture. The outline for this lecture is as follows; in this lecture we will discuss what secure multi party computation, or secure MPC, is.

(Refer Slide Time: 00:34)

Lecture Overview



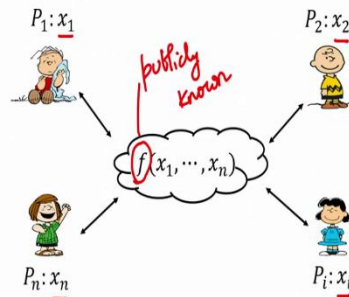
- ☐ What is secure MPC ?
 - ❖ Some real-world applications
- ☐ Various Dimensions to Study secure MPC ?
- ☐ Challenges in dealing with malicious adversaries

We will see some real-world applications. We will see various dimensions in which one can study the secure MPC problem. And since this course is all about studying MPC tolerating malicious adversaries, we will also discuss the challenges in dealing with malicious adversaries.

(Refer Slide Time: 00:59)

Privacy Preserving Information Processing (Computation)

□ Several distributed applications require “availability” and “confidentiality” of sensitive data



❖ Mutually distrustful entities with private data

❖ Jointly want to perform some computation on their private data without revealing their inputs

So, let us start with the motivation behind secure multi party computation. There are several applications which are distributed applications, and in such applications, we require privacy preserving information processing. Namely, in such applications we have several entities. In fact, we call them as mutually distrustful entities. Say we have n number of entities P_1, P_2, P_i, P_n . And they are mutually distrustful in the sense they do not trust each other, and each entity has some confidential data associated with it.

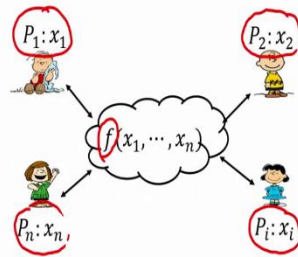
So, for instance P_1 has the data x_1 , P_2 has the data x_2 , P_i has the data x_i and P_n has the data x_n . So, this data could be an any kind of abstract data it could be for instance their personal details or it could be their salary information or it could be their biometric details or it could be some huge database. So, we do not want to go into the exact details of the data, but we abstract it out that each entity here has some private or confidential data.

Even though the entities are mutually distrustful they would still like to perform some joint computation on their data without revealing their inputs. So, the computation which they are interested to perform can be abstracted by some function f which is publicly known. And this function is an n -ary function; that means, it takes n inputs where the i th input is going to be provided by the i th party and the parties want to execute some protocol which allows them to learn the outcome of this function f on the input x_1, \dots, x_n .

But in the process no party should learn anything additional beyond what it can infer from its input and the function output.

(Refer Slide Time: 03:06)

Privacy Preserving Information Processing (Computation)



□ Computing on private data --- secure multi-party computation (MPC)

❖ Cannot be handled by traditional encryption mechanisms

❖ Encryption : provides only confidentiality, no availability

↓ two conflicting goals:
- Available
- Confidential

□ Analogy : Jewels are safe in a locker but at the same time one would like to wear them for some occasion

So, this problem is also called as secure multi party computation or MPC problem. Why secure multi party computation? Because we have multiple parties involved here and together, they would like to perform some computation in a secure way. We often call this problem as computing on private data because we have several entities involved here, and while each of these entities has some private data, we would like to perform some computation involving the sensitive data of all the entities.

So, you might be wondering if we can solve this problem by asking or by letting each entity to encrypt its data x_i ; P_1 can encrypt its data x_1 and make it available to everyone, P_2 similarly can encrypt its data and make it available to everyone, P_i encrypts its data x_i and makes it to available to everyone and P_n encrypts its data and makes it available to everyone.

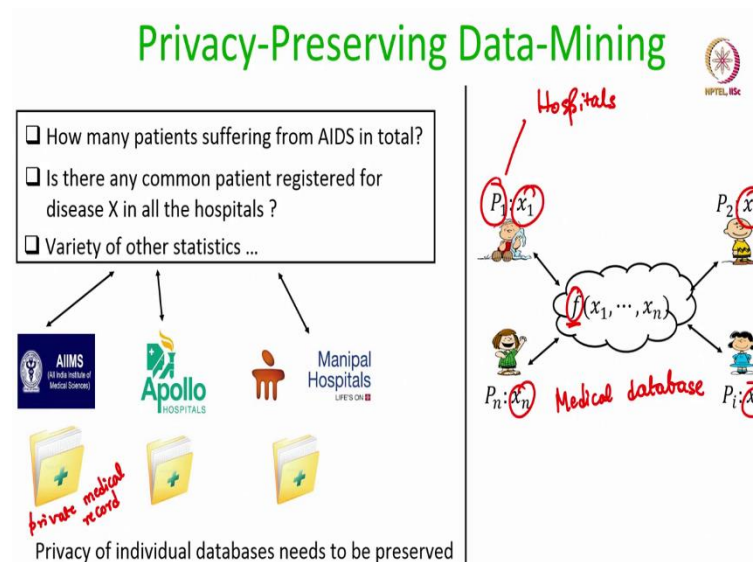
Well, that ensures that the data remains confidential; in the sense that as soon as I encrypt the data it becomes jargon for everyone else. No one can figure out exactly what contents are encrypted. But then, you cannot perform the computation, or you cannot compute the value of the function f on the encrypted data.

So, in some sense encryption, solves the problem of confidentiality; it ensures that the data remains confidential, but it does not make the data available so that computation can be carried out. So, it looks like that we are now trying to solve a problem, secure MPC problem, which has two conflicting goals. Namely, we want the data to be available

because only when the data is available, one can perform any kind of computation on the data. But at the same time, we want that the data should remain confidential.

And this two looks like some conflicting goals and it might look that this problem is not possible to be solved. But in this course, we will see lots of protocols, algorithms on how to solve this problem. An analogy that we can consider here is that of jewels and a locker. So, if you have if you can if you imagine that your sensitive data is jewels, precious jewels which you do not want to be stolen, that you can preserve them by locking in a locker. But at the same time you would like to wear them for some occasions.

(Refer Slide Time: 06:14)



So, now, let us see various real-world applications, examples which are which can be abstracted by this generic problem of secure multi party computation. So, the first problem is privacy preserving data mining. Consider a scenario where we have several hospitals right. So, in this example I am taking three hospitals and each hospital has its own private medical record namely, the patient database. And medical record medical database is a sensitive information which cannot be disclosed in public domain due to legal implications if an any hospital tries to do that.

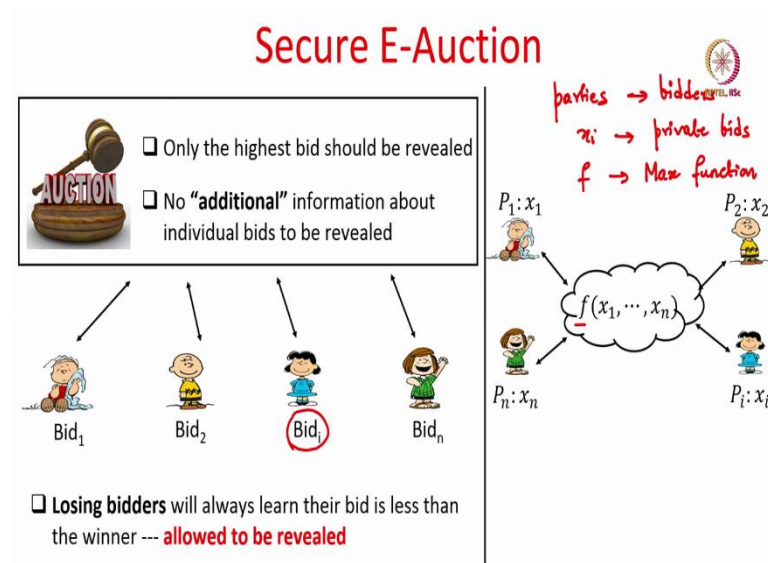
But there might be a scenario where, say, all the hospitals would like to run a protocol to perform a data mining operation across the databases of all the hospitals. So, for instance they might want to compute statistics like how many patients are suffering from AIDS in

total or is there any common patient registered for some specific disease in all the hospitals and like that, a variety of other interesting statistics.

Of course, they can compute the statistics if each hospital makes its medical record available in the public domain, but as I said there are various legal implications regarding that. So, now, the question is, is it possible to perform any kind of data mining operations in a privacy preserving way? So, now, let us see how this problem can be abstracted by the problem of secure multi party computation.

So, here the parties are nothing but hospitals and the private data associated with every party is nothing but the medical records or medical database. And the function f which the parties would like to compute corresponds to the data mining operation or the query which the parties would like to run on the joint database.

(Refer Slide Time: 08:49)



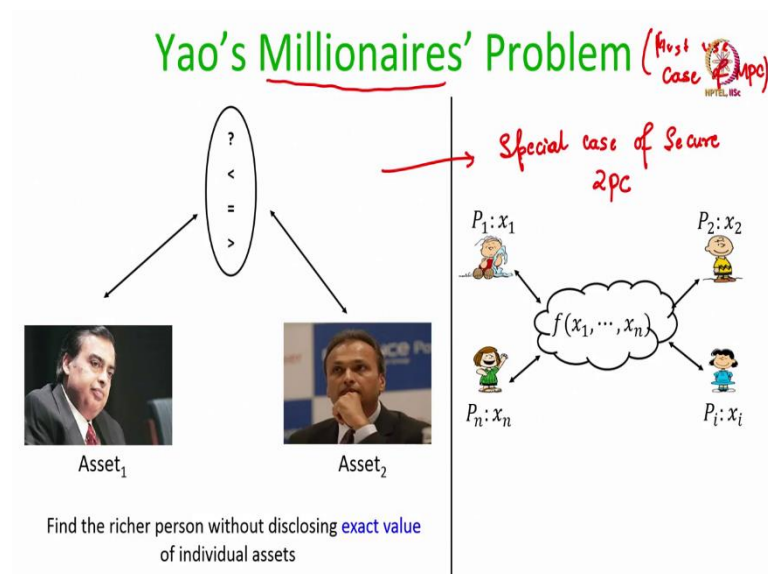
Consider the problem of secure E-Auction where we have n bidders and they are bidding for some valuable object. And we would like that only the highest bid should be revealed during the bidding process and no bidder should learn anything additional about the individual bids. Of course, the losing bidders will learn whether the value of their bid is less than the winner or not that is allowed to be revealed.

Remember the problem description is that if I am the i th party then based on my own input and the function output whatever I am allowed to learn that is fine, but nothing additional

should be learnt. So, if I am the i th bidder and if I am not selected as the winner in the secure E-Auction protocol then I know that the value of my bid is less than the winning bid; that is allowed to be revealed that is not a breach of information.

So, again you can see that how the secure MPC problem abstracts this application. So, the parties are nothing here but the bidders, parties correspond to the bidders and x_i 's correspond to the private bids and the function f corresponds to the max function.

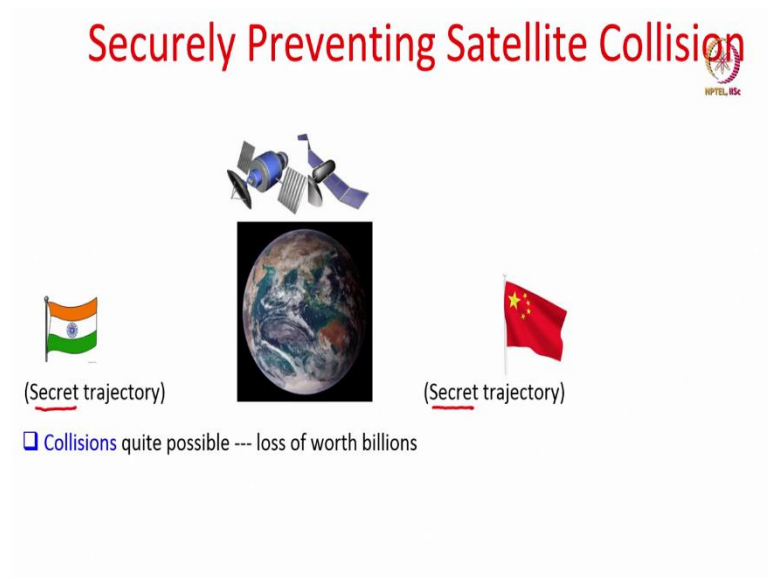
(Refer Slide Time: 10:22)



Consider another interesting problem, the Yao's millionaires' problem. In fact, this is the first use case of MPC. The problem of MPC was introduced by Turing award winner Andrew Yao in his seminal paper and in this in his paper he motivated the problem with this example he called this problem as Yao's millionaires' problem. So, imagine we have 2 millionaires who are not in talking terms with each other, but still they are interested to find out who is richer among them without disclosing the exact value of the asset.

So, this is a special case this is this problem is nothing but a special case of secure 2 PC; 2 party computation because here we have only 2 parties involved. Of course, you can imagine that there are n number of millionaires where n is more than 2 and they are interested to find out who is richer between them that is something like our E-Auction problem.

(Refer Slide Time: 11:43)





Now consider this very interesting application of secure multi party computation. Namely how we can ensure that satellite collision does not happen but that too in a privacy preserving fashion. So, imagine we have two countries who are not in talking terms with each other and they launched their spy satellites. So, since they are launching the spy satellite the trajectory information of their individual satellites will be secret information, because it is a spy satellite. If it is not secret information, then the other country can of course go and destroy the satellite of the other country.


So, that is why they would like to launch their respective spy satellites in a secret way where the trajectory information is kept hidden. But since both the countries are launching their individual satellites without knowing each other's trajectory information it is quite possible that collision occurs between the two satellites.

(Refer Slide Time: 12:50)

Securely Preventing Satellite Collision

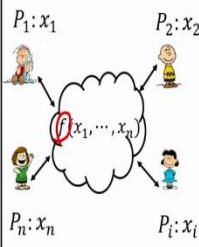

(Secret trajectory)




(Secret trajectory)

- ❑ Collisions quite possible --- loss of worth billions
 - ❖ Several collisions reported in the past
- ❑ Find chances of collision **without disclosing** the trajectories

$n = 2$ parties

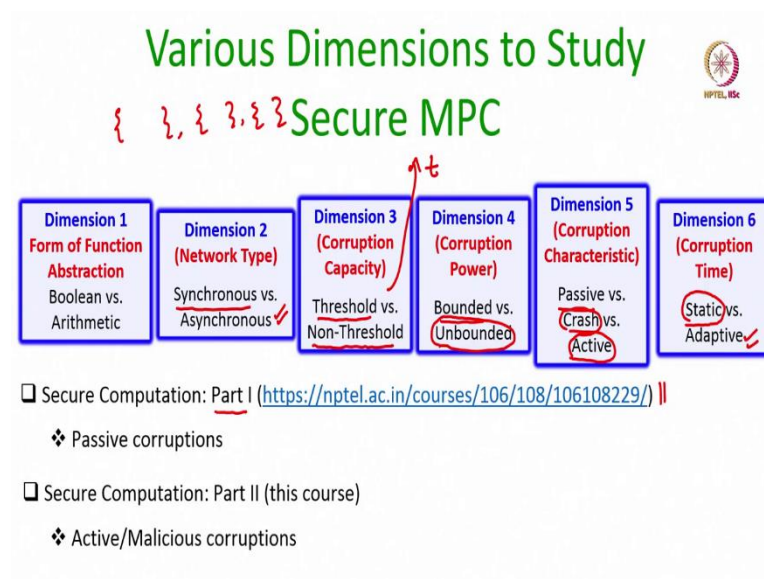


And if you search in Google, you can find out that several such collisions have been reported in the past. So, it is not some fancy example. And if any such collision occurs it causes a loss of worth billions and more than that, the debris which are left over in the space is a serious issue. So, now, the question is, is it possible for these two countries to find out what is the probability is or what is the chances of collision are without disclosing their individual trajectory information?

Of course, if both the countries disclose their trajectory information to each other then they can find out whether there is a possibility of collision or not. But trajectory information is the is the secret information the secret data available with the respective parties which they do not want to disclose. So, again you can see that how this problem very nicely fits into the paradigm of secure multi party computation we have $n = 2$ parties and their x_i information is nothing but their individual trajectory information.

And the function that they are interested to compute is whether what the probability of collision is.

(Refer Slide Time: 14:12)



So, like this there are several real-world applications which can be very nicely abstracted by the problem of secure multi party computation. And this problem is like the Holy Grail problem of secure distributed computing its one of the most fundamental problems in secure distributed computing. It has been studied enormously over the last 4 decades in various dimensions.

So, let us list down the various dimensions in which one can study the secure MPC problem. Dimension number 1 is defined based on how we abstract the function to be securely computed. So, there are two popular ways to abstract out the function which the parties would like to securely compute; either we assume that the function is represented as a Boolean circuit consisting of your logical gates, say the NAND gates, NOR gates and so on.

Or we assume that the function to be securely computed is abstracted or represented by some arithmetic circuit over some appropriate algebraic structure. It could be a group, it could be a ring, it could be a field and so on. Dimension two is based on what kind of communication network is available among the parties. So, whenever we want to design a secure MPC protocol according to the protocol, the parties must exchange messages and those messages will be exchanged over the underlying network.

Now, there are two ways two types of communication models which are considered in the literature and the first model is the synchronous model, where we assume that the

communication channels among the parties are synchronized through some global clock. That implies that if I am one of the sender parties and if I am supposed to send some message according to the protocol, then the receiving party will know within what time that message is supposed to be received by that receiving party.

So, for instance that channel delay could be some minutes, it could be some order of minutes, it could be some order of hours, and it could be order of days. But irrespective of the case, the channel delay will be publicly known, and as a result of that, if any expected message is not received by recipient in the protocol execution, then the recipient can always label the sender party as a corrupt sender party.

A more practical communication model is the asynchronous communication model, where we do not make any such assumption regarding the channel delays. That means, if I send any message as a party during the protocol execution, then the only guarantee will be that it will be delivered eventually, but no one will be knowing when it is going to be delivered.

Moreover, the order in which I have sent the messages not be preserved. When the messages are delivered to the corresponding recipient it might be reordered. Or it might happen that whatever I send tomorrow gets delivered and whatever I have sent today gets delayed by some arbitrary amount, but it will be eventually delivered.

And this asynchronous communication model very nicely captures real world networks like the internet, where there where no one can give you an upper bound, strict upper bound, on the maximum delay within which any message will be delivered to the corresponding receiving party right. However, designing asynchronous protocols is very challenging when compared to designing synchronous protocols.

The third dimension is based on the corruption capacity. So, remember we are assuming that the parties are mutually distrusting and to model the distrust in the system, we assume that we have a centralized adversary who can capture or who can control a subset of the parties. Now there are two ways by which we can model the subset or the cardinality of the subset of the parties which can be controlled by the adversary.

In the threshold model we assume that there is some publicly known threshold, say t , and adversary can corrupt at most t out of n parties they could be any subset of t parties. The exact identity of those t parties will not be known before the starting of the protocol. But

everyone will be knowing that during the protocol execution at most t entities can get corrupt by some adversary.

Whereas, a non threshold adversary is a more generalized form of adversary where we model the corruption capacity through an adversary structure, which is a collection of subsets of potentially corrupted subsets of parties right. That means, we will have several subsets given each of this is a subset of your n parties of various cardinalities and during the protocol execution any of these subsets could be chosen by the adversary for corruption right so, that is called non threshold adversary.

Dimension 4 is the corruption power; that means, how much resources is available with the adversaries. So, here we have two kinds of adversaries which are considered computationally bounded adversaries and computationally unbounded adversaries. Computationally bounded adversary means their running time is bounded by some polynomial function in some appropriate security parameter. And that allows you to use cryptographic tools; modern cryptographic tools like signature schemes and public key encryption schemes and symmetric key encryption schemes and so on.

Whereas the other stream is the unbounded adversary model adversarial model where we do not put any restriction whatsoever on the computing resources or the running time of the adversary. And as a result if you are designing a protocol tolerating computationally unbounded adversaries then you cannot use modern cryptographic tools; because the security of all modern cryptographic tools is based on so called hardness assumptions.

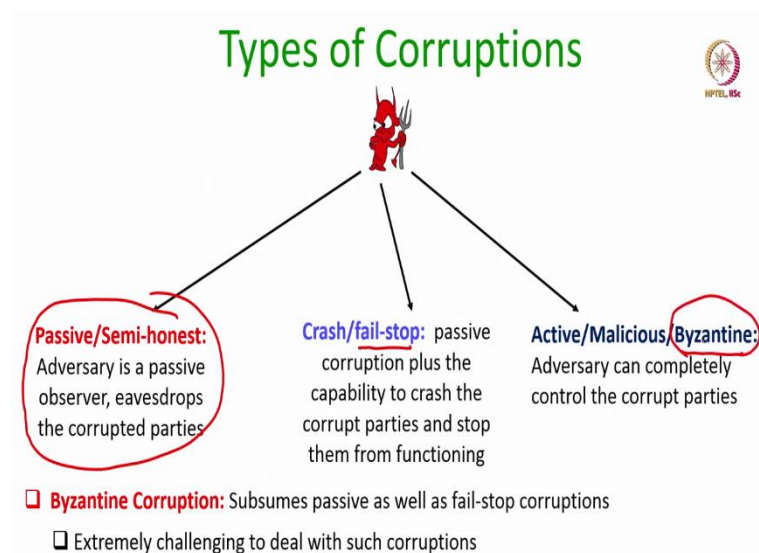
So, in some sense this latter class of protocols provides you everlasting security. Because even if some computational assumptions are broken, they no longer hold in future then still it will be ensured that a protocol which is secure against an unbounded adversary remains secure. The 5th dimension is the type of corruption. So, here we have three categories: passive corruption, crash corruption and active corruption.

Passive corruption means adversary only eavesdrops the state of the corrupt parties, but it cannot force the corrupt parties to behave arbitrarily during the protocol execution. That means, whatever instructions are assigned as per the protocol to a corrupt party the corrupt party will follow those instructions. Crash failure is a slightly powerful form of corruption and active corruption is the most powerful form of corruption.

So, we will discuss about these three types of corruptions very soon. And the 6th dimension is based on the corruption time; whether the adversary decides the set of corrupt parties at the beginning of the protocol itself such kind of adversaries are called as static corruptions. Or, we can have a more powerful adversary adaptive adversary which adaptively corrupts the parties as the protocol proceeds depending upon how much information adversary has already seen during the protocol execution.

So, in my existing NPTEL course title Secure Computation Part I which is available at this link you can find out more detailed discussion regarding the various dimensions in which we can study the secure MPC problem. And the focus of the part I of the course was completely on passive corruptions. Now this course is all about active and malicious corruptions which is the more powerful form of corruption.

(Refer Slide Time: 23:00)



So, let us go and see in more detail the difference between the types of the corruptions. So, as I said there are three types of corruptions the most the simplest form of corruption is the passive or the semi honest corruption. Where the adversary is an eavesdropper and it can only eavesdrop the state of the corrupt parties, but it cannot influence or it cannot cause the corrupt parties to deviate from the protocol instructions.

A slightly more powerful form of corruption is the crash corruption or fail stop corruption, where the adversary has the capability to eavesdrop plus it can cause the corrupt parties and make and stop them from functioning at any time during the protocol execution. And

once an adversary corrupts the parties and stops them from functioning then from that point onward for the rest of the protocol execution the party will be completely inactive it cannot suddenly become active after some time.

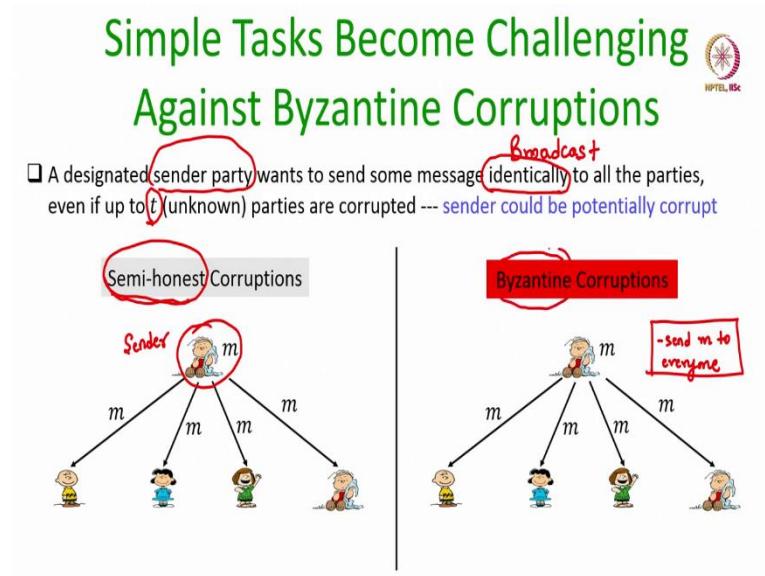
So, that is a crash adversary, but I stress that if that party is alive during the protocol execution, it will behave like as if it is under the control of a passive adversary; that means, it cannot deviate from the protocol instructions. And the most powerful form of the corruption is the active corruption which is often called as malicious corruption or byzantine corruption.

And here the adversary is very powerful it can completely control the state of the corrupt parties. That means, it can not only just eavesdrop the state of the corrupt parties it can cause it them to crash as well at any time during the protocol execution. And it can also cause the corrupt parties to start sending arbitrary values completely deviating from the protocol instructions.

So, for instance if during the protocol execution a party is supposed to send some value x to everyone then a byzantine corrupt party may instead send x' to one set of parties x'' to another set of parties and it may not send anything to another set of parties right. So, that models completely arbitrary behavior on part of the adversary and that is why it is extremely challenging to deal with such corruptions.

So, in part I of the course titled secure computation part I we only considered passive or semi honest corruptions. But in this course, we will consider MPC protocols tolerating the most powerful type of corruptions namely the byzantine corruptions.

(Refer Slide Time: 25:55).



So, to give you a feel about what challenges are involved while designing protocols against byzantine corruptions let us take a very simple task namely that of broadcast. And let us see that how the simple task becomes so challenging as soon as I assume that during the protocol execution, I might have byzantine corruptions. So, what is a broadcast problem? Here you have a designated sender party. So, everyone will know who the sender party is and the sender party has some message; it could be a bit or it could be a bit string we do not care.

And the goal is to ensure that the sender party sends its message identically to all the parties, even if up to t parties in the system are corrupted. And when I say t parties in the system are corrupted, I mean to say that the sender could be one of those t parties. Sender is one of the n parties, it has some message, and everyone is expecting sender to broadcast its message send the same copy of its message to everyone. And everyone also knows that in the system there can be at most t corruption where t is some parameter.

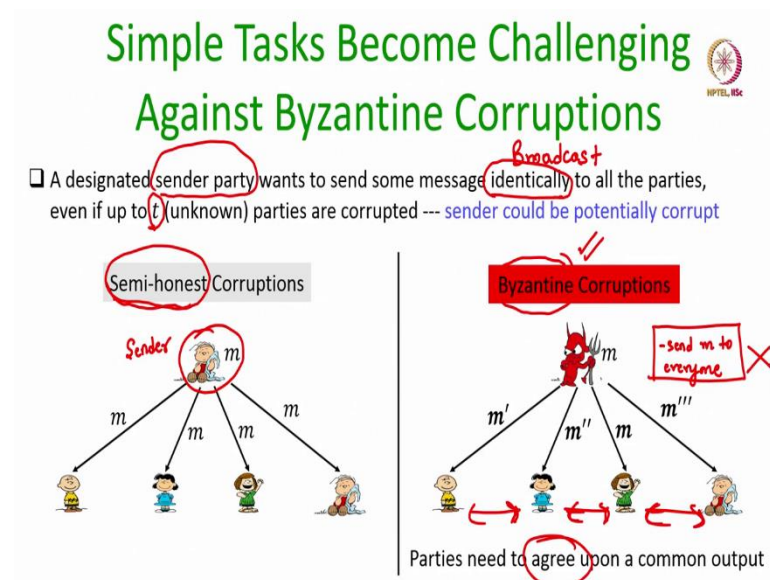
And it could be the case that among those t parties which can be corrupted sender is also one of the entities. So, if we consider this problem in the semi honest corruption setting where we assume that the t corrupt parties could be only passively corrupted then this problem is very easy to solve. So, imagine this is the sender party to broadcast its message what it must do is it has to just send the message m to all the parties; that is all.

So, I assume that there is a communication channel between this party and every other party. So, it can just send a copy of m to everyone else. And since I am assuming that I am in the semi honest corruption model even if this party is under the control of this adversary even if the; even if the sender party is under the control of the adversary that need not be always the case I stress.

But even if it is under the control of an adversary, since we are assuming a semi honest corruption model it will send the same message to everyone; because the protocol instruction is send your message to everyone that is a one line code that is a broadcast protocol in the semi honest setting. So, the semi honest sender or potentially semi honest sender cannot deviate from that protocol instruction, it will send the same copy of its message to everyone else.

But now, if I consider the same protocol the same one-line code send your message to everyone send m to everyone suppose this is the protocol code. And now if we execute this protocol in the byzantine corruption model where the sender could be potentially corrupt.

(Refer Slide Time: 29:22)

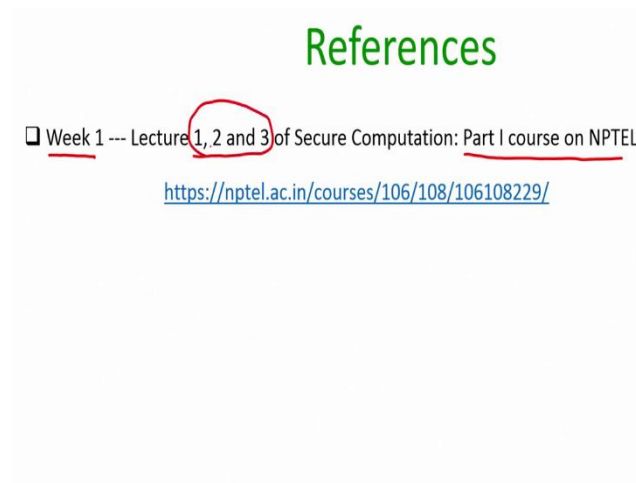


Then it can do the following to different parties it may send different versions of its message, because it need not follow the protocol instructions, because we are assuming now a byzantine corruption model. As a result of this the simple one-line code of sending the senders message to everyone will simply fail because now different parties will have

different version versions of the senders message. So, now, we need to do something more on top of sender simply sending its message to everyone namely, we would require the parties to interact right and agree upon version of the message sent by the sender.

And the amount of interaction which is required to come to a conclusion may vary from protocol to protocol. So, what I have demonstrated here is that a very simple task like sending a message identically to everyone becomes so challenging, so complicated, as soon as we consider byzantine corruptions. Because byzantine adversaries can completely force the corrupt parties to behave in a completely arbitrary fashion.

(Refer Slide Time: 30:38)



So, with that I conclude this lecture. So, let me summarize what we have discussed in today's lecture. In today's lecture we have introduced the problem of secure multi party computation. We have seen various applications various real world problems which can be abstracted by this problem of secure multi party computation. And we have seen various dimensions in which we can study this problem this problem is often called as the Holy Grail problem of secure distributed computing.

We have seen dimensions depend like the various ways to abstract the underlying function, the way we can model the corruption capacity, the nature of the adversary, the timing of the adversary, the computing power of the adversary and so on right. So, the reference for today's lecture is basically the week 1 of the part I of the course and in that week 1 you

can refer to the lectures 1, 2 and 3 to know more about the various dimensions in which we can study the MPC problem.

Thank you.