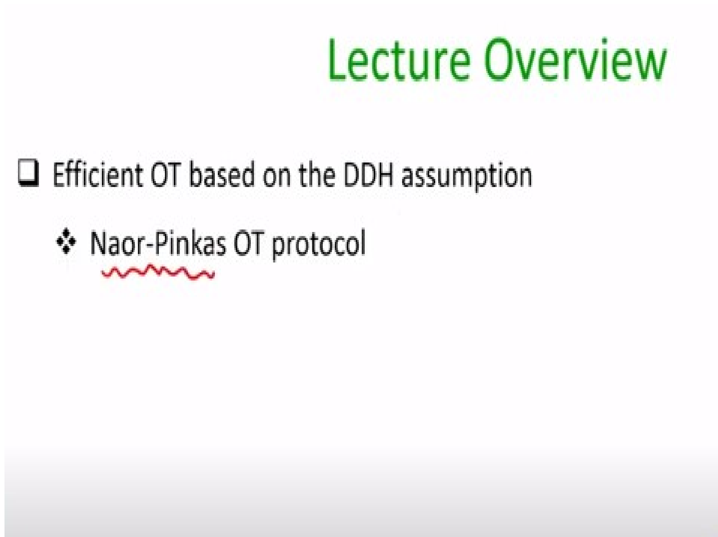
Secure Computation: Part 1
Prof. Ashish Choudhury
Department of Computer Science & Engineering
International Institute of Information Technology-Bengaluru

Lecture - 41
OT Based on the DDH Assumption

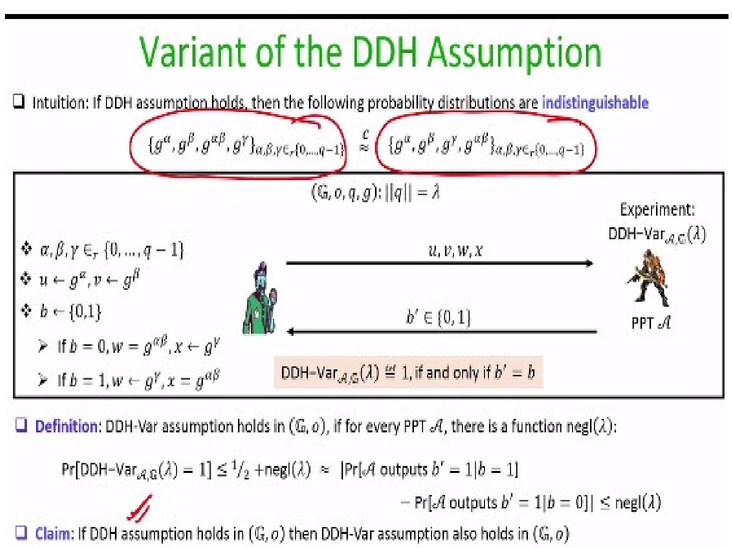
Hello everyone, welcome to this lecture.

(Refer Slide Time: 00:32)



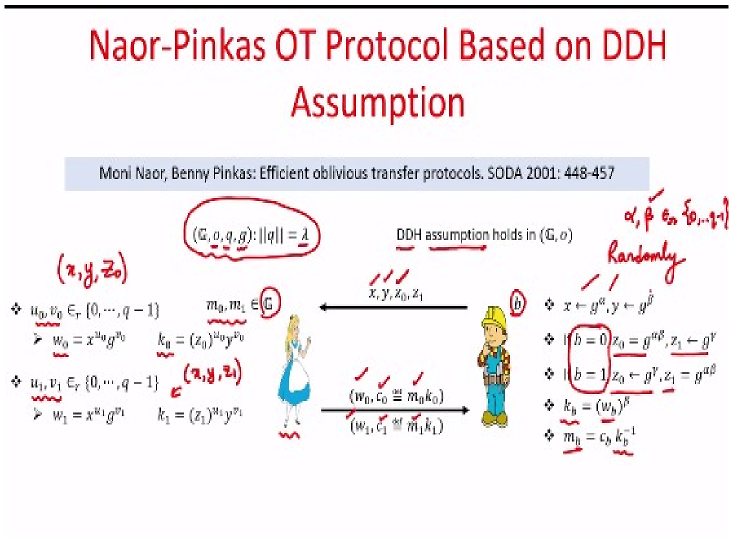
You will now see how to construct oblivious transfer protocols where the inputs of the centers are going to be strings based on DDH and its related, DDH problem and its variant. So this OT protocol is due to Naor and Pinkas.

(Refer Slide Time: 00:51)



So just to quickly recap, this was the variant of the DDH assumption or the DDH problem which we assumed to be difficult to be solved in polynomial amount of time. The idea is that these two probability distributions are computationally indistinguishable for any polynomial time adversary, okay. And we can prove that if the DDH assumption holds in the underlying group, then indeed this variant is also difficult to solve in the group, okay.

(Refer Slide Time: 01:23)



So now coming to the Naor-Pinkas OT protocol and this is a very efficient oblivious transfer protocol, where the inputs of the sender are going to be the messages m naught and m 1 which are elements of the group G where G is a cyclic group. The description of the cyclic group is publicly available. Each element of the group is represented by lambda bits where lambda is your security parameter.

The underlying group operation is o. The group size is q. That means the generator is small g raised to the power 0 and all the way to the generator raised to the power q – 1 will give you all the elements of the group G, okay. And this group is such that the DDH assumption holds. That means, it is assumed that no polynomial time algorithm can solve an instance, random instance of the DDH problem except with negligible, except with probability half plus negligible, okay.

And the input for the receiver is a choice spade and we want to design a protocol so that at the end of the interaction receiver receives the message m sub b and it does not learn anything about the other message. Whereas, we also want to ensure that center does not learn which message got finally transferred to the receiver. That means it should not learn anything about the choice bit b.

So here is how the protocol proceeds. So the receiver goes first assuming that parameters have been established and they are set up once at the beginning of the protocol and then the same setup is going to be used for polynomial number of invocations of the OT protocol between this sender and receiver, okay. So now the receiver is going to send four group elements to the sender.

And let us see how these four elements are computed. So the first two elements are computed randomly okay, and how receiver can compute x and y, which are random group elements. Well, it has to just pick alpha and beta randomly from the set 0 to q - 1 and then compute g to the power alpha, g to the power beta. And now it does the following.

Depending upon what exactly is his choice bit, whether it is b = 0 or whether it is b = 1, it does the following. If b is equal to 0, then the component z naught along with x and y constitute a Diffie-Hellman triple and the component z 1 along with x and y constitute a non Diffie-Hellman triple.

Whereas if he is interested in the message m 1, then he sends, he arranges z naught and z 1 in such a way that x, y along with z 1 constitute a Diffie-Hellman triple, whereas x, y along with z naught constitutes a non Diffie-Hellman triple. Now Alice is not aware what exactly is the value of choice bit b and hence it cannot figure out what exactly is the arrangement of z naught and z 1.

Alice knows that x and y are random group element, but whether it is the third component along with the first two components is Diffie-Hellman triple, or whether it is the fourth component along with the first two components is the Diffie-Hellman triple, Alice is unaware because that is precisely is the variant of the DDH assumption.

So since we are assuming that the DDH assumption holds, Alice cannot figure out the type of arrangement of z naught and z 1. Now what Alice is going to do? Alice has to somehow send the messages m naught and m 1, both of them to Bob in such a way that finally Bob should be able to get back only the message m sub b, but he should not be able to get back the message m 1 – b, okay.

That is what we will do in every oblivious transfer protocol. Sender has to somehow send both the messages and it has to do it in such a way that receiver should have the capability to recover back only one of the messages that it is interested in. So now to send the messages what Alice is going to do is, Alice is going to generate two keys k naught and k 1.

And k naught she will be using for masking the message m naught, k 1 she will be using for masking the message m 1. And then she will mask the messages m naught and m 1 and send it to Bob in such a way that Bob will be able to unmask only one of the messages. So now how she is going to derive the keys? The way she derives the keys is as follows.

So she first computes or randomizes the components x, y and z naught. While she does not know whether x, y, z naught constitute a Diffie-Hellman triple or a non Diffie-Hellman triple. But she considers x, y, z naught as a triple as a whole and tries to randomize it and generate a key out of it. The way she randomizes this triple is as follows.

She picks two random indices u naught and v naught, computes this value w naught and then computes this value k naught, okay. Now this might look like some vague computations here. Do not worry about that. There is some magic hidden here, these are not vague computations. And she takes the other triple namely the first component, the second component and the fourth component.

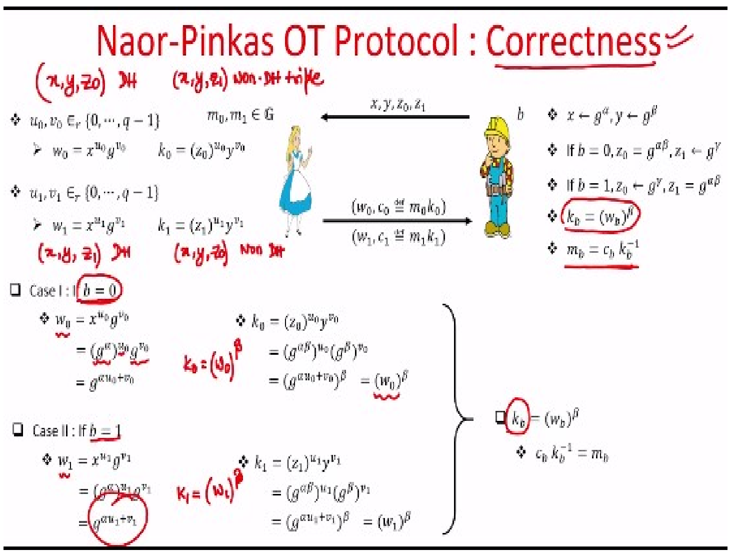
And she does not know whether this is the Diffie-Hellman triple or the non Diffie-Hellman triple, she randomizes this triple as well. So x, y and z 1 they are randomized as follows. She picks a randomizing pair of components u 1, v 1 and computes w 1 and k 1 as follows. And now she sends the masking of her messages. So c naught is the masking of the message m naught and the masking operation is multiplying with k naught.

k naught and k 1 are both going to be group elements and hence c naught is also going to be a group element. And c 1 is the masking of the message m 1 where the key k 1 is used for masking purpose. Now to enable receiver to get back one of these two keys k naught and k 1, Alice also sends the w naught component and w 1 component here. Now if Bob wants to recover the message m sub b, he does the following computations.

He takes the message w sub b and raise it to the power beta. He has the beta, so he raises it. And then whatever value he obtains my claim is that is the value of key k sub b. Once the value of key k sub b is learned by Bob he has to just unmask it. Unmask it means divide it okay and divide in the context of group here is multiplying with the multiplicative inverse of k sub b.

And if he unmask c sub b, he gets back the message m sub b. That is the whole protocol. It just involves two rounds of interaction. And now we have to analyze this protocol why these computations are performed like this, why it will maintain the correctness property, receiver's security and sender's security and so on.

(Refer Slide Time: 09:41)



So let us first quickly understand the correctness property. We have to show here that indeed Bob will be able to recover back the message m sub b correctly. We are not considering the security argument right now. And there are two possible cases depending upon whether the Bob's choice bit b is 0 or 1. If Bob's choice bit is 0, then what can we say about x, y and z naught.

It is a Diffie-Hellman triple. And what we can say about x, y and z 1? It is a non Diffie-Hellman triple, okay. So b is equal to 0, then let us focus on the randomization of x, y and z naught triplet as performed by Alice. Namely let us see what exactly is the value w naught. Now w naught is nothing but x to the power u naught multiplied by g to the power v naught, okay.

And now x is nothing but g to the power alpha and that whole raised to power u naught multiplied by g to the power v naught. And if I now apply the rules of discrete logarithms, we can rearrange the terms in the exponent and everything is multiplied and added. And now if I focus on the k naught portion of the computation here, k naught is z naught. z naught in this case is g to the power alpha times beta.

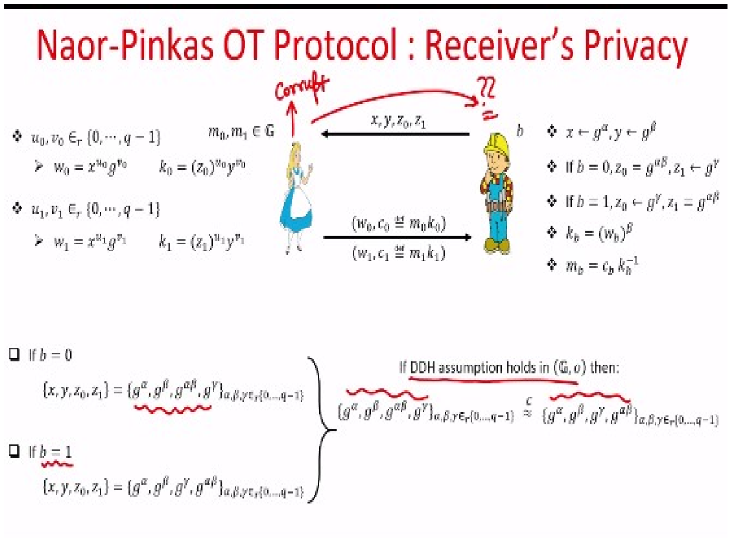
And now if I rearrange the terms, I find out that k naught is nothing but w naught raised to the power beta. The case two could be when Bob's choice bit was 1. If Bob's choice bit was 1, then x, y and z 1 constitutes a Diffie-Hellman triple. And x, y, z naught constitutes a non Diffie-Hellman triple. In this case, let us focus on the w 1 portion or the w 1 value as computed by Alice.

And w 1 is x power u 1, g power v 1. And if again if I expand x, I get the value of w 1 to be this. And with respect to that w 1 if I consider the value of k 1 as computed by Alice, I get that k 1 is nothing but w 1 power beta. So what we have shown here is that it does not matter whether b is equal to 0 or whether b is equal to 1, the value of the key k sub b is the w b element raised to the power beta.

If b is equal to 0, then indeed k naught is equal to w naught raised to the power beta. If b is equal to 1, then the k 1 portion of the key or the k 1 key computed by Alice is w 1 raised to power beta. And Bob has arranged this z naught and z 1 in such a way that it will know what exactly is the key that it should unmask from w naught or w 1 to recover back the message.

So that is why it knows the indexed b. So it can compute the key k sub b, which is required for unmasking the message m sub b by computing this value. And once it got, once it gets k b it can easily unmask the message m sub b. So correctness is guaranteed here. That means Bob indeed will be able to correctly get back the message m sub b.

(Refer Slide Time: 13:30)



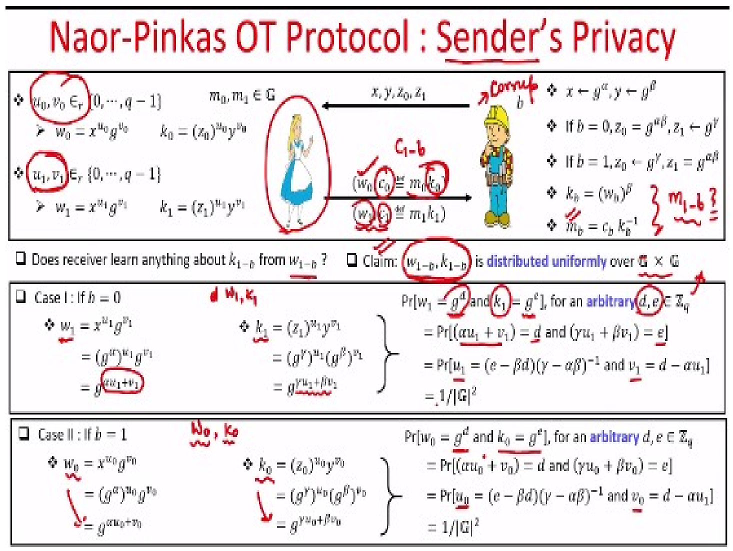
Now let us focus on the receiver's privacy, whether this corrupt Alice learns anything about the choice bit b. If b is equal to 0, then the arrangement of x, y, z naught and z 1 will be as per this probability distribution namely the first three components are Diffie-Hellman triple, the first two components and the last component is a non Diffie-Hellman triple.

Whereas if Bob's choice bit is 1, then the arrangement of z naught and z 1 is such that the first three components are non Diffie-Hellman triple and the first two components along with the fourth component is a Diffie-Hellman triple. Now since we are assuming that the DDH assumption holds in my underlying group, from the viewpoint of a polynomial time Alice she cannot figure out whether x, y, z naught, z 1 is of this type, is of this type distribution or whether it is of type this distribution.

Because if in polynomial amount of time Alice can figure out what is the type of x, y, z naught and z 1 she has seen, then Alice can be used to break the variant of the DDH problem or solve the variant of the DDH problem, which is a contraction to the assumption that the variant of DDH problem is difficult to solve in the group. So that ensures receiver’s privacy.

That means a computationally bounded a polynomial time Alice or sender does not learn anything about Bob’s choice bit.

(Refer Slide Time: 15:16)



Now let us try to argue about sender’s privacy regarding Alice privacy. Here we have to assume that Bob is corrupt, the receiver is corrupt and he, we have to figure out whether he learns anything about the message m sub 1 – b. That means the other message. Of course, it will learn the message m sub b because it is supposed to learn it. For proving the sender’s privacy, we have to argue that he does not learn anything about the other message.

And to learn about the other message he has to learn about the key k sub 1 – b. Because, if it cannot find out what is the key k sub 1 – b, then this value w sub 1 – b, which is the message m sub 1 - b masked with the key k sub 1 - b is going to be like a random element for Bob. So everything boils down to how much he learns about this key k sub 1 - b by interacting with Alice or namely from this message w sub 1 – b.

And what we are going to prove here is a very interesting fact here. We are going to prove that the pair of values w sub 1 - b and k sub 1 - b is a random pair of group elements from the viewpoint of a corrupt Bob. That means, there can be any two group elements. They have got absolutely nothing. It has got no information regarding or no bias. They are completely distributed uniformly over the set G cross G.

That means it could be any pair of group elements, which Bob will see as w sub 1 - b and k sub 1 – b. And hence, it could be any message m sub 1 - b masked with the key case of 1 – b, okay. So that means the element c sub 1 – b, which will be like a completely random element from the viewpoint of a corrupt Bob and hence it cannot figure out whether, what message m sub 1 - b is actually masked in this ciphertext c sub 1 - b.

So let us prove this claim. And for proving this claim we will show that the claim holds both for the case b = 0 as well as b = 1. If b = 0, we have to show that k 1, yes we have to show that w 1 and k 1 are like random elements for the Bob. It cannot figure out what exactly is the value of k 1 and w 1 could be any element, any random element from the viewpoint of the Bob, okay.

So for proving that, for the case b = 0, let us write down the value of w 1. And this w 1 for the case b = 0 is actually computed using this randomizing components u 1 and v 1, which are picked by Alice. And due to that the value of k 1 which is computed by Alice will be this. So that is the value of w 1 and k 1. Now we want to prove that this k 1 and w 1 could be any arbitrary element from the group.

So let us try to compute the probability that what is the probability that this k 1 and w 1 are the group elements g to the power d and g to the power e from the group big G, right? So what we are trying to argue here is that you take any element from the set big G, okay. Let this be the element and let this be with another element. What is the probability that w 1 is this first element and k 1 is this second element?

So this specific element or the arbitrary element which I am taking since it is an element of the set big G, it can be represented as some g power d because it will have some discrete logarithm. Let us call that discrete logarithm is d. And in the same way the second element k 1, which we are trying to find out what is the probability that this is this specific element.

This specific element I can write it as g power e where e is the discrete logarithm of that arbitrary element. So we are trying to analyze is there a probability that indeed w 1 is equal to some g power d and g power e where you fix the d power e, d and e arbitrary from the set 0 to q – 1. And the probability turns out to be nonzero. Why? Because the probability that w 1 is actually the element the g power d is same as this exponent of w 1.

So what is the discrete logarithm of w 1? The discrete logarithm of w 1 will be alpha times u 1 + v 1. Of course modulo q. And the discrete logarithm of k 1 will be gamma times u 1 plus beta times v 1, of course modulo q. You want to analyze what is the probability that these two values are equal to respectively d and e.

And they will be respectively equal to d and e provided the randomizers u 1 and v 1 which are picked by Alice takes these values, okay. And since u 1 and v 1 are picked randomly by the Alice, because Alice has picked these randomizers uniformly at random, u 1 can take this value with probability 1 over q. v 1 can take this value with probability 1 over q. And q is nothing but the order of the group.

So the probability that Bob when he is seeing w 1 and c 1 and w 1 turns out to be some specific element of the group and corresponding k 1 is some another element of the group is 1 over the square of the group size or group order. That means from the viewpoint of Bob this w 1 could be any g power d. He cannot pinpoint that okay the w 1 that I am going to see is only this specific g power d.

It could be any element g power d from the group. That is what we have established. And hence the corresponding k 1 also could be any element from the group. So Bob cannot figure out from w 1 what is the value of k 1 which has been used to mask this message c 1.

It could be any k 1 which has been used by Alice for masking the message m 1 and which has the effect on producing c 1 and hence it cannot find out anything about the message m sub 1 by analyzing w sub 1 because w 1 is like a completely random element from the viewpoint of Bob if he is corrupt. So this was the analysis for the case b = 0. We can run similar analysis for the case b = 1.

If b = 1 then Bob will learn the message m sub 1. We have to show that w naught and k naught are kind of useless for the Bob. Of course, w naught is seen by Bob because he will be seeing w naught. We have to argue that w naught is going to be any random element from the group and independent of that k naught is also going to be any random element of the group.

And k naught is anyhow not seen by Bob. So that means this c naught is completely like a random ciphertext for this Bob because it is the masking of some unknown m naught with a almost random element k naught, okay. Why this is the case? Because again if I now work out the maths here, the w naught value takes this form, k naught takes this form.

And now if we try to argue that what is the probability that w naught is some arbitrary element g power d in the group and the corresponding k naught is some another arbitrary element of the group. Well the probability that, probability of this event happening is same as Alice would have used the randomizers u naught and v naught respectively satisfying these conditions.

But since the randomizers u naught and v naught are picked uniformly at random, the probability that u naught and v naught can take these respective values is same as 1 over this square of the group order. That means, even though Bob for the case b = 1 is seeing w naught, w naught is not any biased element, it is a random element for Bob.

And the corresponding k naught is a uniformly random and unknown element for Bob. And hence, the masking of the message is like a random masking and it cannot figure out what is the message m naught. So that ensures sender's privacy. And now hopefully that helps you to understand that why Alice is performing computations like this.

The underlying idea behind this protocol is that somehow magically Alice should be able to use the key case sub b based on the Diffie-Hellman component, Diffie-Hellman triple component out of these four elements. And Bob should be able to use only that keys k sub b for unmasking the message it is interested in.

But the non Diffie-Hellman triple component of the four things that Bob has sent should be randomized by Alice in such a way that it changes to a completely uniformly random key, so that Bob cannot infer anything about the other message because the other key is uniformly random from the viewpoint of Bob. That is the intuition behind this protocol, okay. So that brings me to the end of this lecture.

Now we have seen an oblivious transfer protocol where the inputs of Alice are no longer bits, they are now group elements. So they are lambda bit long strings. So if Alice has two messages, which are lambda bit long strings, she can encode them as group elements and run this instance of Naor-Pinkas oblivious transfer protocol. This is a very efficient oblivious transfer protocol. Thank you.