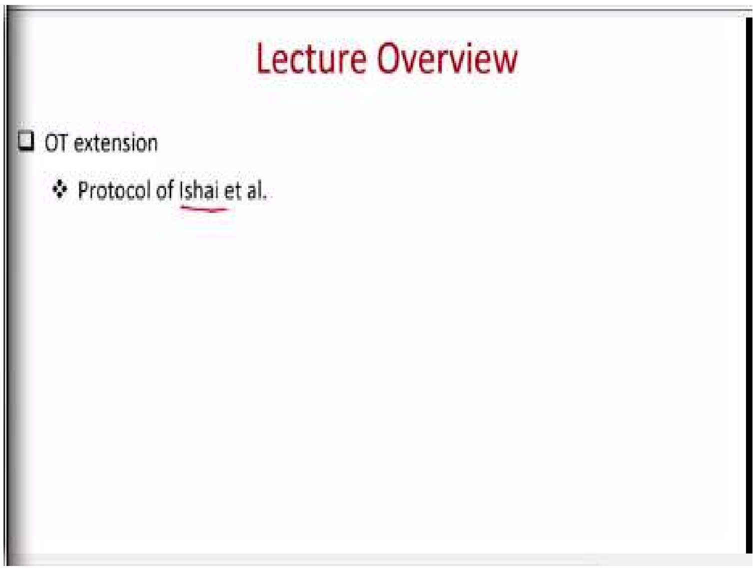
Secure Computation: Part I

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

Lecture-46
OT Extension

(Refer Slide Time: 00:32)

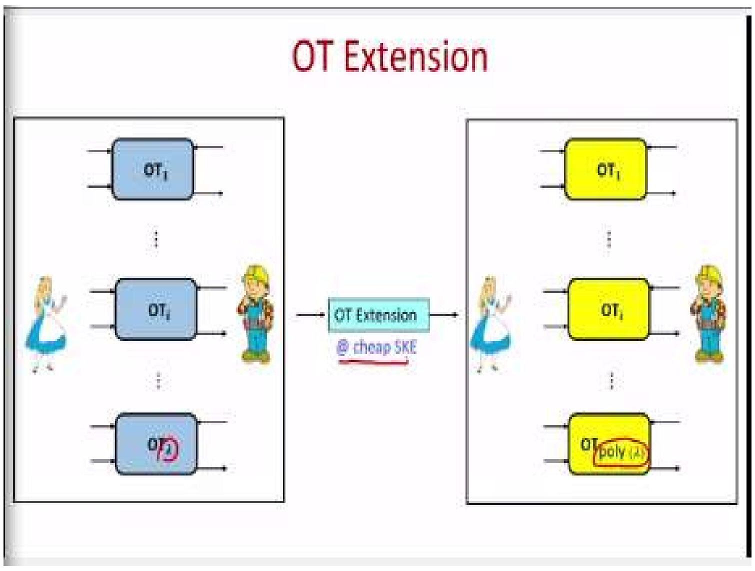


Lecture Overview

- OT extension
  - Protocol of Ishai et al.

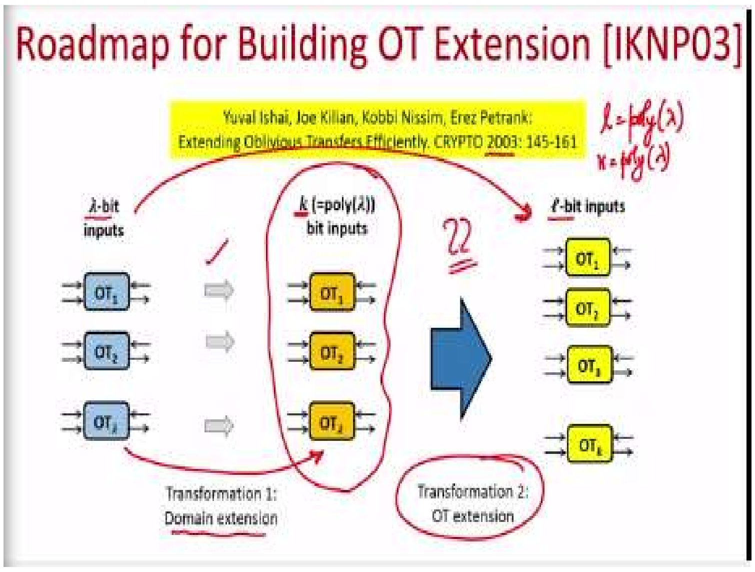
Hello everyone, welcome to this lecture. So, in this lecture we will see a very nice and beautiful protocol for oblivious transfer due to a Ishai et al.

(Refer Slide Time: 00:41)



So, just to quickly recall the goal of the oblivious transfer extension is the following. Suppose sender and receivers are going to be involved in polynomial of λ number of OT instances. We want to get the effect of those OT instances by just executing λ number of OT instances between Alice and Bob and performing some additional symmetric key operations.

(Refer Slide Time: 01:13)


Roadmap for Building OT Extension [IKNP03]
Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank: Extending Oblivious Transfers Efficiently. CRYPTO 2003: 145-161
$\ell = poly(\lambda)$
$k = poly(\lambda)$
λ-bit inputs
k (=poly(λ)) bit inputs
ℓ-bit inputs
OT_1
OT_2
OT_λ
OT_1
OT_2
OT_λ
OT_1
OT_2
OT_3
OT_k
Transformation 1: Domain extension
Transformation 2: OT extension

So, the protocol that I am going to discuss today it is called as IKNP OT extension protocol. A very beautiful protocol published in this paper in CRYPTO 2003. And let us see the roadmap for doing the OT extension. To do the OT extension we will 1st let Alice and Bob participate in λ number of OT instances, where the inputs of the sender are going to be λ bits and now assuming that sender and receiver have executed λ number of OT instances with λ bit inputs.

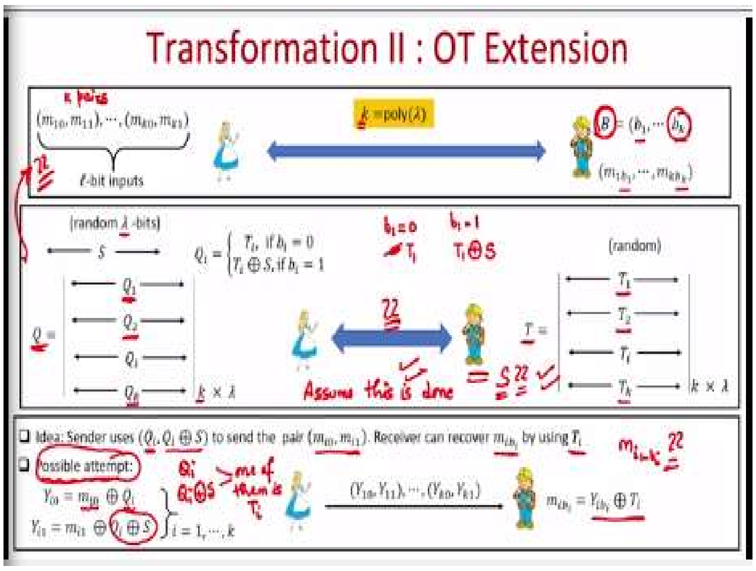
We will get the effect of λ number of OT instances but where the sender's inputs are now k bit long, where k is significantly large than λ. Namely, k is some polynomial function of your security parameter. And how do we go from here to here? We had already seen how to go from here to here, this is done through domain extension. We know that by running the domain extension technique which we had seen in the last lecture.

It is possible to get the effect of λ number of OT instances where the sender's input is k weights which is significantly large than the security parameter by actually running the same number of OT instances but on shorter inputs. So, we know how to get this thing done. But this is not the goal of OT extension, we are still running λ number of OT instances, we want to get the effect of polynomial of λ number of OT instances.

So, the actual OT extension protocol which is the transformation 2 is to take these λ number of OT instances with k bit inputs and transfer it or transfer in the sense transform it into the effect of k number of OT instances, where sender’s inputs are now l bit long. Of course l is going to be some polynomial function in your security parameter and now since k is also polynomial function of your security parameter.

Now you can see that we have actually transformed λ number of OT instances into k number of OT instances, where the inputs are now l bit long strings, so that is a roadmap. One part of the whole transformation, namely transformation 1 we had already seen, the question is how do we get this done? So, now our focus for today's lecture will be on transformation number 2.

(Refer Slide Time: 04:10)


Transformation II : OT Extension
k pairs
$(m_{10}, m_{11}), \cdots, (m_{k0}, m_{k1})$
ℓ-bit inputs
??
$k = \text{poly}(\lambda)$
$B = (b_1, \cdots b_k)$
$(m_{1b_1}, \cdots, m_{kb_k})$
(random λ-bits)
S
$Q_i = \begin{cases} T_i, \text{ if } b_i = 0 \\ T_i \oplus S, \text{ if } b_i = 1 \end{cases}$
$b_i = 0$
T_i
$b_i = 1$
$T_i \oplus S$
$Q =$
Q_1
Q_2
Q_i
Q_k
$k \times \lambda$
??
Assume this is done
$= S$??
(random)
$T =$
T_1
T_2
T_i
T_k
$k \times \lambda$
Idea: Sender uses $(Q_i, Q_i \oplus S)$ to send the pair (m_{i0}, m_{i1}). Receiver can recover m_{ib_i} by using T_i
$m_{i, 1-b_i}$??
Possible attempt:
$Y_{i0} = m_{i0} \oplus Q_i$
$Y_{i1} = m_{i1} \oplus Q_i \oplus S$
$i = 1, \cdots, k$
Q_i, $Q_i \oplus S$ one of them is T_i
$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$
$m_{ib_i} = Y_{ib_i} \oplus T_i$

So, what is the goal of transformation number 2? We have sender and receiver, so we make Alice as the sender, Bob as the receiver. Alice has k pairs of messages, they may not be random messages, they are some private inputs, private messages held by Alice. And in each pair the message length is l bit. Here k is some polynomial function of your security parameter λ. And now for each pair Bob has a choice picked, so from the 1st pair he is interested to receive the message with index b 1 and like that from kth pair he is interested to receive the message with index b_k. We want a protocol according to which Alice and Bob should interact.

And finally depending upon the indices in his vector B, from each pair he gets the corresponding messages. So, from the 1st pair, he gets the message with index b 1 and like that from the kth pair he gets the message with index b k. But for every pair he does not learn anything about the other message that is what will be the sender security. Whereas the receiver security will be that Alice should be completely unaware about what message from each pair got transferred to Bob. Namely, she should not learn anything about this choice vector B that is what we want to ensure.

Of course one way of doing this is that Alice and Bob run k number of OT instances, where for the 1st OT instance is used for the 1st pair, the 2nd OT instance is used for the 2nd pair and like that the kth OT instance is use used for the kth pair. We do not want to do that, we want to run only λ number of OT instances and still get this whole thing done. So, how this is done? Imagine for the moment that we have a magical mechanism to do the following.

How exactly this magical mechanism is implemented? That is the whole crux of this transformation number 2 but for the moment let us first try to understand the requirements of this magical mechanism, what exactly we want to get done through this magical mechanism? In this mechanism, Bob is going to pick a random matrix of values. Let me call that a random matrix at matrix T, it is a matrix of size k cross λ, so it has key number of rows and λ number of columns.

And Alice is going to pick a completely random vector of length λ bits, call it S. And somehow the magical mechanism allows the Alice and Bob to interact and get Alice matrix call it Q, which is also of same size as T. So, let me call the rows of Q as Q_1, Q_2, Q_i, Q_k. And how exactly the rows of this Q matrix is related to the rows of T matrix? Well, it is related depending upon the entries in Bob's choice vector, so remember Bob has a choice vector B.

Now, let us see the relationship between Q_1 and T_1, if $b_1 = 0$, then Q_1 will be same as T_1. If $b_1 = 1$, then q_1 will be T_1 blinded with S. And in the process Alice should not learn what exactly is the value of Q_1, namely she should not learn anything about the value of b_1. She will be receiving a row of length λ bits that is the Q_1, but whether it is T_1 or whether T_1 blinded with S, she should not learn. Similarly Q_2 will be either equal to T_2 or T_2 blinded with S depending upon whether b_2 is 0 or 1.

And in the same way Q_k will be either T_k or T_k blinded with S depending upon whether the last choice bit b_k is 0 or 1. That is what is the whole purpose of this magical mechanism, how exactly this is implemented? Forget about it. That is the whole crux of OT extension or transformation to which we are going to see next. But assume for the moment, it is possible for Alice and Bob to do this or to get this done.

And Bob should not learn anything about S in the whole process that is also is important. So, Alice should not learn anything about the choice vector B. That means whether the rows of the matrix Q or the T rows or whether the T rows blinded with S whereas a corrupt Bob should not learn anything about S. That is what we want to ensure through this magical mechanism. Assume for the moment this is done, assume this is done.

Now the question is how exactly we get the actual OT done? Remember, sender has to somehow send the k pair of messages to Bob and it should be done in such a way that for each pair Bob should get back only one of the messages depending upon his corresponding choice bit. And for doing that we want this interaction between Alice and Bob, so that Alice gets this Q vector. Now the idea here will be the following.

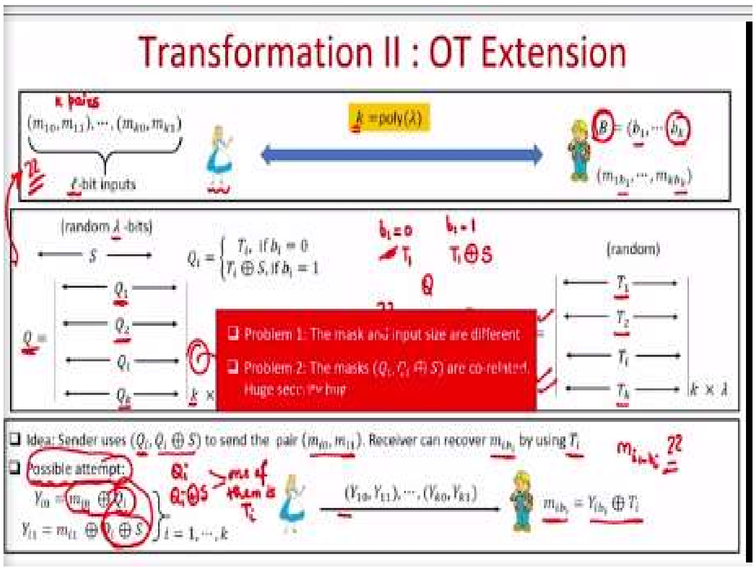
To get the actual oblivious transfer done, Alice has to use somehow the ith row of Q and ith row of Q XORed or blinded with S, she has to use these 2 piece of information to transfer the 2 messages in her ith pair. And that is why the number of rows in the matrix here is k because she has k pair of messages to transfer. For each pair of message she is going to use a row of the Q matrix and a same row blinded with S.

And she has to use those 2 piece of data to send her messages for ith pair and it should be done in such a way that Bob should get back only the message with index b_i for the ith pair by just using the ith row from his T matrix, that is a whole idea. So, based on whatever we have seen in the oblivious transfer protocols till now, you might be tempted to propose the following. That let Alice do the following, for each of a rows i ranging from 1 to k, encrypt the 0th message in ith pair using Q_i.

And encrypt the other message in the ith row using Q_i XORed with S as the pattern. Now we know that Q_i and Q_i XORed with S, one of them is definitely T_i, definitely we know this. Alice does not know whether it is Q_i which is T_i or whether it is Q_i XORed with S which that is T_i. Because she does not know the value of index b_i but she knows that ok one of these entries, namely Q_i and Q_i XORed with S is T_i. So, that is why she is tempting to encrypt the 0th message and the ith pair with Q_i and the message with index 1 with the paired Q_i XORed with S.

And a hope is that Bob should be able to recover back the message with index b_i in ith pair by unmasking T_i because he has T_i. And Alice will hope that since Bob does not have S, because Bob has not learned the value of S in the magical mechanism. He cannot figure out what is the message with index 1 - b_i in the ith pair. Because that will require the knowledge of S and Alice will think that ok, Bob does not know S. So, you might be tempted to propose the following way of sending the messages from Alice to Bob. Assuming that this magical mechanism of communicating the rows of T in an obvious way to Alice has been done but there is a slight problem here.

(Refer Slide Time: 13:52)



Transformation II : OT Extension

k pairs

$(m_{10}, m_{11}), \cdots, (m_{k0}, m_{k1})$

ℓ-bit inputs

$k = \text{poly}(\lambda)$

$B = (b_1, \cdots, b_k)$

$(m_{1b_1}, \cdots, m_{kb_k})$

(random λ-bits)

S

$Q_i = \begin{cases} T_i, \text{ if } b_i = 0 \\ T_i \oplus S, \text{ if } b_i = 1 \end{cases}$

$b_i = 0$ T_i $b_i = 1$ $T_i \oplus S$

Q

$Q = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_i \\ Q_k \end{bmatrix} k \times \lambda$

(random)

$\begin{bmatrix} T_1 \\ T_2 \\ T_i \\ T_k \end{bmatrix} k \times \lambda$

- Problem 1: The mask and input size are different
- Problem 2: The masks $(Q_i, Q_i \oplus S)$ are co-related. Huge security bug

- Idea: Sender uses $(Q_i, Q_i \oplus S)$ to send the pair (m_{i0}, m_{i1}). Receiver can recover m_{ib_i} by using T_i
- Possible attempt:

$\left.\begin{array}{l} Y_{i0} = m_{i0} \oplus Q_i \\ Y_{i1} = m_{i1} \oplus Q_i \oplus S \end{array}\right\} i = 1, \cdots, k$

$\left.\begin{array}{l} Q_i \\ Q_i \oplus S \end{array}\right\}$ one of them is T_i

$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$

$m_{ib_i} = Y_{ib_i} \oplus T_i$

$m_{i, \ell}$??

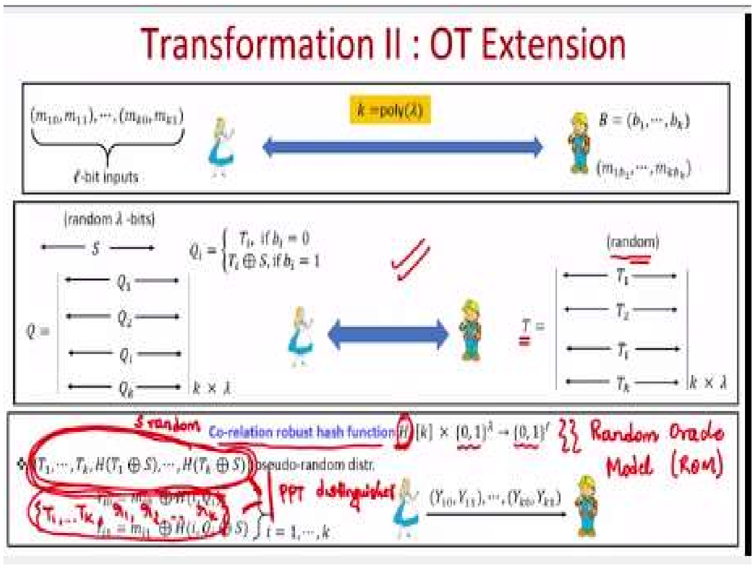
Problem number 1 is the following. What are the length of the messages available with Alice? They are all l bit long. But what is the length of the rows of the Q matrix and the T matrix? They are only λ bit long. So, you have an incompatibility in the size. How can you mask l bit long string using a λ bit string? That is not possible. One way of getting around this problem is that ok, let Alice use a pseudorandom generator on the rows of Q and expand it to get a suitable pad and then do the actual masking of our messages and get it transferred to Bob, fine, that might solve your problem.

But a bigger problem is this problem number 2, there is a huge security bug involved here. And the security bug that is involved here is the following that is because of the following. If you consider both the messages in the ith pair, they are actually masked using related pads; they are not completely independent pad. One of the messages is XORed with Q $_i$, another message is Q $_i$ XORed with S and it is the same as which is used across all the pairs of Alice.

If S would have been independent for each of the pairs of Alice, then again no problem but it is the same as which is used across all the pairs of messages which Alice wants to communicate to Bob. So, if you say for instance, take a case where Alice has just 2 pairs of messages and if it is the same as which is used and Alice uses the same as to transfer both pairs of our messages. Then a very simple attack which Bob can launch is the following.

If we XORes the encryptions of the all the messages that he is receiving, then the effect of S and S will cancel out. And Bob might end up learning the XOR of all the messages of Alice, apart from of course the message that he is interested from each pair and that is a security breach here. So, that is why we might be tempted to use this idea of using the rows of the matrix Q to encrypt the messages available with Alice, doing it in the naive way will not work.

(Refer Slide Time: 16:30)


Transformation II : OT Extension
$(m_{10}, m_{11}), \cdots, (m_{k0}, m_{k1})$
ℓ-bit inputs
$k = \text{poly}(\lambda)$
$B = (b_1, \cdots, b_k)$
$(m_{1b_1}, \cdots, m_{kb_k})$
(random λ-bits)
S
$Q_i = \begin{cases} T_i, \text{ if } b_i = 0 \\ T_i \oplus S, \text{ if } b_i = 1 \end{cases}$
(random)
$Q =$
Q_1
Q_2
Q_i
Q_k
$k \times \lambda$
$T =$
T_1
T_2
T_i
T_k
$k \times \lambda$
S random
Co-relation robust hash function $H: [k] \times \{0,1\}^\lambda \rightarrow \{0,1\}^\ell$
Random Oracle Model (ROM)
$(T_1, \cdots, T_k, H(T_1 \oplus S), \cdots, H(T_k \oplus S))$ pseudo-random distr.
PPT distinguisher
$\{T_1, \ldots T_k, r_1, r_2, \ldots r_k\}$
$Y_{i0} = m_{i0} \oplus H(i, Q_i)$
$Y_{i1} = m_{i1} \oplus H(i, Q_i \oplus S)$
$i = 1, \cdots, k$
$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$

But we still want to retain this idea because once this magical mechanism is done, this obviously communicating the rows of T to Alice. Then almost we are done here, the only question is that how exactly the rows of Q can be used to mask the pair of messages which are available with Alice. And here we use a very nice technique; we use a hash function which we call a co-relation robust hash function.

And his hash function takes a pair of inputs, the first entry is an index in the range 1 to k and the second entry is a λ bit string and it produces a string of length l bits. Why it is called co-relation robust hash function? By co-relation robust hash function, we mean that the joint probability distribution of this 2 k number of entries is pseudorandom from the viewpoint of a computationally bounded adversary or distinguisher.

That means if I want to compare this distribution with another distribution, where the 1st k entries are random. So, the 1st k entries are T_1 to T_k, which are the rows of this matrix T and they are picked uniformly at random. So, the rows T_1, T_2, T_k they are all uniformly random λ bit strings. So, in the other distribution which I am considering here, the 1st k entries are also the rows of T, so they are uniformly random.

But the last entries, last k entries they are also see random λ bit strings r_1, r_2, r_k. By co-relation robust hash function I mean that if we consider a probabilistic polynomial time distinguisher who is either given a sample of type this where the 1st k entries are random and last k entries they are generated by running the hash function with respect to T_1 to T_k and a fixed S, where the S is not fixed as are uniformly random S, but S is not known of course.

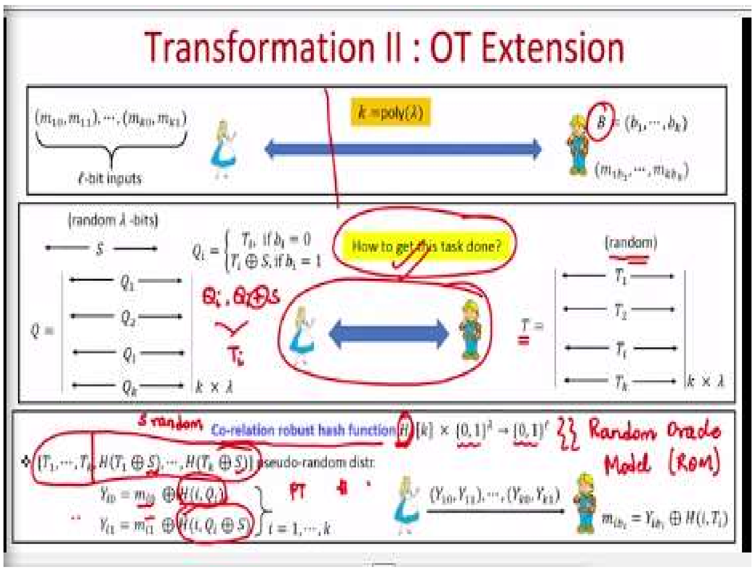
Because if S is also given to the distinguisher then he can figure out whether he is seeing a sample of type this or a sample of type this, where all the 2 k entries are uniformly random. So, S is random and private and there is a polynomial time distinguisher who is either given a sample where all the 2 k entries are uniformly random λ bit strings or he is given a sample of type half random and half generated by running the hash function but with respect to a random S which is not aware.

Then by correlation robust hash function we mean that from the viewpoint of a probabilistic polynomial time distinguisher both these distributions are almost identical. He cannot figure out whether he is seeing a sample of type half random followed by other half generated by running the hash function or a sample where all the entries are uniformly picked at random. Now the question is how do we built this such correlation robust hash function.

So, it turns out that if we assume a random oracle model then again people who have taken the course on foundations of cryptography. They know what do we mean by random oracle model? It is basically a strong model where we assume very strong properties from a hash function. Basically we assume that it is behaving like a uniformly random function and a interaction with the hash function is modeled through oracle queries.

And then in that model we prove certain nice properties for the cryptography constructions which uses the hash function. So, if we assume random oracle model then we can take any hash function in the random oracle model and we can get this notion of co-relation robust hash function. But for practical instantiations you can take the so called secure hash functions that we are available with say the sha functions and so on. So, assuming such a co-relation robust hash function is available, here is how Alice is going to encrypt the messages using the rows of the matrix Q.

(Refer Slide Time: 21:28)



Transformation II : OT Extension

$\underbrace{(m_{10}, m_{11}), \cdots, (m_{k0}, m_{k1})}_{\ell\text{-bit inputs}}$ $k = \text{poly}(\lambda)$ $B = (b_1, \cdots, b_k)$ $(m_{1b_1}, \cdots, m_{kb_k})$

(random λ-bits) S

$Q_i = \begin{cases} T_i, \text{ if } b_i = 0 \\ T_i \oplus S, \text{ if } b_i = 1 \end{cases}$

How to get this task done?

$Q = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_i \\ Q_k \end{bmatrix}$ $k \times \lambda$

$Q_i, Q_i \oplus S$ T_i

(random) $T = \begin{bmatrix} T_1 \\ T_2 \\ T_i \\ T_k \end{bmatrix}$ $k \times \lambda$

S random Co-relation robust hash function $H : [k] \times \{0,1\}^\lambda \rightarrow \{0,1\}^\ell$ Random Oracle Model (ROM)

$\{T_1, \cdots, T_k, H(T_1 \oplus S), \cdots, H(T_k \oplus S)\}$ pseudo-random distr.

$\left.\begin{array}{l} Y_{i0} = m_{i0} \oplus H(i, Q_i) \\ Y_{i1} = m_{i1} \oplus H(i, Q_i \oplus S) \end{array}\right\} i = 1, \cdots, k$ PT

$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$

$m_{ib_i} = Y_{ib_i} \oplus H(i, T_i)$

Suppose, we take the ith pair, in the ith pair, we have the messages with index 0 and index 1. For encrypting the message with index 0, she computes a pad by computing the output of the hash function on input pair i and Q $_i$. And to encrypt the other message in the ith pair, she again computes a pad but now with input pair i and XOR of Q i and S. And that is how she encrypts all the pairs.

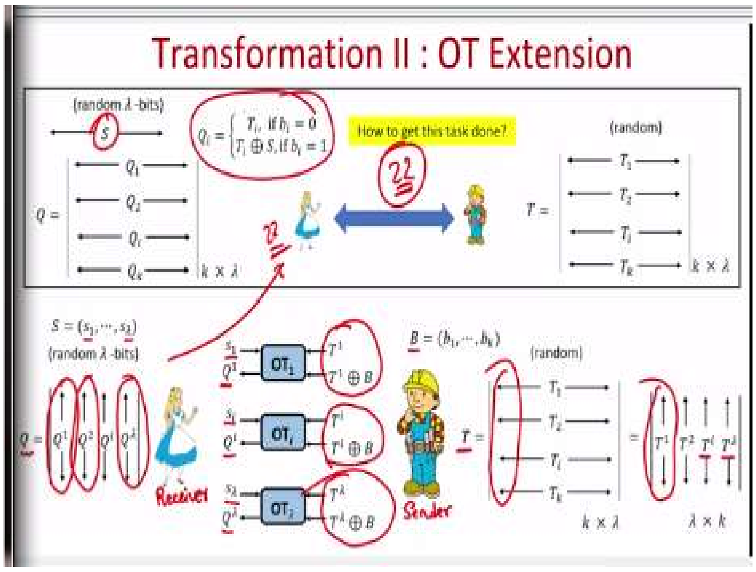
And the idea here is that as we said earlier that Alice knows that one of the entries Q $_i$ are Q $_i$ XORed with S is going to be T $_i$, which of them is T $_i$? She does not know, Bob knows the T $_i$, so he can unmask the ciphertext with index b $_i$ from the ith pair. But he cannot find out the other message because of this notion of co-relation robust hash function. So, that means, even if he is seeing the encryption of all the messages, where it is the same as which is used to prepare the pad unlike the previous mechanism, where it was directly the ith row and a $_i$th row XORed with S which were used as pad and the same S was involved in all the encryptions.

We are not directly using Q_i and XOR of Q_i and S to mask the messages. We are actually running the hash function to generate the pads and since the value of faces not known to Bob. Because during this magical interaction, which we are assuming to exist, we assumed that Bob does not know anything about S. And if he does not know anything about S, then the co-relation robust hash function assumption ensures that all the pads which are used by Alice.

So, Alice has used these 2 k pads in some order, in what order? She has used these pats Alice is not aware because she is not aware of Bob's choice vector. But she has used the pads T_1 to T_k and hash outputs of T_1 XORed with S, T_2 XORed with x, T_k XORed with S in what order? We do not know. Now Bob will be knowing only the T_1 to T_k components of this vector of pads but he cannot figure out anything about the remaining k components of the pats because of this co-relation robust hash function assumption.

And hence he cannot unmask the remaining k messages that are held with Alice and that ensures roughly Alice’s security. So, now everything boils down to how exactly we get this magical mechanism done and that is what is the whole crux of your transformation number 2.

(Refer Slide Time: 24:41)



So, now we will focus our attention on this magical mechanism how exactly this matrix T gets obliviously transferred to Alice, without Alice learning anything about Bob's choice factor B? And Bob not does not learning anything about this uniformly random S which Alice has picked and this is done as follows. So, till now no oblivious transfer protocol has been involved. Assuming that magical mechanism has been done, the actual transfer of messages from Alice to Bob just involves computing pats by running the hash function and XORing the pats with the Alice messages and sending to Bob.

So, you might be wondering where exactly oblivious transfer instances are involved, they are involved in this magical mechanism. So, as I said earlier, Bob picked this vector of random values, T, it has k rows and λ columns. I can interpret the same matrix T in terms of columns as follows. I can imagine that, let me call the 1st column of this T matrix as T^1, let me call the ith column of this T matrix as T^i and let me call the λth column as the column T^λ.

So, I am just observing, I am just viewing the matrix T in terms of it is column, I am not computing the transpose of the matrix T, remember. I am just viewing the entries in T in terms of it is column, I am calling those columns as T^1, T^i, T^λ. There will be λ such columns and the length of each column will be k. Now Alice and Bob are going to participate in λ number of OT instances. And it is the bob who is going to play the role of sender and Alice who is going to play the role of receiver.

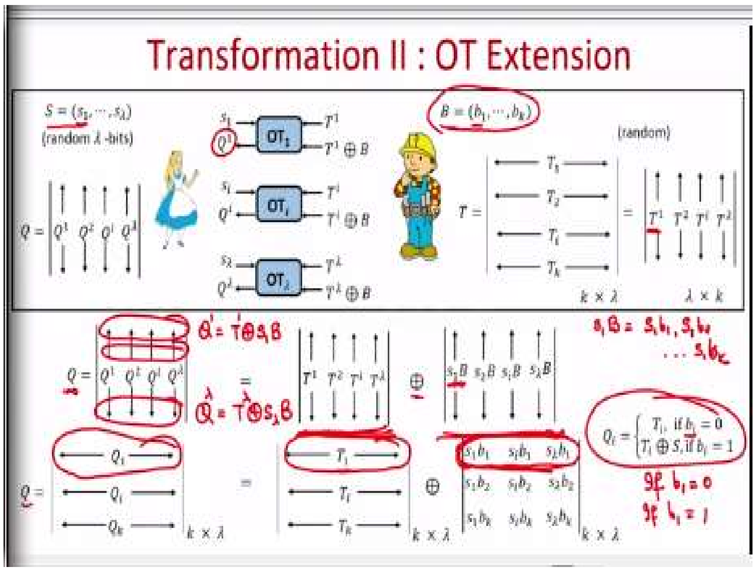
Now this might be confusing because our actual goal was to let Alice play the role of sender, she had k pair of messages and from each pair she wanted to communicate 1 message to Bob and Bob was the receiver. But for getting it done, we require this magical mechanism to be executed. And for this magical mechanism, it is the Bob who is going to play the role of sender and Alice who is going to play the receiver.

Now let us see the inputs of Bob in these OT instances. In the 1st OT instance his messages are going to be the 1st column of this T matrix and the 1st column XORed with his choice vector B, both of them are k bit long. In the same way if I consider the ith instance, it is the 0th message for the ith instance which Bob is feeding is his ith column of the T matrix. And other message is the ith column XORed with his choice vector.

And like that, if I consider the last OT instance these will be the inputs. And Alice will participate with the individual bits of the S vector as her choice bits. Now, based on the correctness property of the oblivious transfer protocol through each of the OT instance Alice is going to recover or receive the outputs Q^1, Q^2, Q^i, Q^λ.

Now let me arrange them column wise and view the entire matrix of values as the matrix Q and my claim is that this magical mechanism has been achieved. That means, by doing this oblivious transfer instances Alice now has the Q matrix and a Q matrix indeed satisfies this property. So, let us analyze that.

(Refer Slide Time: 29:19)



Transformation II : OT Extension
$S = (s_1, \cdots, s_\lambda)$
(random λ-bits)
$B = (b_1, \cdots, b_k)$
(random)
OT_1
OT_i
OT_λ
T^1
$T^1 \oplus B$
T^i
$T^i \oplus B$
T^λ
$T^\lambda \oplus B$
$Q = Q^1\ Q^2\ Q^i\ Q^\lambda$
$T = T_1, T_2, T_i, T_k$
$= T^1\ T^2\ T^i\ T^\lambda$
$k \times \lambda$
$\lambda \times k$
$Q^1 = T^1 \oplus s_1 B$
$Q^\lambda = T^\lambda \oplus s_\lambda B$
$s_1B\ s_2B\ s_iB\ s_\lambda B$
$s_iB = s_ib_1, s_ib_2, \ldots s_ib_k$
$Q_i = T_i$, if $b_i = 0$; $T_i \oplus S$, if $b_i = 1$
If $b_i = 0$
If $b_i = 1$

Let us analyze whether indeed Alice got what we wanted to get her? So, the matrix which Alice got at her end consists of this λ columns, column number Q^1, Q^i, Q^λ. And what can I say about the 1st column that she has received, Q^1? It is either the column T^1 or it is the column T^1 XORed with B depending upon what was the 1st choice bits in Bob's choice vector B.

So, I can say that Q^1 is nothing but T^1 XORed with S_1 times B. Sorry for the mistake I said it incorrectly, depending upon what is Alice choice bit S_1, she either gets the 1st column of T or the

1st column of T XORed with Bob's choice vector B, that is the right statement. In the same way, if I focus on the last column that Alice received, it is either the last column of Bob's matrix T or the same column XORed with B provided the choice bit of Alice was 1.

So, I can say that this relationship holds. So, this relationship I am taking out the XOR outside and I can say that the Q matrix which Alice received at her end is same as the T matrix of Bob XORed with this matrix. And now, what I am doing is, I am just reinterpreting the columns which Alice has received and viewing it in terms of rows. Again, I am not computing the transpose of the matrix Q, I am not making the columns as the row and so on.

So, I am just focusing now the entries of the Q matrix in terms of it is row. So, I can say that the 1st row is Q_1, the ith row is Q_i and a kth row is Q_k. And in the same way, I am now interpreting the Bob's matrix T in it is original form in terms of rows. And now what can I say about the 1st column here, S_1 times B? Remember, B is this whole choice vector, so S_1 times the choice vector B is going to be component wise S_1 times b_1, S_1 times b_2 and like that S_1 times b_k, so that is what is S_1 times B.

And like that, the last column here will be S_λ times B, that is the relationship between the Q matrix and the T matrix. And now you focus on the ith row of the Q matrix which Alice has received. I can say that if b_i would have been 0, then Q_i would be same as T_i otherwise it will be T_i XORed with S. So, say for instance, if I take the say the 1st row of Q matrix, the 1st row of the Q matrix is Q_1, it is T_1 XORed with the 1st row here.

By the first row here everywhere we have b 1 times something, so b_1 times S_1, b_1 times S_2, b_1 times S_i, b_1 times S_λ. So, if b_1 would have been 0 then this whole 1st row would have become 0, in which case Q_1 would have been T_1. If b_1 would have been equal to 1, then this 1st row would have been same as the vector S. And hence Q 1 would have been T_1 XORed with S and the same logic holds for the other rows as well.

And then we can say that this is the property which holds with respect to the ith row and this is what precisely we wanted to do. We wanted to somehow transfer either the ith row of T or the ith row XORed with Alice vector S, without letting Alice know whether she got the ith row of T or the ith row XORed with S and that is what got transferred to Alice. And hence the magical mechanism has been achieved. Now of course we have to analyze the security of this magical mechanism, how much computation it involves and so on, which we are going to do in the next lecture. Thank you.