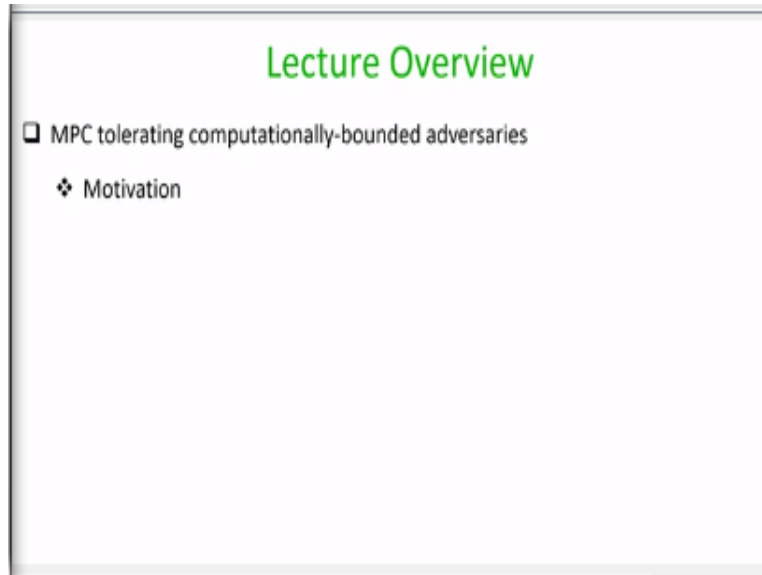


Secure Computation: Part I
Prof. Ashish Choudhury
Department of Computer Science Engineering
Indian Institute of Technology-Bengaluru

Lecture-35
Towards Cryptographically-secure MPC

(Refer Slide Time: 00:32)




Hello everyone, welcome to this lecture. So, till now our focus was on perfectly secure multi party computation or tolerating semi honest corruptions, where we assume that the corrupt parties are computationally unbounded. We will shift our focus to another line of research which is done in multi party computation, namely multi party computation protocols tolerating computationally bounded adversaries. So, in this lecture we will first see the motivation for doing that.

(Refer Slide Time: 01:09)

Perfectly-Secure MPC Against Passive Corruptions : Summary

- ❑ Setup required: pair-wise private channels
- ❑ Optimal resilience:
 - ❖ $t < n/2$ - threshold
 - ❖ $Q^{(2)}$ - non-threshold
- ❑ Circuits over a (field)



Can these "restrictions" be removed assuming a PPT adversary?

→ probabilistic polynomial time

So, before we proceed, let us quickly summarize what exactly we have learned regarding perfectly secure multi party computation against passive corruptions. As part of a setup we require pair wise private channels. That means we require that between every pair of parties there should be a mechanism to do secure communication. And the mechanism should be such that even if there is a computationally unbounded adversary, it cannot figure out regarding the communication which is happening between any pair of honest participants.

In terms of optimal resilience, we have proved that for general multi party computation if we are in the threshold setting then were adversary can corrupt up to t parties. Then the optimal resilience t should be strictly less than $\frac{n}{2}$, whereas if we model the corruption via non threshold adversary in terms of an adversary structure then the necessary condition is that the set of parties should satisfy the Q^2 condition.

Namely the union of any 2 potential corrupt subsets from the adversary structure should not cover the entire set of parties. And this Q^2 condition is a generalization of your $t < \frac{n}{2}$ condition. We have also seen that if we want to design protocols against computationally unbounded adversaries, then for the case of threshold setting we require a finite field because we use polynomials of degree t and those polynomials have to be over a field.

But for the non threshold setting we can design circuits even over a ring. We can design protocols, evaluating circuits even over a ring. So, now a natural question is can these restrictions be removed assuming a PPT adversary? By PPT adversary we mean probabilistic polynomial time adversary. Namely an adversary whose computing power, who's computing resources is upper bounded by some polynomial function in the underlying security parameter.

So the question is that, can we get rid of this setup requirement namely the pair wise private channels and can we improve the resilience bound assuming that we want to now tolerate an adversary who is computationally bounded?

(Refer Slide Time: 04:08)

The slide is titled "Cryptographically-Secure MPC Against Passive Corruptions" in green text. Below the title, there is a list of requirements:

- Setup required: shared cryptographic keys among the parties
- Optimal resilience:
 - $t < n$ $t = n - 1$
 - $Q^{(1)}$
- Circuits over a ring

In the bottom right corner, there is a yellow box containing the question: "Can these 'restrictions' be removed assuming a PPT adversary?". To the left of this box is a cartoon yellow emoji with a hand on its chin, looking thoughtful.

And interestingly, the answer is yes. So, let us focus on cryptographically secure MPC protocols. By cryptographically secure, I mean an adversary who can corrupt parties and where the corrupt parties running time resources is upper bounded by some polynomial function. The party is not allowed to do unbounded computing, unbounded exponential amount of computation and so on. So, if you focus on cryptographically secure MPC against passive corruptions, server focus will still be on passive corruptions.

We are just weakening computing resources, the computing power of the adversary that is all. But the nature of corruption remains the same, namely passive corruption, where the corrupt parties

will honestly follow the protocol instructions, but will try to infer additional information by analyzing the protocol transcripts which they are not supposed to do.

Then in terms of setup requirement, we do not require the presence of pair wise private channels among parties. If we assume that there is a mechanism by which the parties can establish cryptographic keys, keys for encryption, decryption keys for authentication and so on, then we do not require the presence of pair wise dedicated channels or mechanisms for every pair of parties to do private communication.

Because using these cryptographic keys they can encrypt messages and communicate publicly. In terms of optimal resilience, we will see that we can design MPC protocols against computationally bounded adversaries, where in the threshold setting, we can tolerate all but one corruption. Namely even if $t = n - 1$ that means if you have say $n = 100$ participants and even if 99 participants are under the control of the adversary still we can achieve security if we assume computationally bounded adversary.

And of course this is the best that you can hope for because if all the n parties are corrupt and the whole purpose of designing MPC protocol is lost, so that is the best you can hope for. Whereas, for the non threshold setting, we can design protocols tolerating an adversary structure where the set of parties satisfy only the Q^1 condition it need not satisfy the Q^2 condition.


And we can design protocols both for the threshold setting as well as in the non threshold setting for evaluating circuits which are designed over a ring. That means we can take computations which are performed over a ring and those computations can be performed securely using the cryptographically secure MPC protocols.

(Refer Slide Time: 07:12)

Cryptographically-Secure MPC Against Passive Corruptions

- ❑ Setup required: shared cryptographic keys among the parties
- ❑ Optimal resilience:
 - ❖ $t < n$ $t = n - 1$
 - ❖ $g^{(1)}$
- ❑ Circuits over a ring

$n = 2$ $t = 1$
 client Server

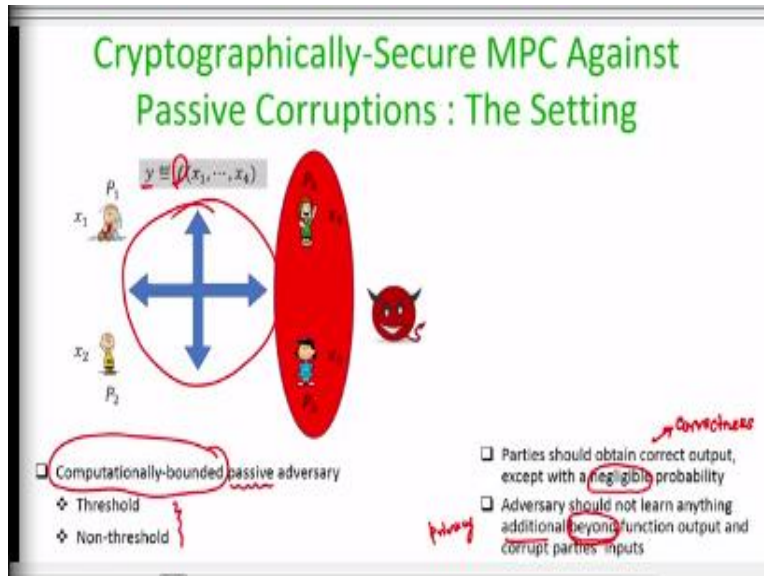


Secure 2PC: A very special and practical case of cryptographically-secure MPC

And more importantly, since we are considering the resilience $t < n$ if we consider $n = 2$ and $t = 1$, then we get a special case of secure 2PC, secure 2 party computation which is a very practical case of cryptographically secure MPC protocol. Remember, in the information theoretic world where we want to achieve perfect security, we can never design a protocol for 2 participants tolerating one semi honest corruption; we have proved the impossibility of computing security and function.

But we will now see that we can design protocols even to securely compute AND function for the 2 party case tolerating one semi honest corruption. So, the secure 2PC is a very practical case of multi party computation because there are several real world problems which fall under this special case of secure 2 party computations. Namely all the problems where you have a scenario where you have a client and you have a server and they want to interact and perform some kind of secure computation, maintaining the privacy and security of their respective data automatically falls under this special case of secure 2PC. So, that is why there are plenty of motivations for studying cryptographically secure MPC, we can get better resilience. We do not require a strong setup among the parties if they have mechanisms to encrypt and authenticate and that is sufficient. We can evaluate circuits, we can perform secure computation for functions which are done over a ring and so on.

(Refer Slide Time: 09:08)



So, now let us see understand the problem definition, the problem setting, we are given a set of n mutually distrusting parties. For demonstration I am taking n to be 4. And they are mutually distrusting, so that distrust in the system is again model by a centralized adversary who is computationally bounded. And it can corrupt a subset of parties in passive fashion or semi honest way.

By passive means it can listen entire communication or entire messages to at those parties have received that those parties have sent, their input, their local randomness and so on. And again like we have done for the case of perfect security, we can have 2 types of corruptions, we can have threshold corruption, where the distrust is modeled by an adversary who can control any t out of the n parties.

The exact identity of those t parties will not be known but the parameter t will be publicly available or we can consider a more general form of the adversary where the adversary is characterized by an adversary structure. And our goal is to design an MPC protocol according to which the parties should interact, exchanged messages and finally obtain the output of the function which is publicly known.

And a function is an NRE function. As I said that a subset of the parties can be corrupted by the adversary. So, assume we take a threshold adversary and say the adversary corrupts party 3 and 4,

then that will be the view of the adversary. The view of the adversary means the inputs and output of the corrupt parties, their local randomness and whatever messages they have sent and received in the protocol which is going to be a random variable.

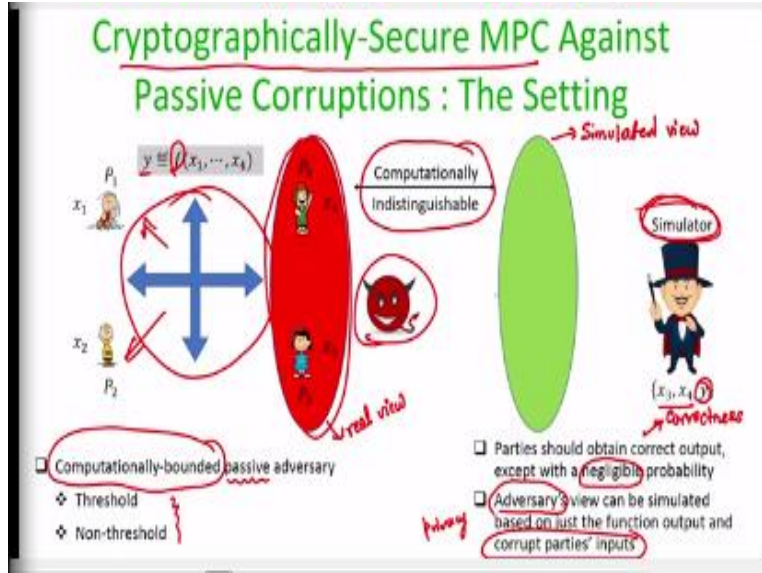
So, we require following 2 properties to be achieved by the MPC protocol. The first property is roughly the correctness property, where the requirement is that at the end of the protocol the parties should obtain the correct output except with some small error probability which we call as the negligible probability. So, what is a negligible probability? I refer you to the NPTEL course on foundations of cryptography where we have defined what we mean by a negligible function of the security parameter.

And why are we now allowing a small error probability? Why do we not demand that the parties should output the correct output or obtain the correct output without any error? Because we are going to now use cryptographic tools and in the cryptographic tools there might be a small error probability which overall translates to the error probability in the correctness of the MPC protocol.

This is unlike your perfectly secure MPC protocols where even the correctness was error free. There the requirement was that the honest parties should obtain the correct output, absolutely without any error. So, this is the correctness requirement. Then, we have the privacy requirement, where we demand that adversary should not learn anything additional other than what it can infer or what it can deduce from the corrupt parties inputs, their output.

Because whatever can be deduced or inferred from the corrupt parties' input and the function output that we can never prevent from getting leaked because that is the inherent nature of the function itself. We require that, apart from that nothing additional should be revealed about the honest parties input based on the interaction that the adversary or the corrupt parties have during the MPC protocol. But this is a very loose statement; we know how to make it more rigorous, more formal.

(Refer Slide Time: 13:19)



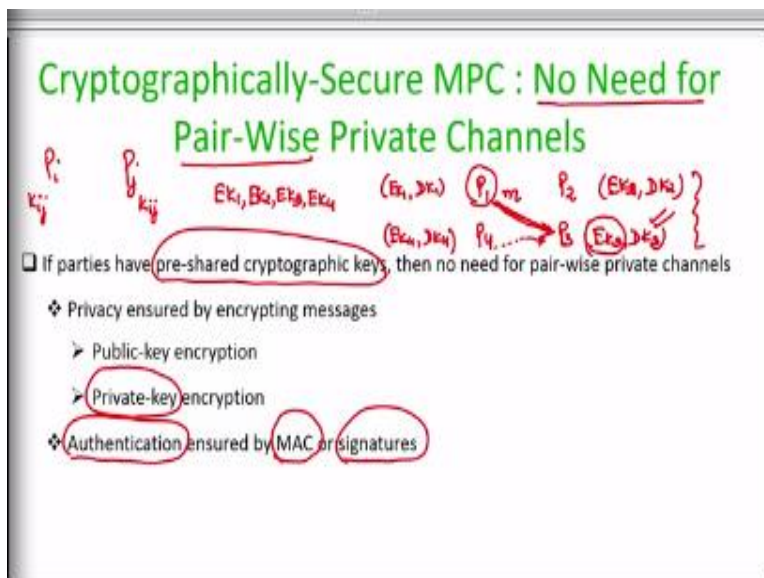
The privacy condition is basically formalized by saying that whatever view the adversary or the corrupt parties generate by participating in the MPC protocol, the adversary can recreate, regenerate or simulate the same view even *without* talking with honest parties. Namely, just based on the corrupt parties', inputs and the function output the adversary can simulate whatever conversation he could have with honest parties.

And more formally, we require the existence of a simulator and algorithm, a probabilistic algorithm, which when given the inputs of the corrupt parties and a function output, can reproduce a view without even talking to the honest parties and without even knowing the inputs of the honest parties such that the simulated view has the same probability distribution as the real view of the corrupt parties. So, in the case of perfect security, the requirement was that the simulated view should be identically distributed as the real view of the adversary.

But now since we are considering a cryptographically secure MPC protocol where the adversary is polynomial time bounded, the requirement will be that the simulated view should be computationally indistinguishable from the real view of the adversary. Again, what do we mean by computationally indistinguishable? That means that there exist no poly time adversary or algorithm which can distinguish a sample picked from the real view from a sample picked from the simulated view, both are almost identical.

So, if we can show the existence of such a simulator, then that basically establishes that whatever this adversary could obtain by interacting with the honest parties, whatever it could learn the messages, it could sit back at home and reproduce the same messages. Not the same messages, but the messages with have that same probability distribution just by sitting at home and without even talking to the honest parties. That is precisely as the simulation strategy.

(Refer Slide Time: 15:49)



So, as I said earlier that when we design cryptographically secure MPC protocol, we do not need any pair wise private channels because the parties have pre shared cryptographic keys. So, assume that we have 4 parties and suppose there is a mechanism by which a setup has been done, where each party has a pair of encryption key and decryption key. Suppose someone has done this setup and this will be like a onetime setup because the same cryptographic setup, and the same set of keys can be used for multiple instances of the MPC protocol.

Say the public keys are available in the public domain of the individual parties. Now if this set up is available, then whenever in the MPC protocol a party is supposed to send a message to another party - say P_1 is supposed to send some message to P_3 then for the case of perfectly secure MPC.

The assumption was that there is a mechanism by which P_1 can send that message in a perfectly secure way to P_3 . And that assumption was abstracted by saying that, ok, there is a dedicated

channel between P_1 and P_3 which no one can tamper. But now, since we are in the cryptographic setting, encryption and decryption keys have been set up.

We do not need such strong setup assumptions because whatever message m P_1 is supposed to encrypt, P_1 is supposed to send to P_3 as part of the MPC protocol, it can encrypt the share m or the message m using the public key of P_3 and send to P_3 . That ensures that even if that encrypted m is being transmitted publicly, the privacy or the security of the encryption scheme will ensure that nothing about m is revealed. And when P_3 receives the encrypted m since it has the decryption key; it can perform the decryption and recover back the MPC messages which P_1 is supposed to send to P_3 .

So, in this demonstration I have assumed a public key encryption mechanism but it could also be a private key encryption mechanism. So, instead of assuming that each party has its own public key, encryption key and decryption key. We can assume that every pair of party has a dedicated symmetric key an AES keys say for instance. And whatever messages are supposed to be communicated between those 2 parties, say between P_i and P_j , whatever messages are supposed to be communicated.

They all can be encrypted say using the key k_{ij} which is a symmetric key already established as part of a setup between P_i and P_j . So, this setup of pre shared cryptography keys is definitely a very, very weak setup. Weak setup in the sense to do such a setup we do not require expensive computation or expensive resources. But to ensure that parties have a mechanism to do perfectly secure communication, we need to spend more resources.

Moreover we also want to ensure that the MPC messages between the parties are exchanged in an authenticated way. That means whenever P_3 receives a message from P_1 , how does it believe that indeed it is coming from P_1 ? Why cannot it be the case that P_4 has injected those messages on the behalf of P_1 and forwarded to P_3 ? That is an authentication problem. So, again assuming that we have pre shared cryptographic keys, if you want to achieve authentication also, while communicating the MPC messages, we can use message authentication codes or signature schemes.

(Refer Slide Time: 20:38)

Shared Circuit-Evaluation in the Cryptographic Setting : The Various Approaches

$c_1 \boxplus c_2$
 $c_1 \boxplus c_2 = c_3 \leftarrow \text{Enc}_{ek}(m_1 + m_2)$

For simplicity, consider threshold corruption $\rightarrow t$ semi-honest parties ($t < n$)

Goal: All values during the computation should be "hidden" in such a way that collective view of any subset of t parties should not reveal the underlying values

$c_1 \leftarrow \text{Enc}_{ek}(m)$ $c_2 \leftarrow \text{Enc}_{ek}(m_2)$ $c_3 \leftarrow \text{Enc}_{ek}(m)$

\boxplus Secret-sharing approach

- ◆ Computation and communication-efficient
- ◆ $O(D_{in})$ rounds
- Suitable for low-bandwidth networks
- Not suitable for high-latency networks

\boxplus Garbled-circuit approach

- ◆ Heavy computation and communication
- ◆ $O(1)$ rounds // Constant
- Not suitable for low bandwidth networks
- Suitable for high-latency networks

\boxplus Threshold public-key encryption based approach

- ◆ Heavy public-key operations
- ◆ $O(D_{in})$ rounds for linearly homomorphic scheme
- ◆ $O(1)$ rounds for fully (FHE) homomorphic schemes

So, now how do we go about designing cryptographically secure MPC protocols? The approach will remain the same; we will try to design generic MPC protocols. And when we do generic MPC protocols we have to assume that your underlying function which the parties securely want to compute is abstracted by a circuit. So, that circuit could be over a ring, over a finite field, over any algebraic structure.

So, let us try to understand the approach of shared circuit evaluation in the context of cryptographic setting. And for explanation purpose, I consider a threshold adversary, namely I will assume that $t < n$, where the value of t is publicly known and any subset of t parties could get corrupt during the circuit evaluation. Now, recall the case of circuit evaluation and with perfect security.

There the goal was that during the circuit evaluation each value at the time of circuit evaluation right from input value all the way to the output value should be kind of hidden away. That if any subset of t parties try to combine their view, then they learned nothing about the exact values of the computation during the circuit evaluation. So, the principle remains the same. The circuit evaluation even in the cryptographically secure MPC protocol has to be done in such a way that each value during the computation should be hidden in such a way that if the set of subset of t parties which are corrupt, try to combine their view of their respective circuit evaluation then they learn nothing about the underlying values. Now, there are 3 approaches to maintain this principle

of shared circuit evaluation in the cryptographic setting. So, let us discuss those 3 approaches and we will touch upon these 3 approaches in the rest of the course.

So, the first approach is what you are already familiar with namely the secret sharing approach, where it will be ensured that each value during the shared circuit evaluation is secret shared with threshold t . The advantage of this approach is that in terms of computation, it is very efficient because to compute the shares, we do not need to perform heavy cryptographic operations and the share size will be very, very small.

So, in terms of communication also we do not have to do too much of communication. But in terms of interactions, it will require a number of rounds of interaction proportional to the multiplicative depth of your circuit D_M . And that is why the cryptographically secure MPC protocols based on secret sharing approach is suitable for low bandwidth networks, what do I mean by low bandwidth networks?

Networks where a parties cannot afford to communicate too much with each other, because the bandwidth among the parties is very less. So, if you are trying to design an MPC protocol for such a setting then definitely the secret sharing approach is recommended. Because when the parties interact, they do not send too large messages, they send very small messages and that will suit the low bandwidth networks.

But since the number of interactions is proportional to the multiplicative depth of the circuit, that means every time we encounter a new multiplication layer in the circuit, the parties have to interact. So, this secret sharing approach is not recommended if the latency in the network is high. That means if it takes enormous amount of time, for the messages to reach from one end to another end or you assume that you have 2 parties who are kind of geographically very, very isolated and network between those 2 parties as a very high latency.

Then definitely every time they would like to interact whenever they encounter a multiplication gate it MPC protocol will take enormous amount of time. So, you have a trade off here, if you want to save on bandwidth, go for a secret sharing approach but the price that you have to pay is

that you have to interact a lot. Then there is another nice beautiful approach for this shared circuit evaluation maintaining this philosophy that during the shared circuit evaluation, each value remains hidden in such a way.

So, by the way I have used the term hidden, I have not used a term shared because sharing the values is one way of ensuring that the value is hidden. But there could be other ways of hiding the value, namely encryption and so on. So, this Garbled circuit approach, it also ensures that each value during the circuit evaluation remains hidden in such a way that the view of the subset of t corrupt parties does not reveal any information about the exact values during the circuit evaluation.

However, compared to the secret sharing approach it has both pros as well as cons. So, the downside is, it requires heavy computation and communication to be performed while evaluating the circuit. Because it performs huge cryptographic operations and enormous amount of communication happens when the parties interact. But the most striking feature of this approach is that it requires constant number of rounds.

It does not matter what some multiplicative depth of your circuit, it could be million, it could be billion, it requires only a fixed round of interaction among the parties. That is the most striking feature of this Garbled circuit approach. And now you can immediately come to the following conclusion, that since it requires only a fixed set rounds of interaction, that means the parties need to interact only fixed number of times irrespective of how big, how deep is their circuit which they are trying to securely evaluate.

Then definitely this approach, this Garbled circuit approach is recommended for the high latency network. Namely, where the parties are geographically very, very isolated and it takes enormous amount of time for the messages to go from one end to another end. And definitely we will prefer a protocol for such a setting where the parties are not supposed to interact too much and definitely a Garbled circuit approach is the right approach.

But then you have to ensure that the bandwidth among the parties is sufficiently good. So, that is why you have tradeoff between the secret sharing approach and the Garbled circuit approach.

There is a third approach based on public key encryption, specifically the threshold public key encryption. So, what is a threshold public key encryption scheme? It is like the usual public key encryption scheme where the encryption key of a party will be available in the public domain, anyone can pick that public key and encrypt the message and send it to the party.

But now instead of having one decryption key, so encryption key will be available in the public domain. But we have say n parties in the system, the decryption key, let me call it s_k will not be available with any single entity, but rather each entity will have a piece of share for the secret key or the decryption key. Now the property of these pieces are the shares for the decryption keys such that if you are given an encryption of the message m .

Anyone can encrypt, because encryption key will be available in the public domain and say c is a ciphertext which needs to be decrypted. The threshold encryption scheme will ensure that no single party can decrypt c . It is only when $t + 1$ or more number of parties individually decrypt the ciphertext c using their respective shares of the decryption key and then they combine the partial decryptions they can recover back the plain text.

So, what I am saying here is, that if P_1 alone tries to decrypt c using his piece of information s_{k1} , it will obtain a share of the decryption call it c_1 . And like that c_i we have our share of the decryption, call it c_i and P_n will have a share of the decryption call it c_n . Now, if only $t + 1$ or more number of this partial decryptions come together or made available, then only we can recover back the message m .

But if only t or less number of partial decryptions are available, we cannot decrypt back the ciphertext c to get the message m , so that is what is the threshold public encryption scheme. So, again coming back to the circuit evaluation principle for the cryptographically secure MPC protocol, we can ensure that the parties evaluate the circuit jointly where each value during the computation remains encrypted as per a threshold encryption scheme.

And if we try to evaluate circuit using this approach, then since it is an instantiation of a public key encryption scheme, the parties have to perform heavy operations, heavy means

computationally heavy operations. And depending upon what kind of encryption scheme we are using, if we are using just ordinary plain homomorphic encryption scheme, then every time the parties encounter a layer of multiplication, the parties have to interact.

That is why the number of rounds of interaction will be proportional to the multiplicative depth of the circuit, if we are using a regular homomorphic encryption scheme. But interestingly, if we have a fully homomorphic encryption scheme or threshold fully homomorphic encryption scheme then parties do not need to interact for every multiplication layer, they have to interact only for a fixed number of rounds. Now what is a fully homomorphic encryption scheme?

It is a form of public key encryption scheme which allows you to perform operations on the ciphertext which gives you an equivalent result of the same operation or some operation over the underlying plaintext. So, what do I mean by that? So, imagine that you have encrypted a message m_1 , you means say one of the parties has encrypted the message m_1 and c_1 is the ciphertext. And this encryption is done using a fully homomorphic encryption scheme.

And say c_2 is an encryption using the same public key for another message say done by another party m_2 . Now when I say it is a fully homomorphic encryption scheme, what it means is the following. Say we want to add m_1 and m_2 without knowing m_1 and m_2 or without decrypting m_1 and m_2 . Then there is some operation which the parties can perform over the ciphertext or the encryptions of m_1 and m_2 , say denoted by this special symbol, this operation will be publicly known.

So, if the parties do perform this operation over the ciphertext c_1 and c_2 , then that will give them some ciphertext, they call it c_3 . And the c_3 will be an encryption of $m_1 + m_2$. Whereas say the parties want to multiply the plaintext m_1 and m_2 without decrypting c_1 and c_2 . Then there will be some operation publicly available which the parties can perform on the encryptions of m_1 and m_2 .

And that will produce a ciphertext say call it c_4 , which will be an encryption of $m_1 \cdot m_2$, so that is why it is called fully homomorphic encryption. That means without even knowing the underlying messages, underlying plain text, you can perform some operations on the ciphertext which will

give you the same result as if addition of the 2 plaintext have been encrypted or the product of 2 plaintext have been encrypted and so on and now if you know how to do addition and multiplication homomorphically.

You can now imagine that, if you take any computation, that computation can be split in terms of a sequence of plus and multiplication. And once all the inputs of the function are encrypted using a fully homomorphic encryption scheme. Then after that the parties do not need to interact at all, they can just locally keep on performing the homomorphic operation on the encrypted data. And then they will obtain the result of the computation in an encrypted fashion.

And if we are using a threshold fully homomorphic encryption scheme, once the result of the computation is available in an encrypted fashion, the parties can partially decrypt them. When they interact and $t + 1$ or more number of partial decryptions are made available, parties can obtain the function output. So, depending upon whether we are using homomorphic encryption scheme or a fully homomorphic encryption scheme, either we require constant number of rounds or number of rounds which is proportional to the multiplicative depth of the circuit.

But in general this approach is computationally expensive because we are performing heavy public key operations. So, with that I end this lecture, just to summarize we started discussing in today's lecture regarding cryptographically secure MPC. We discussed what the motivation for cryptographically secure MPC is and what the various approaches for performing the shared circuit evaluation are. Thank you.