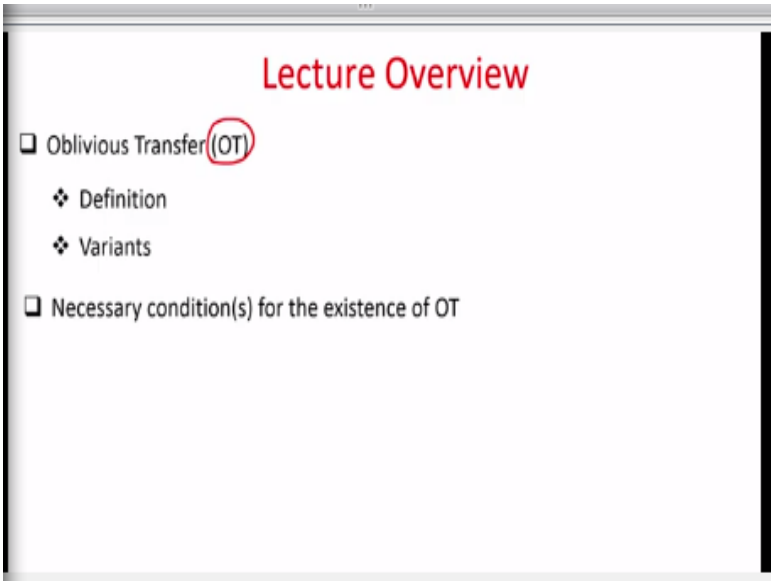
Secure Computation: Part I
Prof. Ashish Choudhury
Department of Computer Science Engineering
Indian Institute of Technology-Bengaluru

Lecture-37
Oblivious Transfer (OT)

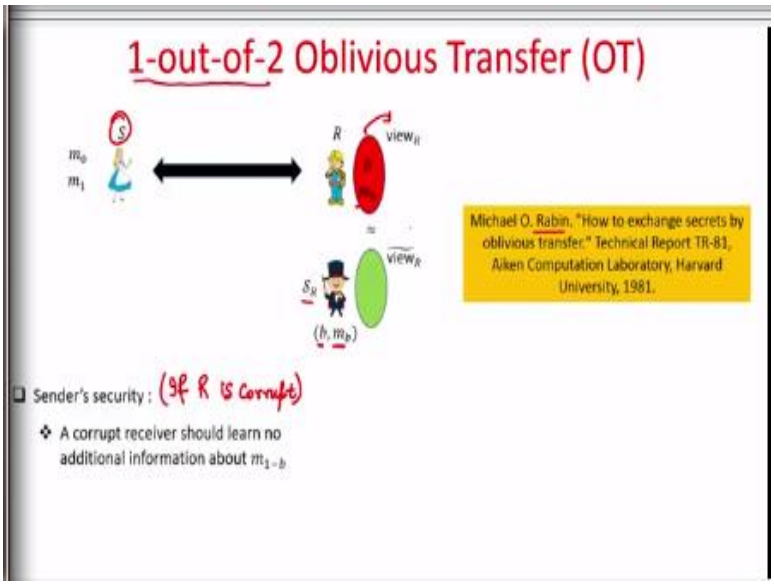
(Refer Slide Time: 00:32)



Hello everyone, welcome to this lecture. So, in this lecture we will now discuss a very important cryptographic primitive which is called as oblivious transfer or in short OT. Why do we want to discuss oblivious transfer? Because remember, we want to design the pre processing phase or implement the pre processing phase for generating additively shared multiplication triples and that is in the honest majority setting. And oblivious transfer is going to play a very crucial role there.

Not only that, oblivious transfer is used in varieties of other cryptographic problems and applications - we will not be going through all the applications of oblivious transfer. Oblivious transfer in itself is a very important cryptography primitive and it has been widely studied. So, in this lecture, we will see the various variants and definition of oblivious transfer and what are the necessary conditions that we require for the existence of oblivious transfer protocols?

(Refer Slide Time: 01:33)



So, we will start with the definition of 1 out of 2 oblivious transfer or OT. And oblivious transfer was introduced or invented by Michael Rabin and his seminal work. So, I am going to discuss a variant of oblivious transfer. So, the way I am going to discuss the oblivious transfer here is not the way Michael Robin invented or proposed it. We will see the variant at or the original definition that Robin gave and later we will see that actually both these variants are equivalent to each other.

The reason we are considering a variant of the oblivious transfer compared to the original form of Robin's OT is that we will use this variant of OT that we are going to see now while designing our MPC protocol and the pre processing phase. So, in the problem statement, we have a sender S and a receiver R. Sender has a pair of private inputs a pair of messages m_0 and m_1, they could be as small as 2 bits or they could be arbitrary long strings, but finitely long strings.

And receiver has a private input which is a selection bit, which is either 0 or 1. This oblivious transfer is a protocol according to which the sender and a receiver will interact. And at the end of the protocol, only receiver receives an output, sender does not have any output. What output does the receiver receive? It receives the message which is dependent on it is selection bit. So, you can imagine that it is selection bit b datermines the message which it wants to receive from the sender.

Sender has 2 messages, a message with index 0, a message with index 1, receiver does not know the values of those messages to begin with. But he has a choice bit, if his choice bit is $b = 0$, then

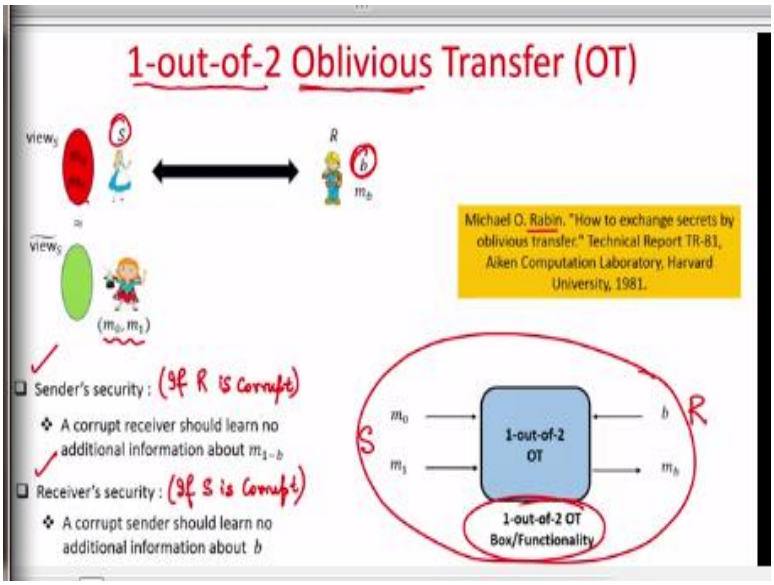
at the end of the OT protocol it should receive the output m_0. If it is choice bit is 1 then at the end of the OT protocol, it should receive the message m_1. Now you might be wondering what the big deal in this is, sender could have directly given both the messages m_0 and m_1 to receiver?

Well, here are some security properties which we require from the oblivious transfer, depending upon whether the sender is corrupt or the receiver is corrupt. So, we first consider the case where if the receiver is corrupt. So, if the receiver is corrupt in the protocol then we require that it should only learn the message which it is interested to receive. So, if it is interested to receive the message m_0, it should receive only the message m_0.

And during the interaction with the sender, it should not learn anything about the message m_1. Whereas if it is interested to learn the message m_1, it should get message only the message m_1 and not the message m_0. That is what we require during the oblivious transfer protocol, that is what the sender's security is. What does it mean that? It means the receiver does not learn anything about the other message.

So, if the receiver is corrupt, then this will be the view of the adversary, it is view will be it is selection bit, the output m_b, its local randomness and whatever messages it has received from the sender. Not learning anything about that the other message is formalized by saying that there exists a simulator which when given the receiver's input, namely it is choice bit and a receiver's output, it could simulate a view whose probability distribution is computationally indistinguishable from the view of the receiver which it would have obtained by interacting with the sender. Namely, what we are saying is that whatever receiver could learn by interacting with the sender, it could learn even without interacting with the sender. Then it is equivalent to saying that it does not learn m_{1-b} – namely, the other message. That is the sender's security.

(Refer Slide Time: 06:28)



Whereas if the sender is corrupt, then we require that during the interaction with the receiver, the choice bit of the receiver should not be revealed to the sender. Sender will know that ok, one of the 2 messages has been transferred obliviously to the receiver, which is why the term oblivious. But it should not learn whether it was the message m_0 or whether it the message m_1 which got delivered to the receiver, that is why the name oblivious transfer.

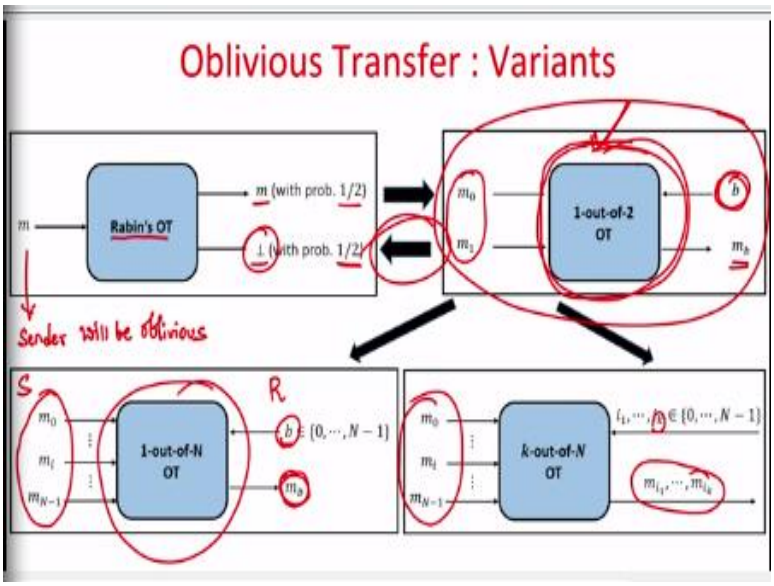
Otherwise the receiver could have just asked the sender, ok, I am interested in the message $b = 0$, please give it to me. But in that process, sender learns that choice bit, which we do not want that to happen during the interaction. So, this is formalized again by saying the following, that whatever is the view of a corrupt sender which it generates while interacting with the receiver, the same view can be recreated or simulated by a simulator just based on the inputs of the sender.

And a simulated view will be should be computationally indistinguishable from the real view of the corrupt sender. So, we will see later the design of the oblivious transfer protocols because as I said oblivious transfer is a very, very fundamental primitive in cryptography. It has got varieties of applications but from now onwards whenever I am going to assume that we have an oblivious transfer protocol 1 out of 2 oblivious transfer protocol I will use this notation.

This notation will mean that I have an secure oblivious transfer protocol achieving both these 2 properties, where sender will feed it is inputs namely m_0, m_1. A receiver will feed it is input

namely it is selection bit and it will obtain the message that it is interested in. Why is it called 1 out of 2 OT? Because the receiver is interested to learn 1 out of 2 messages.

(Refer Slide Time: 08:53)



So, now as I said, that there are various other variants of oblivious transfer. So, let us first start with Rabin's original formulation of oblivious transfer, the way he defined the problem in his original paper. So, in his paper sender has only 1 message as the input, say the message m and receiver can have 2 possible outcomes, it can either obtain the message m as the output with probability half or it will obtain a null output with probability half but sender will be completely oblivious.

What exactly has happened? It will not know whether the message m got transferred or delivered or whether $\perp$ got delivered to the receiver. And if the receiver receives the output $\perp$ then it will not be knowing what the sender's input was. So, that was the original formulation of oblivious transfer by Robin. Whereas the way we have defined oblivious transfer, we have assumed that sender has 2 messages, receiver has a choice bit and depending upon the choice bit it receives the corresponding message.

As I said the reason we are following this definition of oblivious transfer is because it is easy to use this version of oblivious transfer later when we want to design the MPC protocols. But now if you are wondering whether the Robin's version is same as the new version or the alternate version

that we have defined here, the answer is yes. Because it can be shown that if you have a secure protocol for implementing Robin's set OT then you can use that secure protocol to get another OT protocol satisfying the requirements that we have just presented.

And the other way around also holds if you have a protocol for this variant of oblivious transfer, you can design a protocol satisfying Robin's original formulation. Now, the notion of 1 out of 2 OT can be generalized to 1 out of N OT, where now sender has N number of messages. Instead of 2 messages it has now N messages and receiver will have an index depending upon that index which could be any index in the range 0 to $N-1$, it should get back the corresponding message.

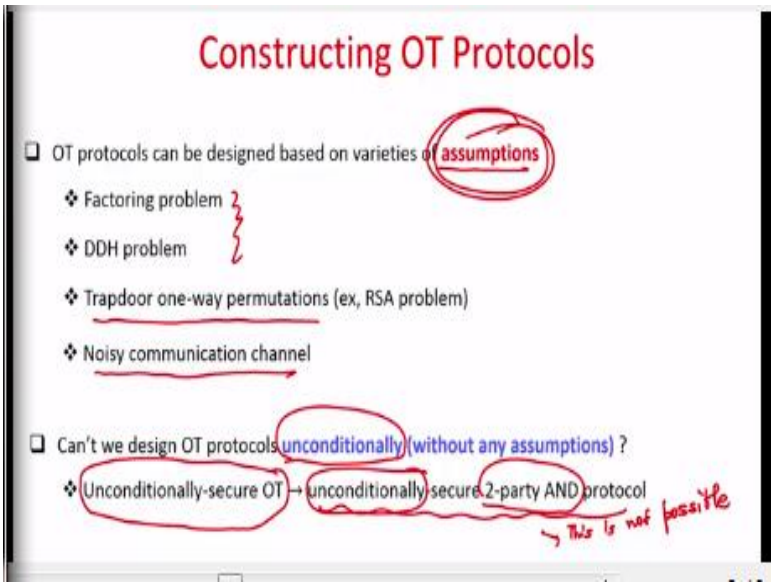
But during the interaction that happens between the sender and the receiver, a corrupt sender should not learn anything about the index of the receiver. Namely, sender should not learn what message got transferred to the receiver and receiver should learn only the message which it is interested in, it should not learn anything about other $N-1$ messages. So, it can be shown that if you have a secure mechanism to implement 1 out of 2 OT, then you also have a secure mechanism to implement 1 out of N OT, we will soon see that.

And then we can further generalize the 1 out of N OT problem to k out of N OT. Namely, in the k out of N OT sender has N number of inputs and receiver has k number of choice indices. And depending upon which k messages it is interested to receive, it receives those k messages. Again, a corrupt sender should not learn what the k messages which got transferred are, a corrupt receiver should learn only the k messages it is interested in, it should not learn anything additional about the remaining messages.

One question that might come to you is that why cannot receiver receive both the messages by participating in the oblivious transfer twice? And in the first instance it participates in, its choice bit will be equal to 0 and in the second instance it will be equal to 1. Well, that is quite possible but the use case of the oblivious transfer later in the MPC protocol will be that we will be using OT instances over inputs which are decided during the MPC protocol.

And in each instance a sender and receiver will be participating with random inputs. So, every time the same sender and receiver gets involved in an OT interaction the inputs and the choice bits will be different. And again it can be shown that if you have a secure instantiation of 1 out of 2 OT then you can get a secure instantiation for k out of N OT as well.

(Refer Slide Time: 14:06)



So, now how do we construct OT protocols? As I said OT is a very, very fundamental primitive cryptography. It is one of the widely studied topics and there are several ways to design OT protocols based on various assumptions. Namely, assuming that we have difficult problems to solve say the factoring problem, the decisional Diffie–Hellman problem, we can construct OT protocols.

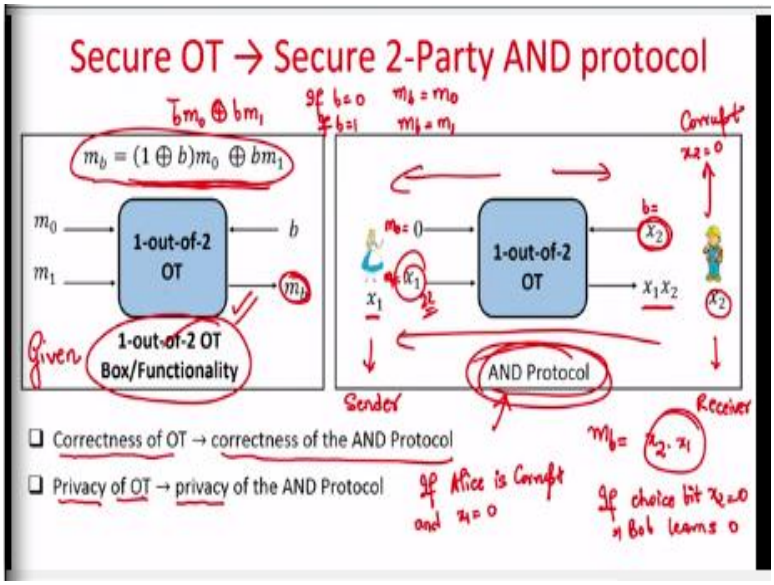
Assuming that we have Trapdoor one-way permutations say the RSA Trapdoor permutation, we can build OT protocols. We can also build OT protocols assuming the communication channel between the sender and a receiver is noisy and so on. So, now an important question that you might ask is the following, why assumptions have to be made while designing or constructing OT protocols, why cannot they be built unconditionally without making any assumptions whatsoever?

Can we build OT protocols assuming that the channel between the sender and the receiver is noiseless and say the corrupt party is computationally unbounded? The answer is no. Because it can be shown that if you have an OT protocol which gives you unconditional security against the

computationally unbounded sender or a computationally unbounded receiver, then using that OT protocol you can construct a secure 2 party protocol for computing the AND of 2 bits and that protocol will be secure even if 1 of the 2 parties is computationally unbounded. But we know that this is not possible, we have already proved it.

So, we already proved that there exist no unconditionally secure two party AND protocol and that automatically rules down the possibility of unconditionally secure OT protocol. And that shows that to design OT protocols, you definitely need to make assumptions, either assumptions regarding the physical channel which is there between the sender or the receiver or some assumption regarding the computing speed of or the computing resources of adversary.

(Refer Slide Time: 16:48)



So, let us see the proof of this implication, how exactly we can convert a secure OT protocol into a secure 2 party AND protocol? So, assume you are given an OT protocol; you are given a secure instantiation of OT protocol and say it is unconditionally secure. Namely, even if the sender is computationally unbounded, it does not learn the choice bit of the receiver or if the receiver is computationally unbounded, it does not learn anything about the other message of the sender.

It only learns the message which it is interested in. So assume we are given one such protocol, using that protocol we have to construct an AND protocol which allows the 2 parties Alice and Bob with bits x_1 and x_2 to securely learn the AND of their respective bits. And when I say secure

AND protocol, that means we want to ensure that if say x_2 is 0 - so we consider Bob is corrupt and if x_2 is 0 it anyhow knows that the function output will be 0.

But the interaction with Alice should not tell Bob whether x_1 was 0 or x_1 was 1, that is the security guarantee we require from the AND protocol. And similarly if Alice is corrupt and if our input is 0, it anyhow knows that the function output is going to be 0 but the interaction with Bob should not reveal whether the x_2 was 0 or x_2 was 1. So, assuming we have an unconditionally secure protocol for oblivious transfer, we will see that how Alice and Bob can use the OT protocol to the securely compute the AND of their respective bits?

And before going into that, let me write down the output of the OT functionality. So, the output of the OT functionality or the OT box or the OT protocol is the message m_b. And that message m_b can be expressed by this expression here - $m_b = (1 - b) \cdot m_0 \oplus b \cdot m_1$, and you can verify it. If indeed $b = 0$, then m_b should have been m_0.

So, if I substitute $b = 0$ in this expression, right, then this $b \cdot m_1$becomes 0 and $(1 - b)$ becomes 1, so m_b becomes m_0. And indeed if $b = 1$, I require that the output m_b should be equal to m_1. So, again let us substitute $b = 1$ in the expression. If $b = 1$, then $(1 - b)$ becomes 0, so $0 \cdot m_0$ is 0 and hence $1 \cdot m_1$ will be m_1. So, I can say that output generated by the OT is this.

So, now the question is that how Alice and Bob can invoke the OT protocol and securely compute the AND of x_1 and x_2? So, here is how they can securely compute the AND. So, let us Alice act as the sender, again Bob also could have acted as the sender, no issue, just for simplicity I am assuming that Alice acts as the sender and Bob acts as the receiver in an OT instantiation. That means they take this OT protocol, now let Alice use the messages 0 and x_1 as her pair of input messages for the OT.

So, she is acting as the sender for the OT, so she has to decide what the messages m_0 and m_1 are. And since Bob is acting as a receiver, he has to decide what should be the choice bit. So, he is using his choice bit as the bit x_2. Now, as per the OT protocol, they will interact. And finally Bob will obtain an output and now if you see the output expression of the OT.

You can see here, that the output that Bob receives is nothing but $x_1 \cdot x_2$. Why so? Because since m_0 is 0, so m_b a bit which the message which Bob is going to receive is $(1-b) \cdot m_0 \oplus b \cdot m_1$. But m 0 is any how set to 0, so it will vanish. So, it is only $b \cdot m_1$ but b is x_2 and m_1 is x_1, so that is what Bob will receive. Once Bob learns the output $x_1 \cdot x_2$, he can disclose it to Alice only.

So, that both of them learn $x_1 \cdot x_2$. Now, the claim is that if your OT protocol, the so called OT protocol that you assumed to exist is correct. In the sense it ensures that Bob receives the correct message m_b then when Bob and Alice participates in the AND protocol using the OT protocol as a sub protocol, they will obtain the correct output. So, the correctness of the OT protocol gets translated to the correctness of the AND protocol.

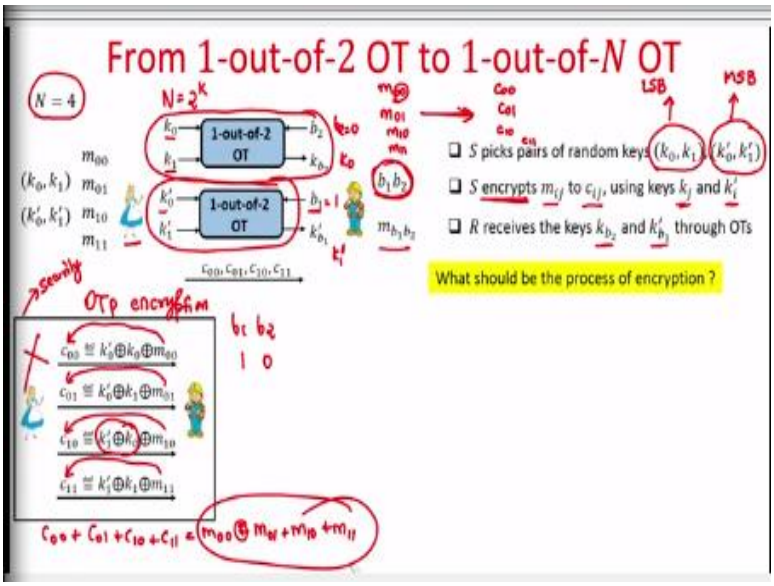
Now what about the privacy? Can I say the following? So, say let us consider the case when Bob is corrupt and say x_2 = 0 because a Bob is corrupt and if x_2 = 1, then anyhow from the function output it will learn x_1, so that is not a privacy breach. But we have to see that if Bob is corrupt and if x_2 is 0, then whether it learns whether x_1 is 0 or x_1 is 1. So, if x_2 is 0 if choice bit x_2 = 0, then Bob learns only the message 0.

It does not learn anything about x_1 and that will preserve the privacy of Bob. On the other hand, if Alice is corrupt and say x_1 is 0, because if x_1 would have been 1 and is Alice is corrupt, then again from the function output she will anyhow find out what is Bob is input. We have to consider the case when Alice is corrupt and x_1 is 0, if Alice is corrupt and x_1 is 0, she will not learn what exactly the choice bit of Bob is, and the choice bit of Bob is nothing but x_2.

So, that shows that if your OT protocol has the privacy guarantee that means if it ensures sender security against a corrupt receiver. And if it ensures receiver security against the corrupt sender, then that OT protocols privacy property gets translated to the privacy property of the AND protocol. And that shows that if your OT protocol was unconditionally secure, then so is the AND protocol.

But we know that there is no perfectly secure or unconditionally secure AND protocol. The OT protocol is there, but it is security is going to be cryptographic not unconditional.

(Refer Slide Time: 24:45)



Now let us see how we construct 1 out of N OT from 1 out of 2 OT. And again for demonstration purpose, I will assume $N = 4$ but the idea that I am going to discuss here will work for any N of the form2^k. So, that means now Alice has 4 private messages, I am indexing them with 00, 01, 10, 11. And Bob has an interest in one of those 4 messages, so it could be either the message with index 00 or the message with 01 or the message with index 10 or the message with index 11.

So, he has now a pair of choice bits b_1 and b_2. Depending upon the value of b_1 and b_2, the output of the OT protocol should be this. Now, the idea behind the protocol is the following. S, the sender, who in this case is Alice, is going to pick 2 pairs of random keys. And basically the key pair k_0, k_1 is for the LSB index. And the key pair k'_0, k'_1 is for the MSB index.

So, sender does not know whether the LSB index is 0 or 1, Alice does not know whether Bob's b_1 is 0 or 1 and Alice does not know whether Bob's b_2 is 0 or 1. So, with respect to each of the indices, Alice is picking a pair of keys. Because each bit position or the choice bit of Bob is going to take 2 possible values with respect to LSB as well as with respect to MSB Alice has picked a pair of keys.

And now the idea is that Alice does not know which of the 4 messages Bob is interested in, she cannot afford to ask Bob - "Bob please tell me what are your indices, I will transfer you those messages" - because that will breach Bob's privacy. So, somehow Alice has to communicate all the 4 messages, in such a way that Bob should be able to retrieve only one of the 4 messages. That automatically tells us that sender somehow has to encrypt all her 4 messages.

So, the way she is doing the encryption here is the following. She is going to encrypt the message m_{ij} to get a ciphertext c_{ij} and this happens using the keys k_j and k'_i. So, what I am saying here is that the message m_{00} will be translated to the ciphertext c_{00}, m_{01} will be translated to c_{01}, m_{10} will be translated to c_{10} and m_{11} will be translated to c_{11}. Now, depending upon the indices here, Alice will use the corresponding keys.

And now a receiver depending upon his choice bits, he should get back only the keys which he needs to recover back or decrypt back one of the 4 ciphertexts. And how he can do that? For that we can invoke 1 out of 2 OT instances, so this OT instance Bob is executing as a receiver and Alice's inputs for this OT instances are her pair of keys for the b_2 bit position or the LSB bit position.

So, for the LSB she has picked the keys k_0 and k_1, if b_2 is 0 Bob will get k_0, if b_2 is 1 he will get the key k_1. Similarly the other OT instance is used for the b_1 position. For the b_1 position Alice has picked a pair of keys k'_0, k'_1. If b_1 is 0 Bob will obtain k'_0, else he will obtain k'_1.

Now, the question is how exactly this mapping should happen from the messages to the ciphertext? It is not that we can use arbitrary form of encryption. So, let us first try OTP encryptions and see whether that satisfies our requirement or not. So, you can see here that m_{00} is translated to c_{00} by using the keys k_0 and k'_0 as OTP pads. Message m_{01} gets translated to c_{01} using the keys k_1 and k'_0 and similarly m_{10} got translated to c_{10}, m_{11} got translated to c_{11}.

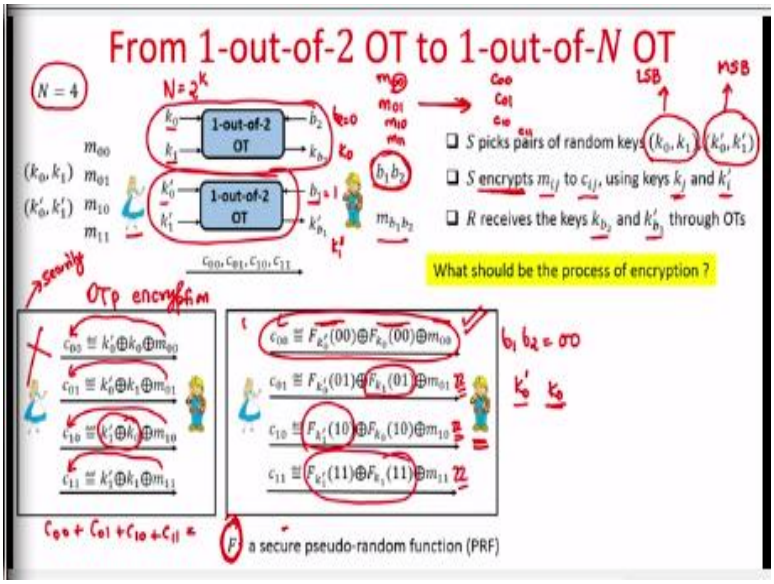
Suppose this is the way the encryptions of the 4 messages have been computed apart from the OT instances. Now, the question is does this ensure security? Of course, Bob will get the message that it is interested to, because say for instance if b_1 and b_2 are 1 and 0 respectively, then through the

first OT instance, if it is participates with $b_2 = 0$, then it obtains the key k_0. And in the other OT instance if it is participating with 1, then it gets k_1'.

And hence it will have both these keys to decrypt c_{10} and get back m_{10}. So, correctness is there for Bob, but does Bob learn anything additional in the process? And the answer is yes. If Bob takes the XOR of all the 4 ciphertexts here, then basically the effect of all the k's cancel out and he learns of and additional information namely the XOR of all the messages of all Alice messages, which is not supposed to be learned by Bob.

Because if at all sender security is achieved in the protocol, then Bob should *only* learn the message which it is interested in. But if this is the way the Alice messages are encrypted and Bob learns additional information, namely it learns the XOR of all the messages of Alice and that is why this is definitely not the way Alice should encrypt her messages.

(Refer Slide Time: 32:28)



So, let us see another method of encrypting and this is assuming we have a pseudorandom function. Again what is the pseudorandom function? I refer you to the NPTEL earlier course on foundations of cryptography. Basically it is a keyed function operated by a key and apart from the key we have an input also being provided to the function. And this function imitates like a true random function if the value of key is not known.

That means even if you know the input of the function, you cannot predict output of the function if the key is not given to you. And for most practical purposes this pseudorandom function can be instantiated using an AES. So, now what I am proposing here is the actual method of encryption is that let Alice use this key pairs k_0, k_1, k_0', k_1' as AES keys. And let the mapping from m_{ij} to c_{ij} be the same as we have discussed.

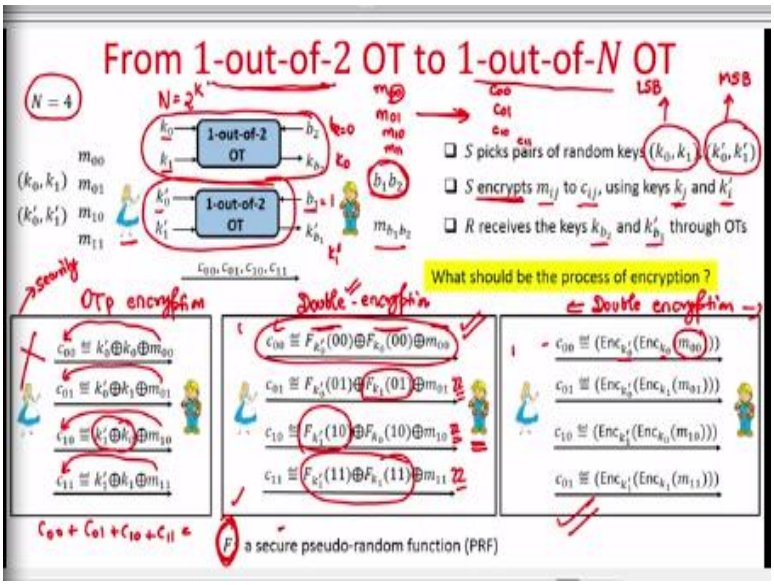
Except that instead of plane XORing the key is with the messages, we generate random pads and with XOR those with the message which needs to be encrypted. And now you can see that Bob cannot launch the same attack which we could launch in the previous case because now if we performs the XOR of all the 4 ciphertext that he receives, then the effect of the pads does not cancel out.

Because the pseudorandom function with different inputs will give you different pads and that is why they will not cancel out. And hence Bob will learn only the message that it is interested in. Say for instance, if b_1, b_2 was say 0,0, so of course Bob will have the key, through the OT instances it will receive the keys k_0' and k_0. So, it will have sufficient information to decrypt the first ciphertext, namely the ciphertext is c_{00}.

Because it will have the keys k_0' and k_0 and a description of pseudorandom function or say AES is publicly known. So it can regenerate this pad and XOR it with the ciphertext to c_{00} to get back the message m_{00}. But say let us try to analyze whether it can learn anything from the second ciphertext.

In the second ciphertext, there is a pad generated by the key k_1. But since the key k_1 is not available with Bob because through the OT instance, he has learned only k_0, it does not learn k_1. That is why it does not have sufficient information to decrypt; the second ciphertext. In the same way if I consider the third cipher text, right, for decrypting third ciphertext it requires this pad which is not available with Bob. And for the fourth ciphertext, it requires both these pads which are not available to him. And that ensures that Bob in this process does not learn anything about the other messages.

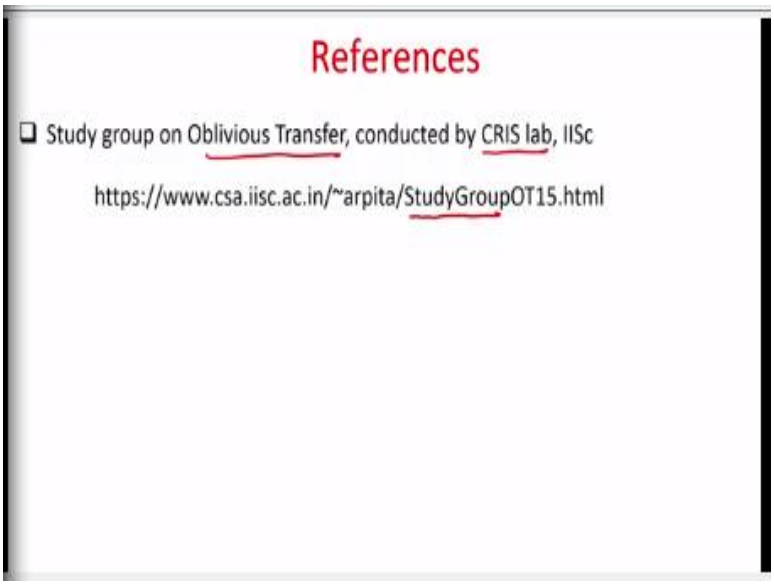
(Refer Slide Time: 36:02)



Well, there are several other forms of encryption as well Bob, Alice could have done the following, she could have done what we call us double encryption. Well, this PRF based construction is also a form of double encryption, why double encryption? Because we are actually using 2 pads to encrypt the message but it is not actually a form of double encryption. The actual form of double encryption will be this last form, where the message m_{00} is encrypted twice once using the key k_0. And then finally the result is used as a message to be encrypted again using another key k'_0 to get the final ciphertext to c_{00}.

So, this method is also a secure method, if your encryption scheme satisfies security property or if your pseudorandom function is actually a secure pseudorandom function, then this second method is also a secure method. But this first method of just blindly XORing the messages with the keys, it is not a secure message because that reveals additional information to Bob. So, that shows that if you are given a secure instantiation of 1 out of 2 OT, then you can get a secure instantiation of 1 out of N OT as well.

(Refer Slide Time: 37:26)



So, there are plenty of nice resources available online to learn more about oblivious transfer at Indian Institute of Science in this CRIS lab, cryptography information security lab. Some time back in 2015, we conducted a very interesting study group just on the topic of oblivious transfer, here is the link for the study group. So, there you can find several other resources and state of the art research papers related to oblivious transfer, thank you.