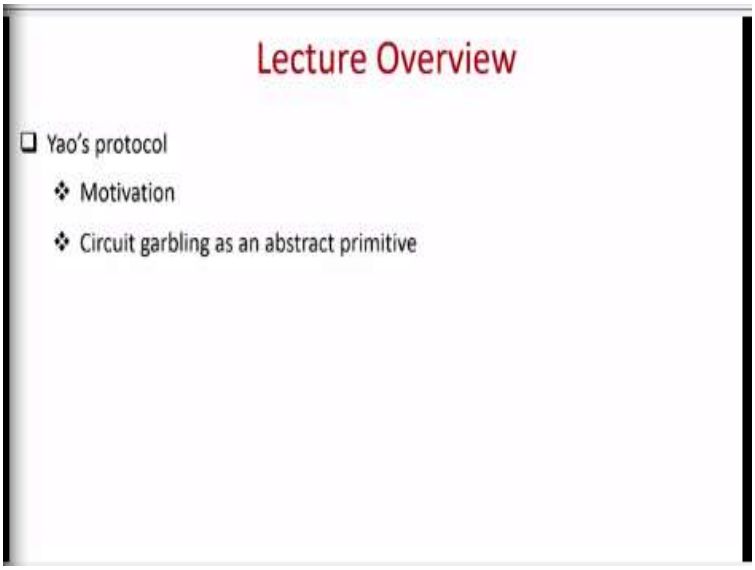
Secure Computation: Part I
Prof. Ashish Choudhury
Department of Computer Science Engineering
Indian Institute of Technology-Bengaluru

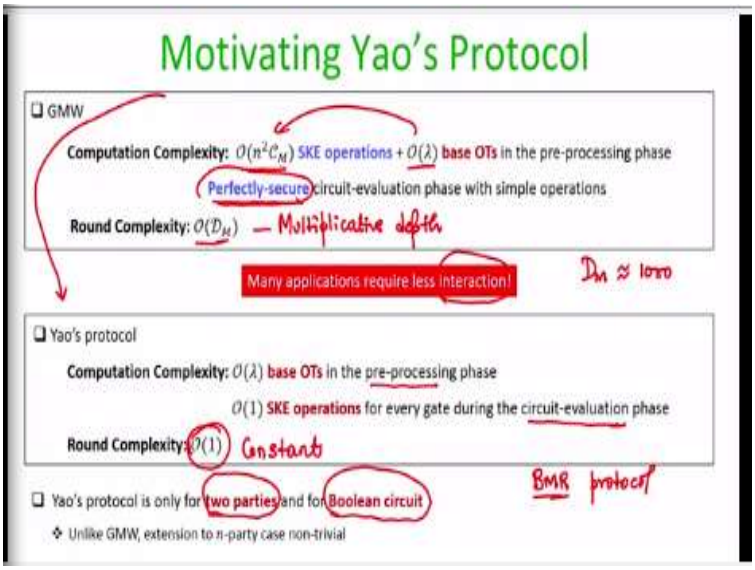
Lecture-48
Yao's Protocol for Secure 2PC

(Refer Slide Time: 00:31)



Hello everyone. Welcome to this lecture. So, we will now start discussing about Yao's protocol and we will first see the motivation for Yao's protocol. And we will see circuit garbling as introduced by Yao as an abstract cryptographic primitive.

(Refer Slide Time: 00:48)



So, let us first see the motivation for Yao's protocol. So, if we consider the GMW protocol which is a cryptographically secure MPC protocol. We can say the following, in terms of computation complexity we require a huge number of OTs to be performed. Namely, if there are a C_M number of multiplication gates in the circuit then the parties will be involved in n^2 times C_M number of OT instances to securely compute cross terms.

But now we know that we do not need to actually run those many number of OT instances using the help of OT extension protocols. In the pre processing phase we can run only order of lambda number of OT instances base OTs. And then using the IKNP extension protocol, we can get effect of so many OTs here. And the actual circuit evaluation phase protocol of GMW, once the Beaver's triplets or the multiplication triples are generated is perfectly secure.

Because there we evaluate the linear gates in a non interactive fashion, whereas for multiplication gates, just 2 values have to be publicly reconstructed. In terms of number of interactions, for every layer of multiplication gate during the circuit evaluation phase the parties have to interact. So, if D_m is the multiplicative depth of the circuit, then the amount of interaction which is required in the protocol is proportional to the multiplicative depth.

However, it turns out that there are several critical applications where we cannot afford to have a protocol which requires significant amount of interaction among the parties. So, for instance, if I

take a computation or a circuit or a function where say the multiplicative depth is of order 1000. Then the running the GMW protocol for evaluating such a function will require the parties to interact 1000 number of times at least.

But if interaction is a critical resource or if interacting multiple times is not possible, not feasible among the parties, then we cannot run the GMW protocol. And that is precisely is the motivation for the Yao's protocol. Because in Yao's protocol, the amount of interaction which is involved among the parties is a constant, it is completely independent of the size of the circuit, the number of gates in the circuit, the depth of the circuit and so on.

So, even if you have a huge circuit where say the multiplicative depth is 1000 or 10,000 or 20,000 we require only a fixed set amount of interaction among the parties. And that is the best part of the Yao's protocol. So, in networks or in settings where latency is a huge issue, where the round trip delay is too much, that means it takes enormous amount of time for one message to go to other end and so on and come back.

Then there we prefer to run the Yao's protocol because there the parties need to talk to each other only a fixed number of time. But the downside of Yao's protocol is that, it requires, it involves cryptography operations both in the pre-processing phase as well as in the circuit evaluation phase. So, this is unlike your GMW protocol where the circuit evaluation phase is perfectly secure.

But in the Yao's protocol, even the circuit evaluation phase is cryptographically secure; it will be secure only when the adversary is computationally bounded. And it requires you to perform cryptographic operations though both in the pre-processing phase as well as in the circuit evaluation phase. So, that is why in terms of computation and as well as in terms of communication, it is expensive compared to GMW protocol.

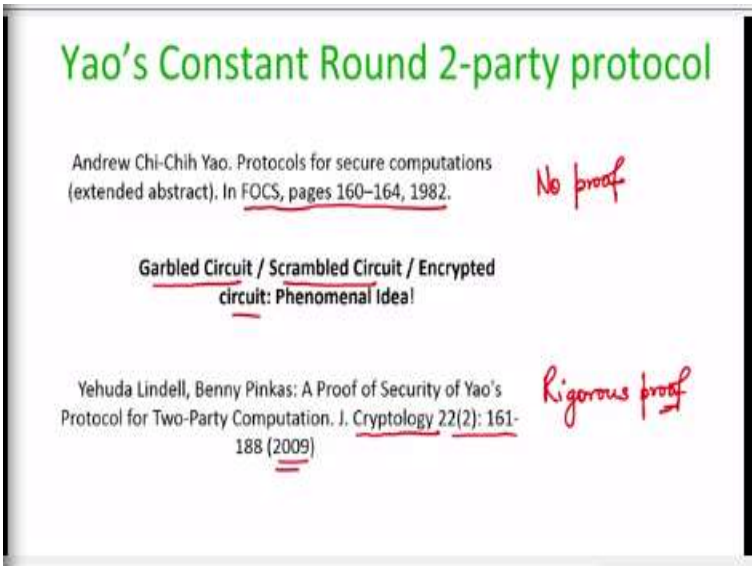
But you do a significant saving, not significant, tremendous saving in terms of number of rounds because it is a constant round protocols. Also unlike the GMW protocol couple of shortcomings which are associated with Yao's protocol are the following. So, first of all the original protocol

was proposed only for the 2 party setting. And it was only after a significant gap the protocol was extended for the n party case.

But unlike the GMW protocol, where the extension from the 2 party case to the n party case was very trivial, we do not have to do anything, it was simple natural generalization extension. The extension of the Yao's 2 party protocol to n party protocol is not that simple, it is not that trivial. And due to interest of time, I will not be going through the n party version of Yao's protocol. But if you are interested, you can refer to this protocol called as BMR protocol. BMR stands for Basal metabolic rate.

So, they came up with the n party version of Yao's protocol but it is not a simple generalization extension of the Yao's 2 party protocol. Other disadvantage or shortcoming of Yao's protocol is that, it is designed only for the Boolean circuit. And research is still going on to make the technique of Yao's protocol useful for any circuit, say arithmetic circuit over a ring or a field and so on. And again this was not the case for the GMW protocol. The GMW protocol works over any ring, it could be a Boolean ring, it could be an arbitrary ring, it could work over a field and so on.

(Refer Slide Time: 07:13)

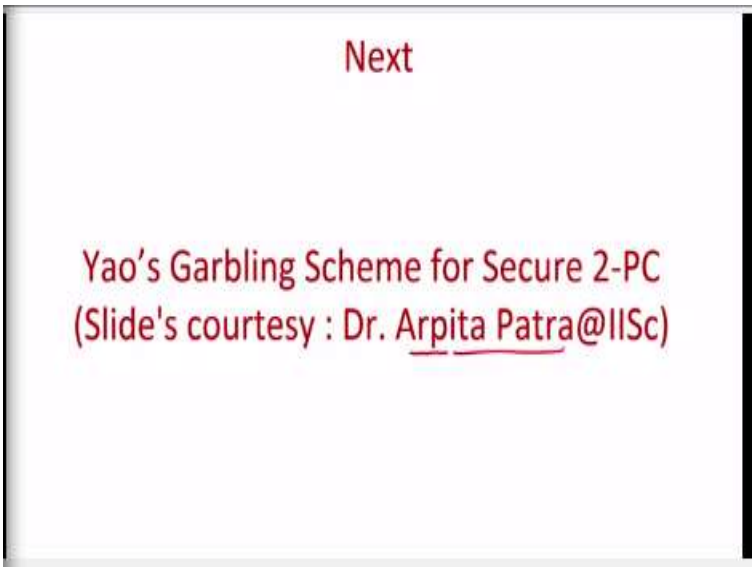


So, the Yao's protocol for 2 party case FOCS published in this seminal work, in fact this is the first paper which introduced the problem of secure computation. So Yao is credited with the invention of or coming up with the formulation of secure computation. And his protocol was based

on this phenomenal idea of garbled circuit which is also called sometimes a scrambled circuit, encrypted circuit and so on, it is a phenomenal idea.

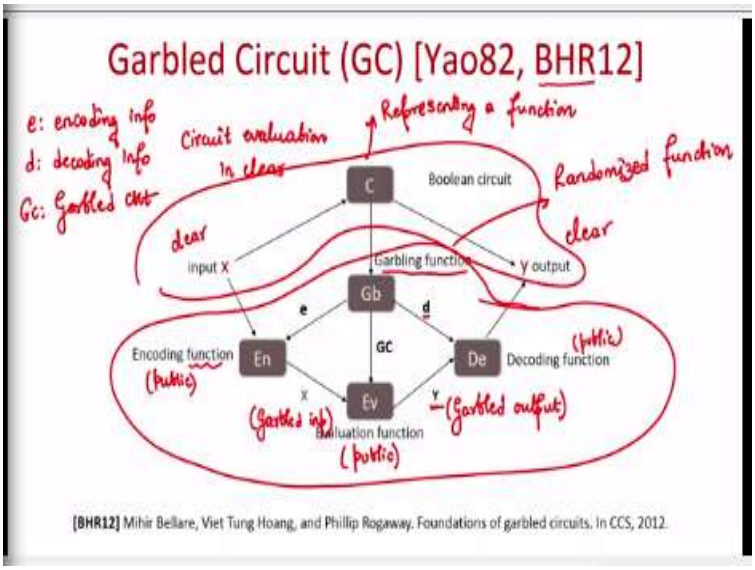
So, even though Yao proposed the problem of secure computation, he also came up with a solution for the 2 party cases, no proof was available, no proof was given in that paper. And it was only later in 2009 where a rigorous formal security proof was given for the Yao's protocol. So, I will not be going through the proof part of the Yao's protocol due to interest of time, but if you are interested to understand the proof you are strongly recommended to go through this paper.

(Refer Slide Time: 08:28)



So, now next what we are going to discuss is the following. We are going to discuss the Yao's original garbling scheme, garbling protocol; it is optimizations and so on. And most of the slides for the Yao's garbling scheme for secure 2-PC, I will be taking from my colleague, who is prepared these wonderful slides.

(Refer Slide Time: 08:53)



So, the notion of garbled circuit or GC was introduced by Yao, but his garbling technique was very specific and later on this paper by Bellare et al., Abstracted garbling as an independent cryptographic abstract primitive. And there could be several instantiations of this abstract garbling primitive. We can imagine that Yao's original garbling technique is one of those instantiations.

But the advantage of considering or treating the garbling technique as an abstract primitive is that later tomorrow one can come up with various other instantiations of the same notion of garbling. And then one can compare, analyze which instantiation is better, what properties are good? In one instantiation what properties are not good in other instantiation and so on?

So, let us first try to understand abstract primitive of garbled circuit. What exactly is garbling and garbled circuit? So, imagine you are given a Boolean circuit representing a function and it is a well known fact that you take any function, any computation it can be expressed as a Boolean circuit. In fact we have the so called universal gates like NAND gate, NOR gates where in terms of only those gates you can express any given computation.

So, if we consider a circuit evaluation in clear, then how exactly you will evaluate the circuit? So, if you are given a clear input X and a circuit, then once the input X is given the circuit given. You can evaluate each and every gate one by one by one and then you can obtain the clear output. That

is what is the clear evaluation, circuit evaluation is clear, but this does not preserve the privacy of X, we do not want to disturb.

Now, what garbled circuit do is the following. So, it is when I say a garbled circuit, by that I mean a garbling function which will be parameterized by a security parameter lambda. And that garbling function will be a randomized function, so it will not be the case that every time you run the garbling function, you obtain the same outputs, it is a randomized function. It is like equivalent to your key generation algorithm for encryption scheme where if you run the key generation algorithm, it outputs you a random key.

In the same way a garbling function an abstract garbling function is a randomized function parameterized by a security parameter which you run, it gives you 3 piece of information. e is called as the encoding information, d is called as the decoding information and GC is called as the garbled circuit. Now using this 3 piece of information encoding information, garbled circuit and decoding information, we can actually evaluate the circuit in a different way.

Now by applying the encoding information on the clear input X, using an encoding function which is publicly known, we can convert the clear input X into what we call as a garbled input, a noisy version of the input. Or you can imagine an encrypted input but it is not encrypted in strict sense. So, that clear input X is converted into some other representation, that some other representation is called as the garbled input.

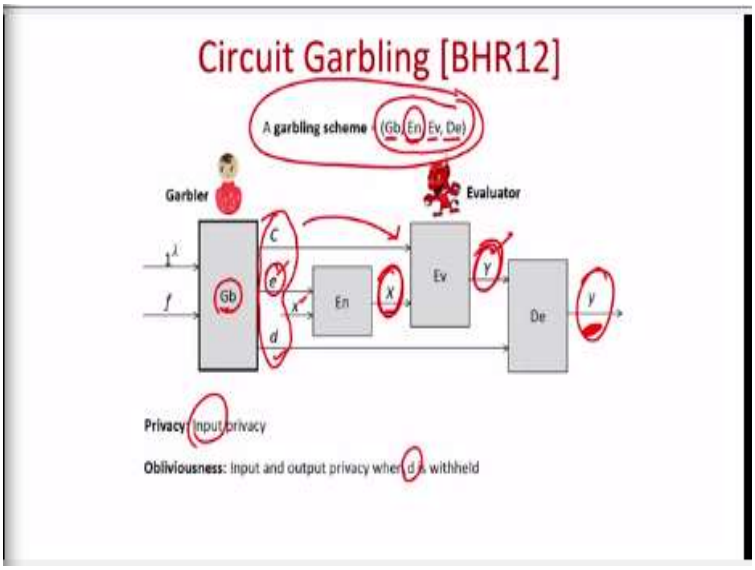
And now, given the garbled circuit and the garbled input, we can apply some publicly known evaluation function and get a garbled output. Now to convert this garbled output into clear output, we can apply some decoding function again which will be publicly known using the so called decoding information. So, now what you can imagine is that you have 2 ways of evaluating the circuit or computing the function output.

If you are given the clear input X, then the part that you can follow is this, evaluate each and every gate one by one. But if you do not want to reveal X to anyone in clear, what Yao suggested is the following. You follow the other path here, namely you convert your input X into a Garbled input,

a noisy version of the input and give a garbled version of circuit to anyone who wants to evaluate the circuit.

And that person who is going to evaluate the circuit, he will not be evaluating the actual circuit he will be evaluating a garbled version of the circuit on a garbled input. And he will obtain an output which is completely useless for him. Because until and unless I do not give him the decoding information, he cannot convert that garbled output into the clear output. That is the other way of going from X to Y.

(Refer Slide Time: 15:06)



So, when I say a garbling scheme, it is a collection of the Garbling function Gb, which produces the Garbled circuit, encoding information, decoding information along with the encoding function, evaluation function and a decoding function. And now this is an abstract collection of algorithms, so it is a collection of abstract algorithms, you can come up with any instantiation of Gb, En, Ev and De, what it has to satisfy certain properties?

The way these 4 components interact with each other is as follows. So, as I said the function Gb is a randomized function. It will take the function f the Boolean function, the representation of the Boolean circuit and it will output the garbled circuit c, the encoding information e and the decoding information d. Now any entity who is provided the encoding information and the clear input can apply the encoding algorithm or the encoding function here.

And obtain the encoded input or the garbled input which when used along with the garbled circuit in the evaluation function will produce a garbled output. And my decoding function should be such that that garbled output along with the decoding information should give me the output y. Forget about this (()) (16:48) and this y should we same as a f (X). So, that is why you cannot come up with any arbitrary Gb, En, Ev, De function.

They have to be 4 functions satisfying the requirements that I have stated in this pictorial flowchart, the way these components get involved. So, now how exactly we can use an abstract garbling scheme to do secure 2 party computation? Well, one of the parties, any of the 2 parties can be assigned as the role of Garbler. That Garbler can run the Gb function, the details of the Gb functions is publicly known.

So, it can run the Gb function by taking the function as the input and it can come up with the garbled circuit, the encoding information and the decoding information. This is like your secure communication where one of the 2 parties either the sender or the receiver runs the key generation algorithm. So, I am saying in the context of secure 2-PC, let one of the parties, we can fix it as part of the protocol, whether it is the first party or the second party.

Any one of those 2 parties can be assigned as the role of garbler who is going to run the garbling function and obtain the garbled circuit, encoding information and decoding information. And there will be another party who will play the role of evaluator and he will be provided the garbled input and the garbled circuit. And now he has to evaluate the garbled circuit on the garbled input to get the garbled output.

And if we do now provide him the decoding information, it can give the clear output back or it can just send back the garbled output to the garbler and then the garbler can use the decoding information to get back the clear output of it. So, now you can see that how it helps to get secure 2 party computations done. If we have a corrupt evaluator, will he learn anything about the input? The answer is no. And that is ensured by the security of your garbling scheme.

Even though he is provided the encoded information, he do not know the encoding piece of information, namely the piece is e. He knows only the encoded input, he does not know the encoding information e and hence it cannot figure out what exactly is the input over which he is evaluating the circuit. And in fact, if we withheld the decoding information, not only the input, the output is also, remain private from the evaluator.

Because after evaluating the circuit, he only learns the garbled output, it cannot figure out whether that garbled output corresponds to y or y prime or y double prime and so on. So, that is the way we are going to use an abstract garbling scheme. In the next lecture, we will see a concrete instantiation of this garbling scheme. Thank you.