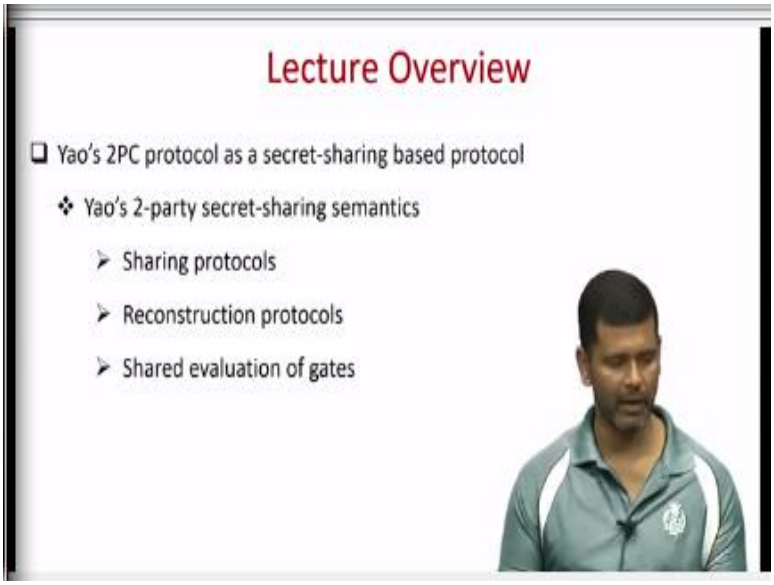
**Secure Computation: Part I**

**Prof. Ashish Choudhury**
**Department of Computer Science Engineering**
**Indian Institute of Technology-Bengaluru**

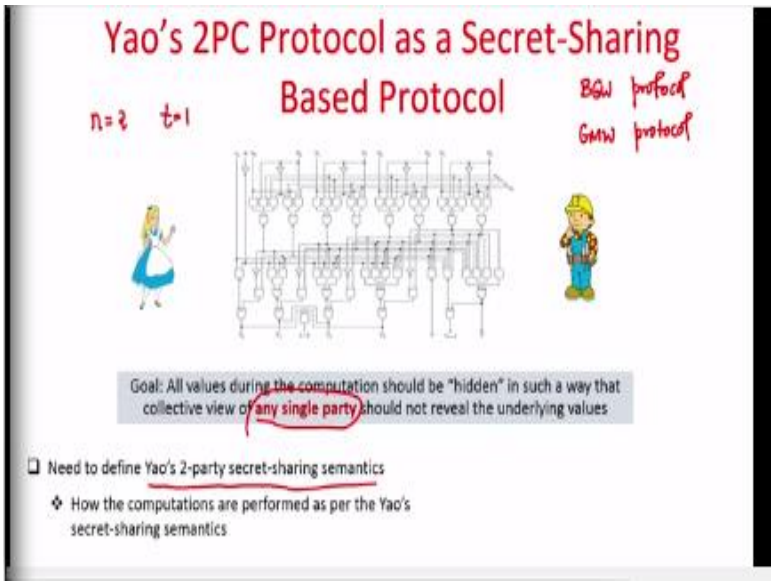
**Lecture-52**
**Interpreting Yao's Secure 2PC Protocol as a Secret Sharing Based Protocol**

**(Refer Slide Time: 00:32)**




Hello everyone, welcome to this lecture. So, in this lecture we will see how to interpret Yao's 2 party protocol as a secret sharing based protocol, where the parties will be evaluating the circuit based on the semantics of what we call as Yao's 2 party secret sharing. So, we will see the full syntax and semantics of Yao's 2 party secret sharing, what exactly is the sharing protocol, reconstruction protocol and how the shared circuit evaluation is done as per this sharing semantic.

**(Refer Slide Time: 01:04)**



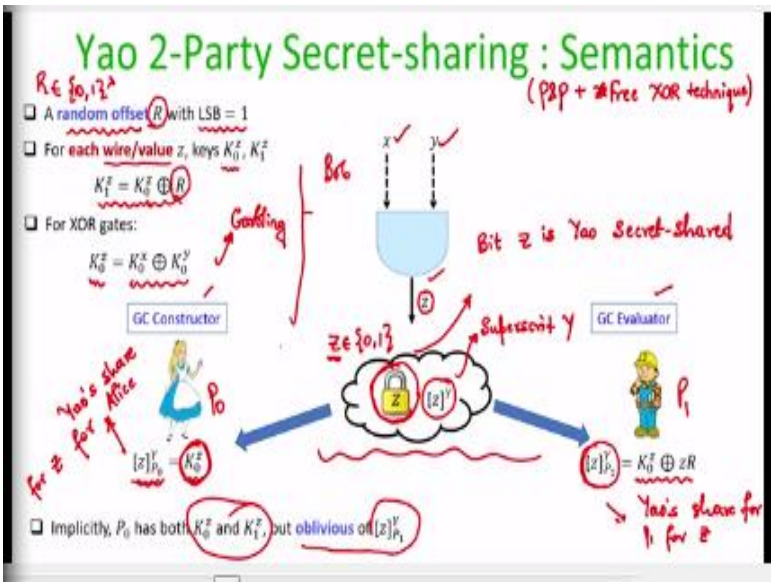

So, recall that in the Yao's 2 party protocol we have 2 parties - one of the parties performing the role of garbler who garbles the circuit and another party plays the role of evaluator. And what do we want to do here? We want to view this entire protocol executed between the circuit constructor and a circuit evaluator as some kind of secret sharing based protocol where the 2 parties are jointly evaluating the circuit in such a way that the collective view of any single party does not reveal anything about the underlying values and the input values. And that precisely is the invariant which we maintained till now, when we saw the secret sharing based MPC protocols. So, remember that we saw the BGW protocol as well as the GMW protocol.

And in both these protocols, the party contains the invariant that each value during the circuit evaluation remains secret shared with threshold $t$ in such a way that the collective view of any subset of $t$ Parties does not reveal anything about the underlying values. We want to do something similar even in the context of Yao's 2 party protocol. And in the context of Yao's 2 party protocol we are in a special case because there are only 2 parties and one of them is allowed to be corrupt.

So, what do we want to do now? We want to view visualize the entire computation done in the Yao's 2 party protocol as an instantiation or as a special case of secret sharing based protocol. So, for that we have to first identify what exactly is the sharing semantic of the so called Yao's 2 party secret sharing scheme and once that semantic is clear, we the have to understand how exactly the computations done by the 2 parties can be visualized as a special case of secret sharing based

protocol. Now, you might be wondering why exactly we want to do this. This will be later useful when we will see how the parties can perform mixed set computation.

**(Refer Slide Time: 03:25)**




So, now let us go through the Yao's 2 party secret sharing semantics. And remember, we have 2 parties here - one of the parties playing the role of constructor, another party playing the role of evaluator. So, they perform different roles unlike the BGW protocol and GMW protocol where the actions performed by all the parties are symmetric, symmetric in the sense they perform similar actions.

In Yao's protocol, the constructor plays a different role compared to the evaluator. So, imagine we have a gate like this and we will be considering Yao's 2 party secret sharing semantics based on the point and permute optimization plus the free-XOR technique. Of course, if you do not want to follow the point and permute optimization or the free-XOR then secret sharing semantics will be slightly different.

For our discussion, we will assume that the 2 parties are interested in evaluating the circuit by following the point and permute optimization coupled with the free-XOR technique. So, as per this sharing semantic there will be a global offset *R* which will be randomly chosen by the constructor. And it will be known only to the constructor and this will be fixed at the beginning of the protocol.

Remember, every time the protocol is executed, Yao's protocol is executed, the offset will be selected from the scratch, not the case that the same value offset is used once is used for all the instances of Yao's protocol between Alice and Bob. So, this random offset $R$ is a random bit string of length $\lambda$ bits, where $\lambda$ is your security parameter. And it is random except that its LSB is 1, this ensures that the permutation bits associated with the different keys corresponding to a wire or a value are complement of each other.

So, now imagine that there is a value $z$ and remember in the Yao's secret sharing protocol all the values are bits. Because Yao's protocol is used for evaluating a Boolean circuit, so the bit $z$ will be either 0 or 1. So, we now want to understand how exactly we can imagine or interpret the computations performed in the Yao's secret sharing protocol as a case of secret sharing scheme. And for doing that we have to identify what exactly the shares of Alice will be.

What will be the shares of Bob for $z$ as per the so called the Yao's 2 party secret sharing semantic? So, I will be interchangeably using the term wire and value. So, $z$ can be interpreted as the label of a wire or you can also interpret that since $z$ is a wire which can take either the value 0 or 1, so it can have 2 possible values. So, in Yao's 2 party secret sharing schemes corresponding to this wire or a value there will be 2 keys associated, both of them are of $\lambda$ bits.

One key corresponding to the value $z = 0$ another key corresponding to the value $z = 1$. And the 2 keys namely the keys corresponding to 0 and 1 they are related as per this relationship. Namely, if we perform the XOR of the key corresponding to $z = 1$ and $z = 0$, we get the offset $R$. Moreover, if this value $z$ or the wire $z$ is the outcome of XOR gate, then the 0 key corresponding to this value $z$ is set to be the XOR of the keys corresponding to the gate inputs.

And as per our sharing semantic of Yao's 2 party secret sharing, the shares of Alice and Bob will be as follows. So, the share for Alice will be the 0 key corresponding to $z$. And the share for Bob will be the key corresponding to the actual value of $z$. So, what I am saying here is $z$ can take the value either 0 or 1 irrespective of what exactly is the value of $z$. Even if $z = 0$, the share of Alice

will be the 0 key associated with 0, even if $z = 1$ the share associated with the share for Alice will be the 0 key associated with 0.

So, that means the share of Alice is completely independent of what exactly the value of $z$ is .Namely, her share will be always the 0 key associated with that value. The key or the share associated with Bob will be the actual key corresponding to the actual value of $z$. So if $z = 0$, then Bob's share will be the 0 key corresponding to 0. If $z = 1$, then Bob's share will be the key corresponding to $z = 1$.

And we will use this notation to represent that the bit $z$ is Yao's secret shared. That means if I denote the value $z$ inside a lock, that means neither Alice nor Bob know the full value of $z$, they only have a share of the bit $z$. And this notation means the superscript $Y$ denotes that the value is secret shared as per the Yao's semantic. I would also like to stress here is that even though Alice's share for the bit $z$ will be only the 0 key corresponding to that bit, implicitly she will be having both the keys, why so?

Because if you recall the Yao's secret sharing protocol, it is the GC constructor who picks the keys corresponding to all the wires in the circuit. So, if $x$ is a wire, $y$ is a wire, $z$ is a wire, Alice would have picked all the keys corresponding to $x, y$ and $z$. As per the sharing semantic, we will say that her share for the value will be only the 0 key, but implicitly she will be having both the keys corresponding to that bit.

But she will be oblivious of what exactly the key available with Bob is. Because if she is also aware of what key is available with Bob, then that means that she knows the value of $z$ as well, which may not be the case. And this precisely matches the way we do this secure computation as per the Yao's protocol. If you recall the Yao's protocol then Alice or the GC constructor does the garbling.

And later on, if I consider Bob then somehow we maintained the invariant that Bob for each wire in the circuit holds an appropriate key. Namely, if we consider a wire $w$ and if during the circuit evaluation in clear, wire $w$ takes the value 0, then Bob should have the key corresponding to $w =$

0. If during the circuit evaluation in clear $w$ takes the value 1, Bob should have the key corresponding to $w = 1$.

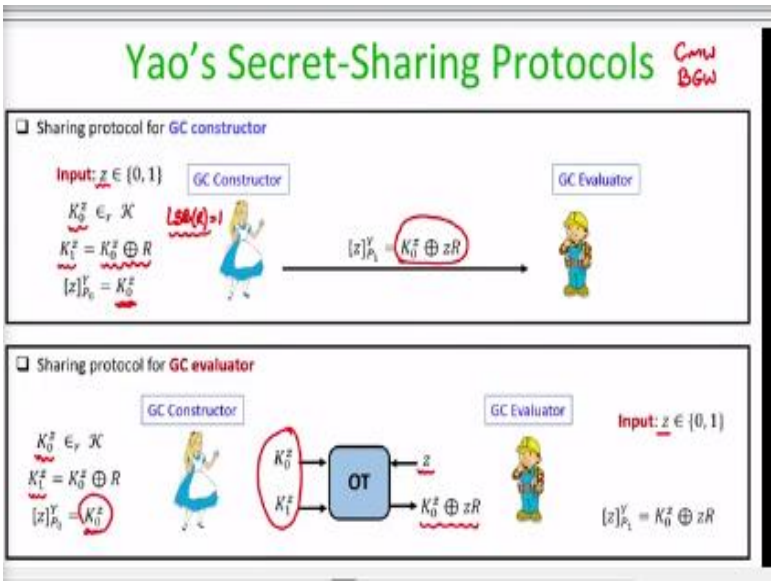
But Bob will be oblivious of whether the key it is holding is corresponding to 0 or 1 and that is precisely is the sharing semantic of Yao's 2 party secret sharing. Implicitly, Alice will have both the keys, Bob will have only one of the keys depending upon the exact value of $z$. And Alice will be oblivious of what exactly the key available with Bob is or what the share of Bob is And Bob will not be identifying anything about the value of the bit through his share.

Because, remember, the key associated with any bit or any wire is indistinguishable. But since we are using here free-XOR technique, both the keys associated with a wire are not completely random. But since the global offset $R$ is uniformly random and known only to Alice or the GC constructor, Bob just based on its share cannot identify whether the key available with Bob as a share is the key corresponding to the value 0 or the value 1.

So, $[z]^Y_{P_0}$ is the notation which we will use to denote Yao's share of $z$ for Alice or for $P_0$, so I call Alice as the party $P_0$ Bob as the party $P_1$. And $[z]^Y_{P_1}$ notation denotes Yao's share for $P_1$ for $z$. Also remember that it is not the case that for the same value of $z$, the shares of Alice and Bob will always be the same because it depends upon the keys associated with the value 0 which are picked by Alice, so they are picked randomly.

And also the random offset is also picked randomly and that is ensured that every time Alice and Bob execute the protocol. And if the value of $z$ remains the same they might have different shares with different probability depending upon what are the values of the 0 key, 1 key and what exactly is the value of the random offset?

**(Refer Slide Time: 14:33)**




So, that is sharing semantic of your secret sharing. Now we have to see how exactly the sharing is done by the dealers here. And here we can have 2 possible dealers - either the dealer could be the constructor itself or the dealer could be the evaluator. And depending upon who is going to play the role of dealer, the actual sharing protocol will be different. Again, this is unlike your GMW protocol or BGW protocol where the steps of the sharing protocol are same irrespective of who is going to play the role of the dealer.

So, for instance in the BGW protocol we have used Shamir secret sharing. The steps of Shamir secret sharing protocol remains the same irrespective of who exactly wants to share a value. In the same way in GMW protocol, we used additive secret sharing, and the steps are the same irrespective of who is playing the role of the dealer. But now in the Yao's secret sharing based protocols since the constructor and the evaluator are playing different roles the sharing semantics are also different here, have different interpretation.

The steps of the sharing protocol will be different if Alice wants to share a value and the steps of the sharing protocol will be different if Bob wants to share a value. So, let us first see the steps of the sharing protocol assuming that Alice or the GC constructor wants to share a bit. So, imagine that she has a private input $z$, which is either 0 or 1 which she wants to secret share. For doing that, she will pick the 0 key associated with the value $z$ uniformly at random from the key space.

And the key corresponding to $z = 1$ is picked as per the free-XOR optimization and remember that the LSB of $R$ is 1. And that automatically ensures that the permutation bits for the 0 key and the 1 key they are complement of each other. Remember, we are also following the point and permute optimization as well. And that requires that the permutation weights associated with $z = 0$ and $z = 1$ should be complement of each other and that is ensured by this restriction, namely the LSB of the global offset is 1.

And now Alice sets her share for $z$ to be the 0 key irrespective of what the actual value of $z$ is, even if $z = 0$ she will set her share to be the 0 key, even if $z = 1$ she will set the key corresponding to 0 as a share. And now she will give Bob his share for the bit $z$ and Bob's share will be the actual key corresponding to the value of $z$. So, if $z = 0$, Bob will get the 0 key, if $z = 1$ Bob will get the 1 key and remember the keys are indistinguishable.

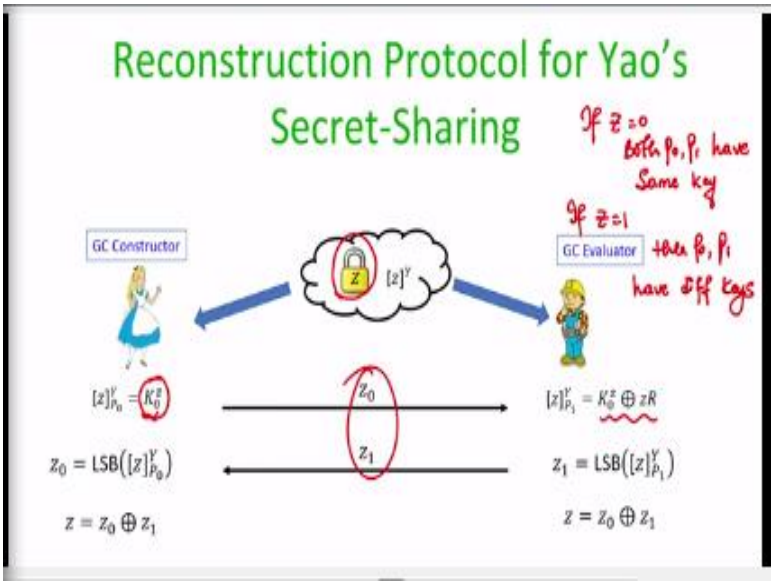
So, just based on the value of the key which Bob is receiving, Bob cannot figure out whether it is $z = 0$ which is secret shared or whether it is $z = 1$ which is secret shared. So, this is how Alice can secret share her bit. And if you see closely here the secret sharing protocol is nothing but the step in the Yao secret sharing protocol where for all the inputs in the circuit which are owned by Alice, Alice provides Bob the corresponding keys are the garbled input.

So, this step is nothing but providing the encoded $z$ to Bob and keeping the 0 key as share for Alice. Now, let us see the steps of secret sharing protocol, if Bob wants to secret share a value. Thus Bob wants to secret share a value where his private input is $z$ then Alice will do the following. Alice will select both the shares, the 0 share as well as the 1 share or equivalently the 0 key or the 1 key as per the steps of free-XOR technique.

And then she will set her share to be the 0 key associated with $z$. Now, we need to ensure that Bob should get the key corresponding to the actual value of $z$ and he should get only that key, he should not get the other key. And this can be done by executing an instance of oblivious transfer protocol where Alice plays the role of sender with her messages being the keys corresponding to the values of $z$. And Bob participates as a receiver with $z$ as his selection bit at depending upon the value of $z$, he gets the corresponding key.

And he learns only that key, he does not learn about other key. So, this will be the protocol which will be used by Bob to secret share his input $z$. Again if you see this protocol closely this protocol is nothing but the step in the Yao secret sharing protocol where for all the values which are owned by Bob as the circuit input. Or the function input Bob participates in the OT instances and obtains corresponding encoded a garbled input. That is precisely what we are doing here; we are adding an extra step here that we are asking Alice to retain the 0 key as her shares for Bob's input $z$.

**(Refer Slide Time: 20:39)**

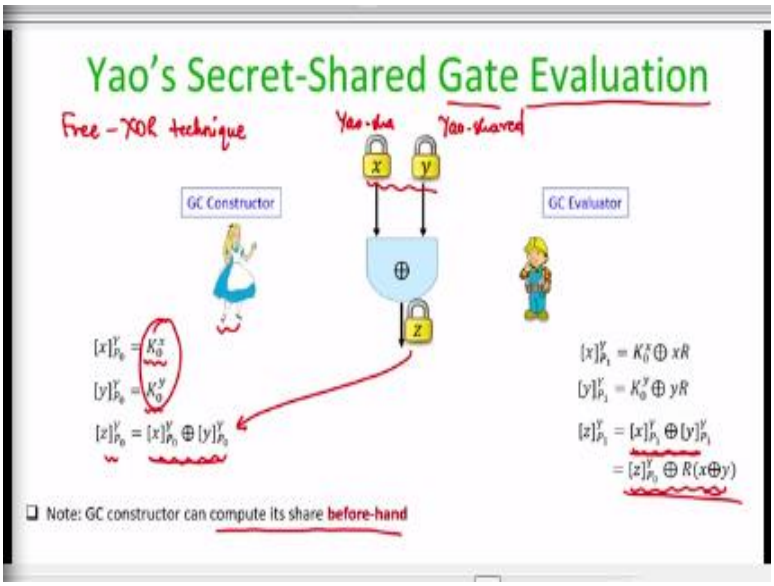



Now, let us see the reconstruction protocol. Assuming that we have a bit $z$ which has been secret shared as per the Yao secret sharing protocol. So, suppose there is a bit $z$ which has been secret shared, Alice share is the 0 key, Bob's share is the actual key and now they want to reconstruct a value $z$. To reconstruct a value $z$ what Alice and Bob can do the following. They can exchange their permutation bits of the shares held by them. So, remember this 0 key has a permutation bit and the key held by Bob also has a permutation bit. If the value $z$ = 0, if $z$ would have been 0 both $P_0, P_1$ have same key.

Namely, Alice's share is the 0 key and Bob's a share is also the 0 key and hence LSB will be 0 for both the keys. And hence if they XOR them they get back 0, if $z$ = 1 then $P_0, P_1$ have different keys, namely, Alice's share would have been the 0 key and Bob's share would have been the 1 key. And hence their permutation bits would be complement of each other, namely $z_0$ and $z_1$ will

be complement of each other and if we XOR them we get back the value 1, so that will be the reconstruction protocol.

**(Refer Slide Time: 22:22)**




So, we had seen the secret sharing protocol, we had seen the reconstruction protocol and now what is left is the shared gate evaluation protocols. Because these are the 3 ingredients which we require for any secret sharing based MPC protocol. If you want a secret sharing based MPC protocol for evaluating a circuit, we need mechanisms to share the inputs, we need a mechanism to reconstruct a shared input and we need a mechanism to evaluate gates over the shared inputs.

So, now we will see how we evaluate gates as per the Yao secret sharing interpretation or semantic. And the steps of the gate evaluation will be different depending upon the type of gate. If the gate which parties want to evaluate is a XOR gate and imagine that the inputs of this XOR gate are already Yao shared, namely $x$ is Yao shared and $y$ is also Yao shared. Now, we want to ensure that $z$ also becomes Yao shared as per the Yao's 2 party secret sharing semantics.

So, if $x$ is secret shared then the share for Alice will be the 0 key and similarly the share for $y$ will be the 0 key which are held by Alice. And then since we are using the free-XOR technique, remember that we are using the free-XOR technique. The free-XOR technique does not require the XOR gate to be garbled. So, the GC constructor or Alice in this case can simply set her share for $z$ to be the XOR of the shares of $x$ and $y$.

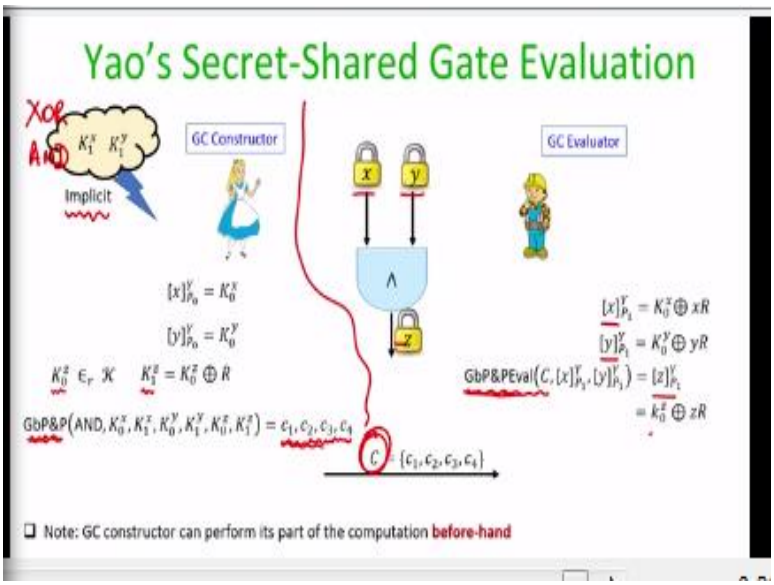
Namely, she will set the share for $z$ to be $K_0^x \oplus K_0^y$ held by Alice as per the Yao's sharing semantic. And in the same way, Bob will do the same, Bob will set his share for $z$ to be the XOR of the shares for the $x$ value and $y$ value held by Bob. And now if you see closely here, by doing so Bob is actually computing this value. Namely, it is $[z]_{P_0}^Y \oplus R(x \oplus y)$

And that is precisely what we want to ensure as per year 2 party secret sharing semantic. As per the Yao's secret sharing 2 party semantic, Alice should have the 0 key associated with $z$ and that is what she has. Because indeed the XOR of the $K_0^x$ and XOR of $K_0^y$ is $K_0^z$ and we want Bob to hold the key corresponding to $z$ depending upon what the value of $z$ is. If $z$ is 0, then Bob should hold the 0 key associated with $z$, if $z = 1$ then Bob should hold the key corresponding to $z = 1$.

And the value of $z$ depends upon the actual values of $x$ and $y$ and that is what we are guaranteeing here. So, evaluating the XOR gate is very simple here. And remember that to evaluate this XOR gate, Alice or the GC constructor can compute it is share beforehand. What does that mean? So even if the values of $x$ and $y$ are not determined, when the GC constructor is garbling the circuit and the inputs of the circuit are not yet ready, Alice can do the following.

Alice, since she knows the 0 key associated with $x$, the 0 key associated with $y$, she can simply set her share for the value $z$ irrespective of what is the value of $z$ is going to be at the runtime during the execution of the protocol to be the XOR of the keys associated with the $x$ wire and $y$ wire. And this task can be done by Alice beforehand itself. But Bob or the evaluator cannot compute its share for the value $z$ beforehand. Because Bob's share for the value $z$ will be the shares of $x$ and $y$ held by Bob which are determined at the runtime when the actual values of $x$ and $y$ are fixed.

**(Refer Slide Time: 26:55)**




Now, let us see how an AND gate can be evaluated. And why we are focusing on only XOR gate and AND gate? Because we can always express any Boolean circuit or any function in terms of a Boolean circuit consisting of only these 2 types of gates, namely the XOR gates and the AND gates. So, we had already seen the process of evaluating the XOR gate, the AND gate can be evaluated as follows.

So, again Alice has her shares for $x$ and $y$ which are the 0 keys associated with $x$ and $y$ respectively. And Bob, at the runtime will have his shares for the $x$ and $y$, which are the actual keys corresponding to the actual values of $x$ and $y$. I also stressed that implicitly Alice also have the other keys associated with $x$ and $y$, namely, the key corresponding to $x$ = 1 and the key corresponding to $y$ = 1.

Now to make the value $z$ available in secret shared fashion as per the Yao secret sharing semantic, what Alice will do is the following. She will garble this AND gate. And for garbling, this AND gate, she will pick the 0 key associated with $z$ randomly, she will set the 1 key associated with z $K_1^z$ as per the free-XOR technique. And then she will prepare the 4 ciphertexts $c_1, c_2, c_3, c_4$, as per the point and permute optimization using her shares of $x$ and $y$ and the implicit keys corresponding to the $x$ wire and $y$ wire held by Alice.

So, remember for preparing the garbled circuit, we need all the 4 pairs of keys corresponding to the gate input and Alice indeed have all those 4 pairs of keys, because Alice only picked all the keys here. And depending upon the value of the permutation bits which are present in these keys, $c_1, c_2, c_3$ and $c_4$ill have different interpretations. So, it is not the case that $c_1$ is an encryption prepared using the 0 key for $x$ and 0 key for $y$, it depends upon the permutation bits.

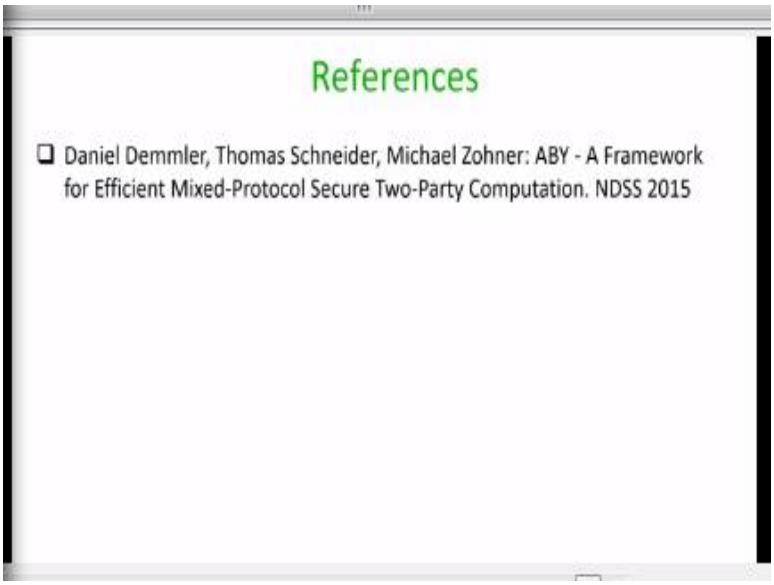
And now Alice will send these 4 permuted ciphertext to Bob and Bob in order to get his share for the value $z$, he will evaluate this garbled AND gate. So, that evaluation process I call as GB point and permute eval, what exactly is that evaluation process? So, Bob will take his share for the $x$ wire, his share for the $y$ wire depending upon the permutation bits which are there in the $x$ share and $y$ share. He will go and decrypt only the appropriate entry out of these 4 ciphertexts.

And as a result, he will get a key over the $z$ wire and that will be Bob's share. And by the correctness of the point and permute technique, it will be indeed guaranteed that the key which Bob obtains over the $z$ wire is actually the key which is supposed to hold as per the Yao's 2 party sharing semantics. And again, I stress here that the actions of Alice or the GC constructor she can do beforehand itself before the circuit evaluation starts, before the values of $x$ and $y$ get fixed.

Because to compute her share for the value $z$, all that ingredients required are available with Alice beforehand itself. Namely, she has all the 4 pairs of keys associated with the $x$ and $y$ wires, and she has the global offset. So, she can always prepare the collection of these 4 ciphertexts and she can keep her share for the $z$ wire as the 0 key. It is only the Bob's share which is determined at the runtime during the circuit evaluation.

Because in order to determine his share for the $z$ wire, he has to evaluate the circuit and to evaluate the circuit he needs the appropriate keys over the $x$ and the $y$ wires which are nothing but his share of the value $x$ and value $y$ respectively. And you can see here that in this process, the privacy of $x, y$ and $z$ is preserved. Because that comes from the privacy of your Yao's secret sharing protocol based on the point and permute optimization.

**(Refer Slide Time: 31:43)**




So with that I end today's lecture. So, I followed this paper, a framework for efficient mixed protocol secure 2 party computation to explain the sharing semantics of Yao's 2 party secret sharing. Thank you.