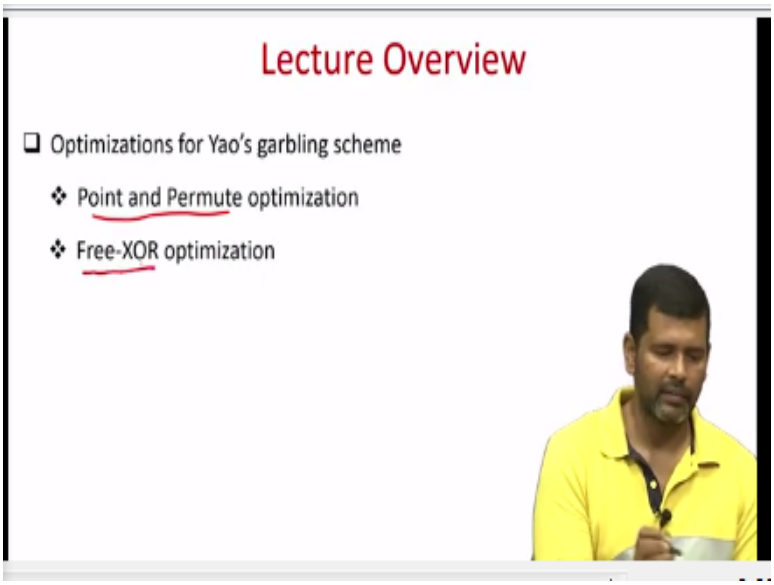
Secure Computation: Part I
Prof. Ashish Choudhury
Department of Computer Science Engineering
Indian Institute of Technology-Bengaluru

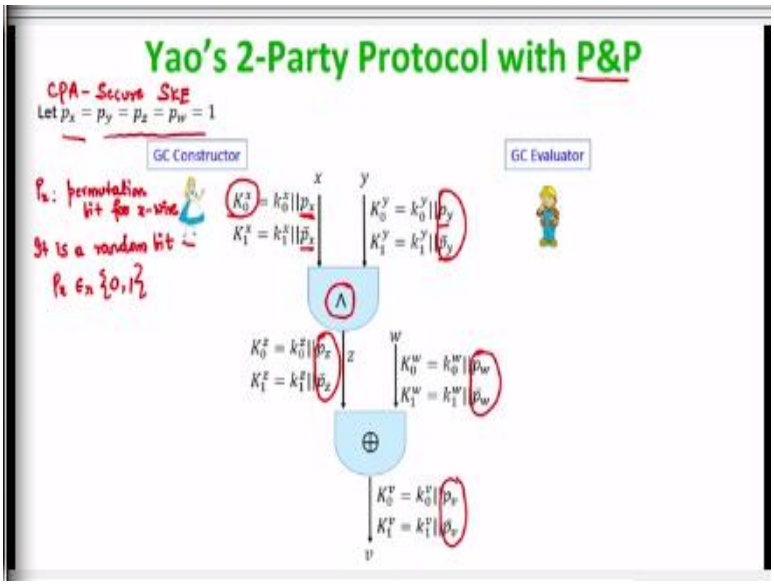
Lecture-51
Optimizations for Yao's Garbling

(Refer Slide Time: 00:32)



Hello everyone, welcome to this lecture. So, in this lecture we will see a couple of optimizations for the Yao's garbling scheme, we will see the point and permute technique which allows you to do the garbling using any regular secure symmetric key encryption and without requiring any special correctness property. And then we will see the free-XOR technique.

(Refer Slide Time: 00:57)



So, let us first see the point and permute technique here. For the purpose of demonstration, I am taking a circuit where we have an AND gate followed by XOR gate which needs to be evaluated. And now we are assuming that we have just a CPA secure symmetric key encryption, which we know how to build using pseudo random functions which can be instantiated using AES. And, we do not require any special correctness property.

So, as part of garbling the circuit, the garbled circuit constructor will still be assigning indistinguishable keys for each of the wires. So, you can see here for the x wire 2 keys are assigned, for the y wires 2 keys are assigned and so on. But the last bit of each of the keys has a special interpretation here. So p_x here will be called as the permutation bit for x wire and it is a random bit which could be either 0 or 1 with equal probability.

And the important thing is it is a random bit, it is not the case that since this whole key is for x being 0, the value of p_x has to be 0. That means p_x is a random value which is either 0 or 1. That means even though this entire key is for x being 0, the permutation bit p_x could be 0 with probability half it could be 1 bit probability half. It is only the constructor who will know what the value of the permutation bit is.

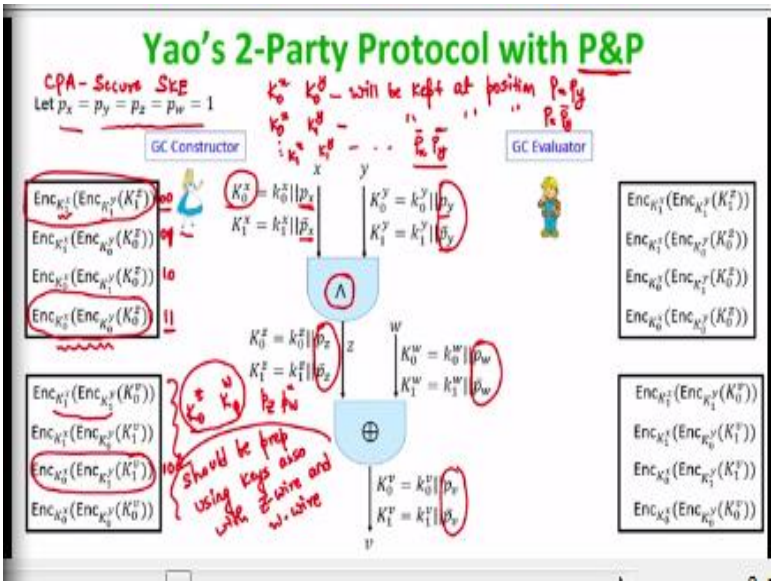
And the permutation bit which is there for the key corresponding to $x = 1$ it is just a complement of the permutation bit assigned to the key for x being 0. In the same way, you can interpret the

permutation bits here for the y wire, the permutation bits for the z wire and the permutation bit for the w wire and a permutation bit for the v wire. And the permutation bits for individual wires are randomly picked.

So, the permutation bit of y wire is completely independent of the permutation bit of the x wire and so on. So, the GC constructor is still doing the garbling of the wires as we had done till now, except that the last bit of the keys have some special interpretation. So, for simplicity, imagine that the constructor has picked the permutation bits to be all 1. Again this is a random event, for simplicity I am taking this case for the purpose of demonstration. It could be the case that say the permutation bits for x and y are 0, whereas the permutation bits for z and w are 1 and so, any combination could happen.

Any of the 16 combinations could happen. Now, Alice has to encrypt or Alice has to prepare the garbled AND gate. Preparing the garbled AND gate means preparing the double encryptions, encrypting the keys corresponding to z being 0, z being 1 using the various key combinations of the x key and the y key. Earlier she was preparing the 4 double encryptions and then she was randomly permuting them. Now using the help of this permutation bits she can avoid permuting those 4 ciphertext, let us see how.

(Refer Slide Time: 05:25)



So, the choice of the shuffling of the 4 ciphertext for the garbled AND gate is now decided by the values of these 4 permutation bits, what does it mean? So, it means the following. The encryption using the keys k_0^x, k_0^y will be kept at position $p_x p_y$ and the encryption using the keys k_0^x, k_1^y will be kept at position p_x, $\overline{p_y}$. And like that the encryption using the keys combination k_1^x, k_1^y will be kept at positions $\overline{p_x}\ \overline{p_y}$.

That means the choice of this permutation bits associated with the x wire and y wire will determine the position of the corresponding double encryption among the 4 ciphertexts. So, if we take the concrete values of p_x, p_y both being 1, now you can see that the encryption prepared using the keys with x being 0 and y being 0 is kept at the position number 3 assuming that the positions are indexed 00, 01, 10, 11.

So, depending upon the values of the permutation bits here, since p_x and p_y are both 1, the encryption using the key k_0^x, k_0^y has to be kept at position number $p_x p_y$.Since p_x, and p_y are both being 1, that means the ciphertext indexed with 11 will be actually the encryption of the key z being 0 prepared using the keys x being 0, y being 0. And in the same way you can see here that the encryption prepared with the keys for x being 1, y being 1 is now kept at the first entry.

A first entry means indexed with 00, why so? Because the encryption prepared with the keys x being 1 and y being 1 has to be stored at position number $\overline{p_x}\ \overline{p_y}$. Since p_x and p_y are both 1 here, their complements are 00 that means the ciphertext with index 00 actually is the encryption prepared with the key for x being 1, y being 1. Now only Alice knows this arrangement here because only Alice knows the permutation bits here.

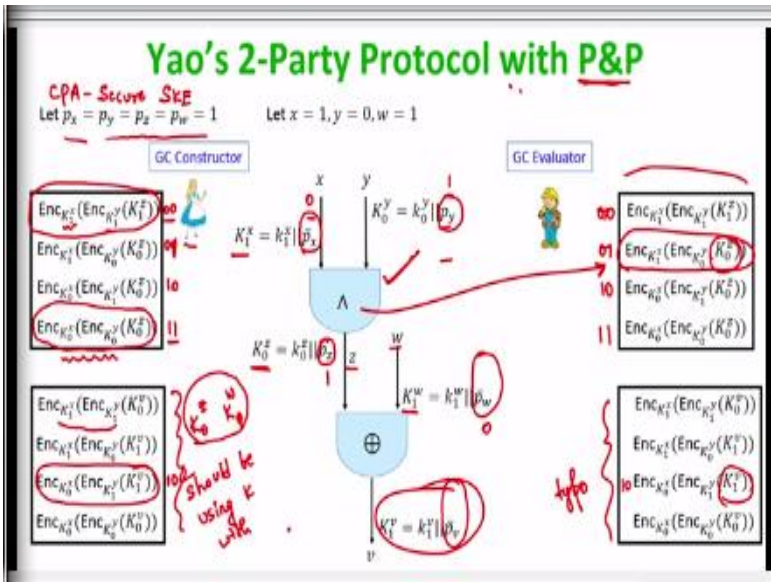
Bob is completely unaware of the permutation bits and their interpretation, it is only Alice knows the interpretation here. And again following the same logic when Alice will be preparing the double encryptions for the XOR gate, she will arrange the encryptions as determined by the permutation bits associated with the z wire. So, say for instance if I take the say this third entry here, why the third entry is like this here?

So, if I take the third entry here, the third entry should be corresponding to the key for z being 0 and w being 0. Now the ciphertext prepared using these 2 keys should be stored at position number $p_z\overline{p_w}$. What is the value of the permutation bit p_z? It is 1. What is the value of the permutation bit p_w? It is 1. So, its complement will be zero that is why it is kept at position number 10 and so on.

If the permutation bits would have been something else, then that would have automatically shuffled the position of the ciphertext. So, that is the only difference now compared to our regular garbling. And just associating this simple permutation bits automatically allows us to get rid of special correctness property that we required from our symmetric key encryption from our earlier garbling scheme.

Now this garbled table will be provided to the evaluator Bob and for him, it is now still 4 ciphertexts for each gate arranged in some random shuffled order. So, he cannot figure out whether the tables are arranged in the same order as the actual truth table or the order in which they have been shuffled. It could be in any order.

(Refer Slide Time: 11:54)



So, now let us see a concrete execution here and how exactly Bob will be determining which entry he has to decode for evaluating the gates here? So, I am taking the case when the inputs are x being 1, y being 0 and w being 1. So, if x is 1, then Bob will be seeing the key corresponding to x being

1. And if y is 0, then Bob will be seeing only the key corresponding to y being 0, he will not be seeing other keys.

And now he has to evaluate this AND gate, earlier using the 2 keys that he had 1 for the x wire, 1 for the y wire, Bob would have tried to decrypt all the 4 ciphertext. And a special correctness property would have ensured that he gets a non null output only from one of the 4 decryptions. But computing for decryptions for each gate is too much. Now, let us see how Bob can just go and do single decryption and magically identify which of these 4 ciphertexts he has to decrypt? What Bob will do is the following.

From the 2 keys that he has 1 corresponding to the x wire and 1 corresponding to the y wire, Bob will focus only on the permutation bits. He will not be understanding the interpretation of those permutation bits. So, for instance in this case $\overline{p_x}$ will be 0 and p_y will be 1, but just because the permutation bit for the x wire is 0 does not indicate to Bob that actually he is seeing the key corresponding to $x = 0$.

In fact he is actually seeing the key corresponding to x being 1 and that is why I said that the permutation bits are completely picked independent of the actual interpretation of the values associated with the wire. So even if it is $x = 0$ or $x = 1$ the permutation bit could be any random bit. In the same way just because the y key has the permutation bit 1, does not mean that he is actually seeing the key corresponding to $y = 1$, in fact he is seeing the key corresponding to $y = 0$.

So, based on the permutation bits Bob cannot find out what exactly is the label of the entire key that he is seeing for the x wire and y wire. But in terms of evaluation, what it has to do is the following. Since combined label of the permutation bits of x and y is 01 and the ciphertexts are index 00, 01, 10, 11. He has to go and only decrypt the entry stored at index number 01.

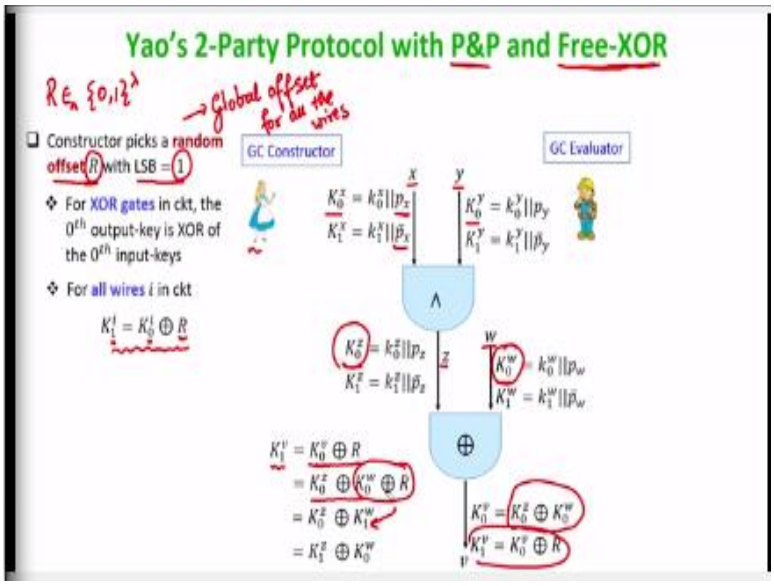
And this will not tell Bob anything whether he has decrypted an entry corresponding to the first row of the truth table or the second row of the truth table or the third row of the truth table or the fourth row of the table. So, in this case he will go and decrypt only the second entry and after

decrypting the second entry, the message that will come out is this key. So, that is the key that he will associate with the z wire.

So, here in the garbled table the encryptions are prepared using the z key and w keys not the x key and the y keys. So, now this gate is evaluated, now he has to evaluate this XOR gate, he has 1 key for the z wire, 1 key for the w wire. And again it will do the same process here; it will go and focus on the permutation bits. But again the value of the permutation bit does not help him to find out anything about the label of the entire key.

So, in this case the value of p_z is 1, the value of $\overline{p_w}$ will be 0, that means he has to go and decrypt the entry stored at index number 10, he do not have to try any other remaining 3 entries. And as a result, he will be able to get back this plaintext and that is the garbled output that he will finally obtain. So, now you can see that how by just assigning random permutation bits with each wire and just ensuring that the permutation bits assigned to each wire are complements of each other, allows Alice or the constructor magically to shuffle the ciphertext for each garbled gate. And magically it allows Bob also to identify which entry it has to decrypt without telling him anything about the semantic of that entry, so that is the point and permute technique.

(Refer Slide Time: 17:25)



Now let us see the free-XOR technique. How free-XOR technique allows to completely get rid of garbling while garbling the XOR gates? That means if they are XOR gates in the technique and if

you are using the free-XOR technique while preparing the garbled circuit, then neither the constructor nor the evaluator has to do anything to take care for this XOR gates. So, the way this free-XOR technique can be combined with the point and permute technique is as follows.

So, now the constructor will do the following, it will pick a random offset R which will be known only to the constructor. By the way one thing I forgot to mention here is that each time Alice and Bob want to participate in Yao's protocol, Alice has to do the garbling again. Because, if you see the syntax of the abstract garbling scheme, every time Alice and Bob participates in secure computation, the garbling has to be done randomly.

That means every time Alice wants to garble the circuit, it will be picking independent keys and performing the actions that I have discussed. It is not the case that the garbling is done once and for all and the same garbled circuit is used again and again and again for performing multiple evaluations of the circuit. So, coming back to the free-XOR technique, in the free-XOR technique combined with point and permute to garble the entire circuit, Alice will do the following.

It will pick a random offset R of length lambda bits, where lambda is your security parameter with the restriction that it is LSB is 1. Why this restriction? Because it will ensure that the permutation bits assigned to every wire are complement of each other. So, the first step that Alice will do is the following. For each wire, except the output wires of the XOR gates, Alice will assign randomly the 0 keys.

So, the x wires, y wires, they are assigned a random 0 keys where the last bit is the permutation bit. Now the z wire is output of an AND gate, so for that wire also a random 0 key will be assigned. This w wire is an independent wire, so for it are random 0 key is assigned but this v wire is the output of a XOR gate. So, for this v wire the 0 key is not assigned randomly.

In the point and permute technique without the free-XOR technique, even for the v wire a random 0 key would have been assigned. But now in the free-XOR technique, we are not assigning the 0 key for the outputs of the XOR gate. For the outputs of the XOR gate, the 0 key is set to be the

XOR of the 0 keys associated with the inputs of the XOR gate. So, in this case the XOR gate has the inputs z and w and their 0 keys have been already assigned here now.

And in the free-XOR technique, the 0 key associated with the output of this XOR gate which is the v wire is set to be the XOR of the 0 key assigned to the z wire and the 0 key assigned to the w wire that is how all the 0 keys are assigned. So, just for the XOR gates, there is a special provision that the 0 key is set to be the XOR of the 0 keys of the input. Otherwise all the other 0 keys are picked uniformly at random.

And now in the free-XOR technique, the keys corresponding to the value of 1 for each wire are no longer picked uniformly at random. Earlier they were picked uniformly at random but in the free-XOR technique, the keys corresponding to the value 1 associated with each wire is set to be the XOR of the 0 key for the wire and the random offset R. So, you can imagine that R is a like a global offset known only to Alice picked randomly for all the wires.

It is global offset in the sense that the key assigned with every wire corresponding to 1 is set to be the XOR of the key corresponding to 0 assigned to the same wire and the global offset R. That means now in terms of picking random values, the computation of Alice has been saved. Earlier Alice had to pick a pair of random keys for each and every wire in the circuit but now she has to pick random keys only 1 random key for every wire in the circuit.

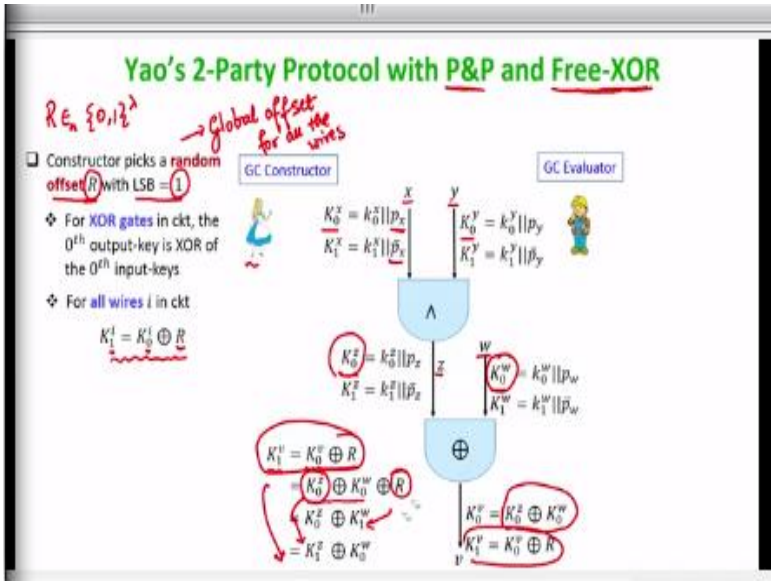
Of course for the output of the XOR gates, she does not even have to do that. And once she has picked the 0 keys for every wire randomly for the other key she do not have to pick random values. She just has to XOR the global offset with the keys which she has already picked for the value being 0. And now since we have ensured that the LSB of this global offset is 1, that automatically ensures that whatever restriction was imposed on the permutation bit for the point and permute technique to be applicable is automatically ensured.

Because once the 0 key has been fixed, whatever is the permutation bit that XORed with 1 will produce a complimentary bit? Why XORed with 1? Because when I will do the full XOR with R, the LSB of R is 1 that will automatically flip the permutation bit which has been assigned to the 0

key. So, that is how the keys corresponding to the value 1 will be picked, now the magic here is the following.

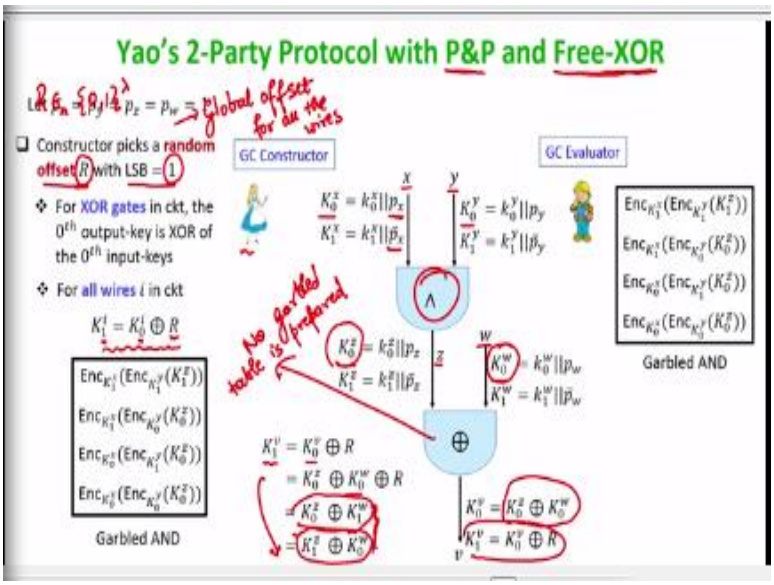
My claim here is the way the keys are assigned to the gates here if I consider the XOR gate here I can say that the following relationship holds. If I consider the key assigned to the v wire with value being 1, it is anyhow the XOR of the 0 key and the global offset. And what is the 0 key assigned to the v wire? It is anyhow the XOR of the 0 keys assigned to the inputs of this XOR gate and now I can say the following. If I consider this value, this is nothing but the w key corresponding to value 1.

(Refer Slide Time: 25:58)



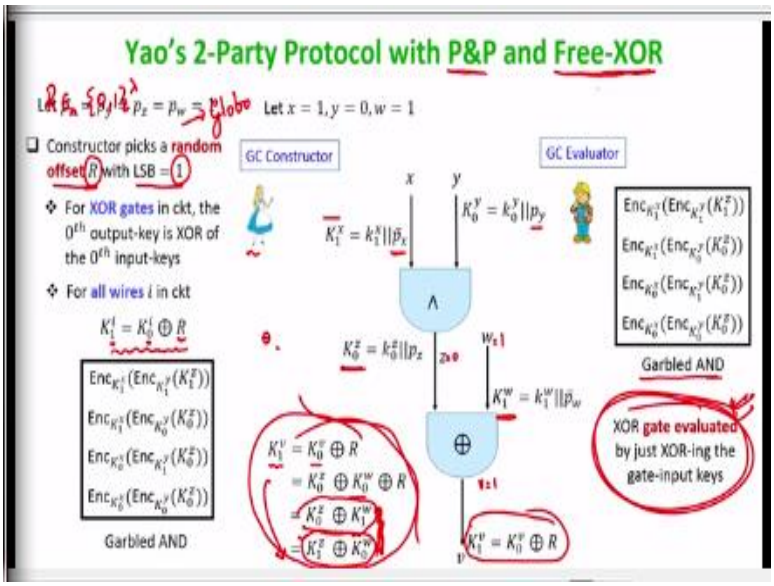
That means the value assigned to the v wire for the bit 1 is either the XOR of the key assigned to the z wire with 1 and w wire with 0 or it is the XOR of the z wire key assigned value 0 and the w wire key assigned value 1.

(Refer Slide Time: 26:48)



That means these 2 properties hold and this will be crucial later when we will see the evaluation of the circuit. So, this is how the wires are garbled by the constructor in the free-XOR technique. Now the construction of the garbled gate happens as it was done earlier. So, the AND gate will be garbled and the ciphertexts will be prepared and they will be kept according to the values of the permutation bits. But for the XOR gate, no garbled table is prepared. Now if no garbled table is prepared, you might be wondering how the evaluator will be evaluating this XOR gate at the time of circuit evaluation. Well, it will be clear very soon.

(Refer Slide Time: 27:56)



So, at the time of evaluation, evaluator will obtain only one table because it is only one gate which is garbled in this example. So, he will obtain a set of 4 shuffled ciphertexts. And again imagine

that the values of x, y and w are this. So, it will see a key over the x wire, he will see a key over the y wire. Then it will go and parse the permutation bits and it will go and decrypt the corresponding ciphertext.

After decrypting the corresponding ciphertext, he will now have 1 key over the z wire, 1 key over the w wire. But now he has to find out what key he should have over the v wire? And to find out what key over the v wire he should have, he just goes and XOR the keys over the z wire and w wire, that is all, that is simply the evaluation of the XOR gate. And my claim is by doing the operation; he will magically obtain the right key that he is supposed to obtain over the output of this XOR gate, why so?

In this case the keys over the z wire and the key over the w wire are corresponding to z being 0 and w being 1. That means the evaluator is actually evaluating the circuit for the case where the inputs of this XOR gates are $z = 0$ and $w = 1$. That means he should obtain the key corresponding to $v = 1$ because if the inputs of the XOR gates are different and the output is 1. And indeed if you see that if we XOR this full 0 key associated with z wire and it is full 1 key associated with the w wire, that will give us the full key with the v wire corresponding to 1 because of this relationship. Because that is how the constructor would have associated the keys with the outputs of this XOR gates. The keys associated with the output of this XOR gates have this property that the 0 keys is the XOR of the 0 keys with the input wires.

So, if at the time of evaluation Bob would have got 0 keys for both the input wires of this XOR gate. Then by doing this evaluation process, anyhow he would have got obtained the 0 key associated with the output of this XOR gate. And that is an invariant which we wanted to maintain. But if we have a 0 key over one of the input wires and 1 key over another input wire, then even by XORing them, he will obtain the key corresponding to the value 1 over the output wire.

Because of the relationship that is maintained at the time of garbling the wires. And that is why in this free-XOR method, the constructor do not have to prepare any ciphertext for garbling this XOR gates. And for evaluating this XOR gate, the evaluator just has to perform the XOR of the keys

that it has over the input wires of the XOR gate. That will magically produce the right output wire key for that XOR gate. So, that is the free-XOR technique.

(Refer Slide Time: 31:43)

So, again I am not going through the full formal security proof and so on. If you want to go through the full security proof, you can go to the references which are available at this site. Thank you.