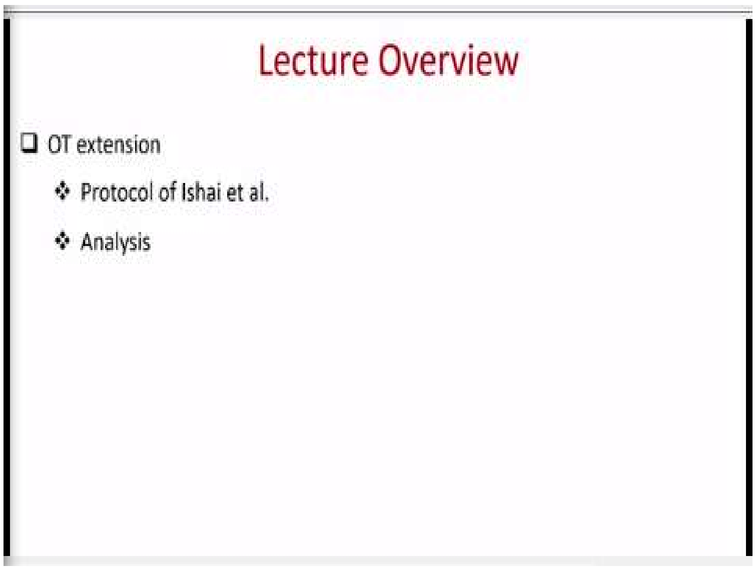
Secure Computation: Part I

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

Lecture-47
Analysis of IKNP OT Extension

(Refer Slide Time: 00:30)

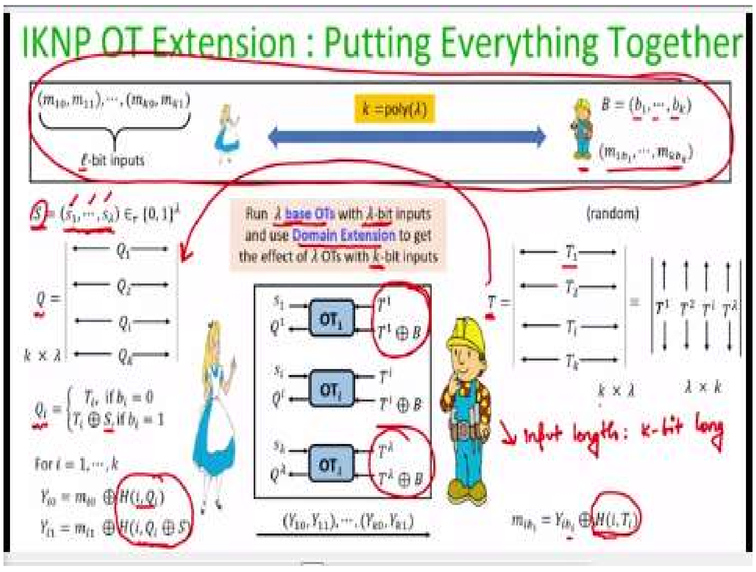


Lecture Overview

- OT extension
  - Protocol of Ishai et al.
  - Analysis

Hello everyone, welcome to this lecture. So, in this lecture we will continue our discussion on the OT extension protocol of Ishai et al. And we will analyze that protocol.

(Refer Slide Time: 00:40)


IKNP OT Extension : Putting Everything Together
$(m_{10}, m_{11}), \cdots, (m_{k0}, m_{k1})$
ℓ-bit inputs
$k = \mathrm{poly}(\lambda)$
$B = (b_1, \cdots, b_k)$
$(m_{1b_1}, \cdots, m_{kb_k})$
$S = (s_1, \cdots, s_\lambda) \in_r \{0,1\}^\lambda$
Run λ base OTs with λ-bit inputs and use Domain Extension to get the effect of λ OTs with k-bit inputs
(random)
$Q =$
Q_1
Q_2
Q_i
Q_k
$k \times \lambda$
$Q_i = \begin{cases} T_i, & \text{if } b_i = 0 \\ T_i \oplus S, & \text{if } b_i = 1 \end{cases}$
For $i = 1, \cdots, k$
$Y_{i0} = m_{i0} \oplus H(i, Q_i)$
$Y_{i1} = m_{i1} \oplus H(i, Q_i \oplus S)$
s_1
Q^1
OT_1
T^1
$T^1 \oplus B$
s_i
Q^i
OT_i
T^i
$T^i \oplus B$
s_λ
Q^λ
OT_λ
T^λ
$T^\lambda \oplus B$
$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$
$T =$
T_1
T_2
T_i
T_k
$k \times \lambda$
$T^1 \; T^2 \; T^i \; T^\lambda$
$\lambda \times k$
input length: k-bit long
$m_{ib_i} = Y_{ib_i} \oplus H(i, T_i)$

So, let us first put everything together regarding the transformation 2 that was involved in the oblivious transfer extension of IKMP. And we will couple it with the transformation number 1 and then see the complete picture. So, we wanted to get the following done, Alice had k pair of messages, each message was of length l bit. And for each pair Bob has a choice and we want to ensure that depending upon his choice bit for each pair he should get a corresponding messages.

And k here is some polynomial function of your security parameter λ, we do not want to run a full-fledged k instances of OT but we want to run only λ instances of OT. How this can be done? Even though Bob is the receiver here, we actually let Bob play the role of sender for the following inputs. So, he picks a random matrix of bits call it T which has k times λ number of random bits.

So, in terms of rows which we have k rows each of size λ bits and in terms of columns we have λ columns each of length k bits. So, the superscript interpretation is the column interpretation, subscript interpretation is the row interpretation. And Alice picks a random vector of choice bits S_1 to S_λ. And let Alice and Bob participates in λ number of OT instances. In these OT instances Bob's inputs are the columns of his T matrix and those columns XORed with his choice vector B.

Whereas, Alice inputs are the respective entries from her S factor and depending upon the entries in the choice vector B, we saw in the earlier lecture that Alice will receive the matrix Q where ith row of Alice matrix Q will be either the ith row of the matrix T or the ith row of the matrix T XORed with the full vector S here. So, these are the λ OTs which Alice and Bob have to execute, they are often called as the base OTs.

And here we can use domain extension as well to make it more efficient, why domain extension? So, what are the input length of Bob's messages? They are all k bit long. And how many such OT instances have to be executed λ such OT instances. We know that using domain extension Bob can actually run λ number of OT instances with only λ bit inputs. Namely, Bob has to do, Bob has to pick λ pair of PRG seeds and participates with those pairs of PRG seeds seats and let Alice from each OT instance receive only one of the PRG seeds.

And then Bob can pad his Actual messages of this base OTs using the outcomes of the PRG'S. And based on what PRG seed Alice has received from each OT instance, she will be able to recover back only one of the messages for these base OTs. So, actually even though in the base OTs Bob's messages are of length k bits, we do not want Bob to run actually a full-fledged OT even with k bit inputs, he can run these λ instances with λ bit random PRG seed pairs and get the effect of this base OTs.

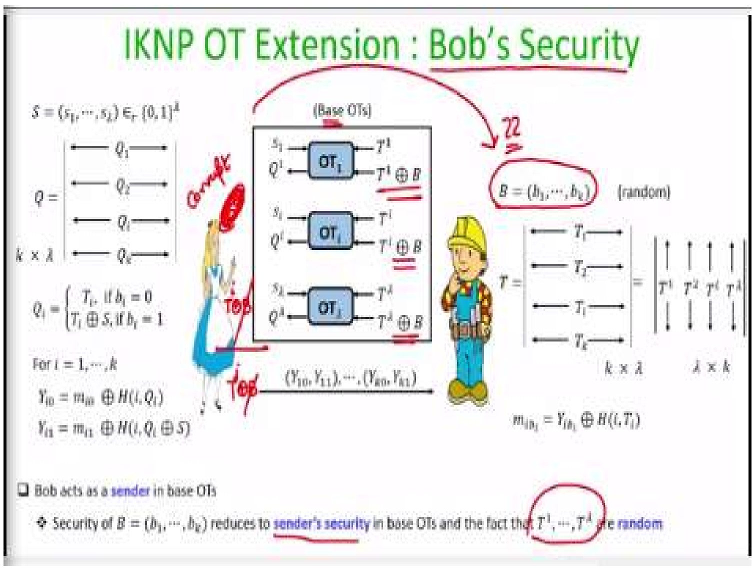
And that will ensure that somehow this matrix of values T gets obliviously transferred to Q. So, even though Bob is the overall receiver of this final OT, he is acting as the sender while executing this magical mechanism and transferring his matrix T in an oblivious fashion to Alice. And Alice will not be knowing whether she is receiving the rows of T's or whether she is receiving the rows of T XORed with her own vector S.

And now once she gets the rows of the matrix Q, she has to prepare pads using the ith row for her ith pair of messages. So, for the ith pair she is using the entry i Q_i and i Q_i XORed with S, one of them is T_i which one she is not aware. And using the outcomes of the co-relation robust hash function, she mask her messages in the ith pair. And now from each pair Bob will be able to recover back the appropriate message by computing the corresponding pad.

Because he knows that to recover the message with index b_i from the ith pair the row T_i has to be used to compute the pad. So, he can compute the same pad and XOR it back, so that is the entire OT extension protocol. So, to get the effect of k number of OT instances we have to just run λ number of base OT's and that to only λ bit random inputs. And that is the whole power, whole magic of this OT extension protocol.

In practice k is significantly large compared to λ, k is could be order of millions or billions or λ could be of order say 2048 and so on. So, by running only a small number of OTs and doing the public key operations which are involved in only those λ number of OT instances, we are actually getting the effect of million number of OT instances without running so many public key operations.

(Refer Slide Time: 08:09)



IKNP OT Extension : Bob's Security

$S = (s_1, \cdots, s_\lambda) \in_r \{0,1\}^\lambda$

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

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

For $i = 1, \cdots, k$

$Y_{i0} = m_{i0} \oplus H(i, Q_i)$

$Y_{i1} = m_{i1} \oplus H(i, Q_i \oplus S)$

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

$T = \begin{bmatrix} \leftarrow T_1 \rightarrow \\ \leftarrow T_2 \rightarrow \\ \leftarrow T_i \rightarrow \\ \leftarrow T_k \rightarrow \end{bmatrix} = \begin{bmatrix} \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ T^1 & T^2 & T^i & T^\lambda \end{bmatrix}$

$k \times \lambda$ $\lambda \times k$

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

- Bob acts as a sender in base OTs
  - Security of $B = (b_1, \cdots, b_k)$ reduces to sender's security in base OTs and the fact that $T^1, \cdots, T^\lambda$ are random

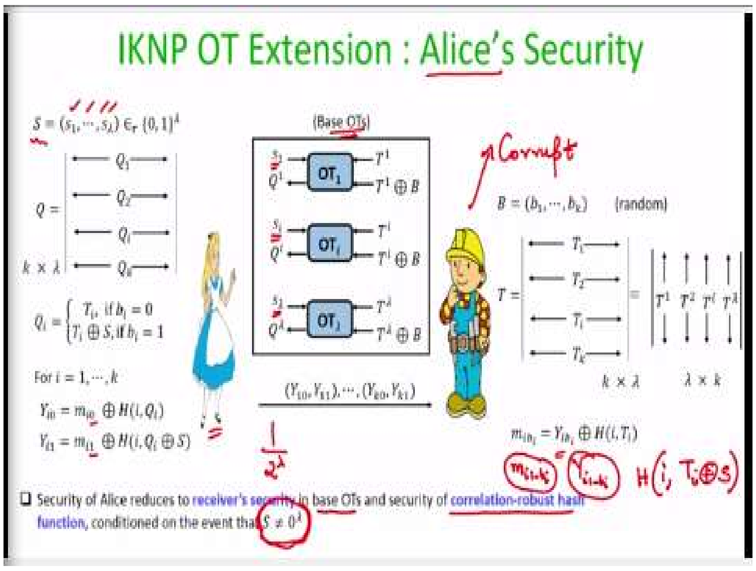
(Base OTs)
s_1 Q^1 OT_1 T^1 $T^1 \oplus B$
s_i Q^i OT_i T^i $T^i \oplus B$
s_λ Q^λ OT_λ T^λ $T^\lambda \oplus B$
$(Y_{10}, Y_{11}), \cdots, (Y_{k0}, Y_{k1})$
corrupt
TOB
TOB
??

So, now let us quickly analyze the security of this entire OT extension protocol whether Bob's security is maintained or not, Alice's security is maintained or not. So, for Bob security we have to argue that nothing about his choice vector B is learned by a corrupt Alice. We have to argue whether she learns anything about this choice vector. So, what exactly she learns through interaction about the choice vector of Bob? Well, the choice vector of Bob is involved only in this base OTs, but they are not directly used as inputs by Bob.

He is masking those inputs with random columns of the T matrix. And each of the instance even though it is the same B which is involved, it is XORed with independent pads because in each instance an independent column of the full matrix T is used. So, now the sender's security in the base OTs ensure the following. Either Alice will receive the ith column or she will receive the ith column XORed with B but if she is receiving the ith column XORed with B, she will not be learning the ith column.

For the jth OT she will be learning the jth column or the jth column XORed with B. So, even if she cancel, even if he XORs these 2 things, she learns basically XOR of 2 random columns and which is completely independent of the actual choice vector of Bob. So, that is why the sender's security of base OT ensures that nothing about Bob's choice vector B is learned and the fact that each of the columns of the T matrix are picked randomly.

(Refer Slide Time: 10:16)



IKNP OT Extension : Alice's Security

$S = (s_1, \cdots, s_\lambda) \in_r \{0,1\}^\lambda$

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

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

For $i = 1, \cdots, k$

$Y_{i0} = m_{i0} \oplus H(i, Q_i)$

$Y_{i1} = m_{i1} \oplus H(i, Q_i \oplus S)$

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

$T = \begin{bmatrix} \leftarrow T_1 \rightarrow \\ \leftarrow T_2 \rightarrow \\ \leftarrow T_i \rightarrow \\ \leftarrow T_k \rightarrow \end{bmatrix} = \begin{bmatrix} T^1 & T^2 & T^i & T^\lambda \end{bmatrix}$ $k \times \lambda$ $\lambda \times k$

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

- Security of Alice reduces to receiver's security in base OTs and security of correlation-robust hash function, conditioned on the event that $S \neq 0^\lambda$

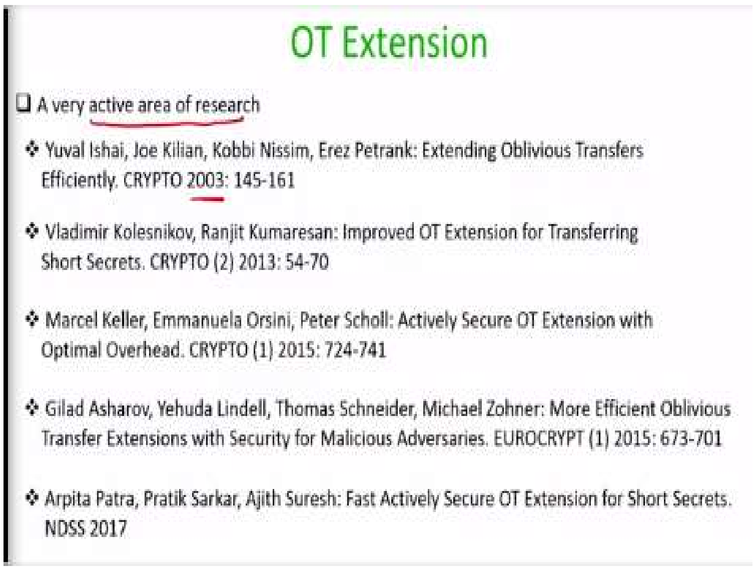
Now what about Alice's security in the whole procedure? We want to argue here that if Bob is corrupt, then for the ith pair he learns only the message with index b $_i$, we have to argue that he does not learn anything about the message with index 1 - b $_i$ from the ith pair. And this reduces to the security of receiver in the base OTs because Alice is acting as the receiver in this base OTs and she is participating with the individual entries of the S vector.

If somehow Bob learns the full S vector, then the problem arises but if S vector remains hidden, then even though for the other message in the ith pair, he receives the corresponding ciphertext. He receives the ciphertext **y** $_{i\ 1}$ - b $_i$, he receives this ciphertext, fine. But this other ciphertext it is prepared using the pad T $_i$ XORed with S. And if S is hidden from Bob, then the security of the co-relation robust hash function h of i T $_i$ XORed with S is like a random pad for the Bob.

So, if Bob does not learn anything about S and if S is a non zero vector then we can simply base our Alice's security on receiver's the security of base OT and a co-relation robust hash function. Now what is the probability that S is an all 0 vector, remember S is picked uniformly at random by Alice. And the probability that is an all 0 vector is 1 over 2 power λ which is a negligible function in the security parameter. Why we are arguing about the condition S being all 0 vector?

Because if S is an all 0 vector then in the ith pair both the message with index 0 as well as the message with index 1 will be XORed with output of H $_i$ and Q $_i$. And once Bob receives decrypts one of the encryption he will be able to decrypt the other encryption as well if S would have been all 0. But if S is not 0 then from Alice's security or receiver's the security of the base OT nothing about the individual entries of this S factor is learned by Bob. And then we can argue about the security of Alice based on the co-relation robust hash function.

(Refer Slide Time: 13:05)



OT Extension

- A very active area of research
  - Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank: Extending Oblivious Transfers Efficiently. CRYPTO 2003: 145-161
  - Vladimir Kolesnikov, Ranjit Kumaresan: Improved OT Extension for Transferring Short Secrets. CRYPTO (2) 2013: 54-70
  - Marcel Keller, Emmanuela Orsini, Peter Scholl: Actively Secure OT Extension with Optimal Overhead. CRYPTO (1) 2015: 724-741
  - Gilad Asharov, Yehuda Lindell, Thomas Schneider, Michael Zohner: More Efficient Oblivious Transfer Extensions with Security for Malicious Adversaries. EUROCRYPT (1) 2015: 673-701
  - Arpita Patra, Pratik Sarkar, Ajith Suresh: Fast Actively Secure OT Extension for Short Secrets. NDSS 2017

So, that completes the description of the IKNP OT extension protocol. It is a very, very active area of research because as I said oblivious transfer protocols they are very important cryptographic primitive. And they are very expensive to implement because they involve public key operations. Through OT extensions, we would like to perform only less number of OT instances and get the effect of a huge number of OTs by performing cheap symmetric key operations.

And this is very, very relevant in the practical context when you want to deploy GMW protocol on later on the Yao's protocol, where oblivious transfer protocols are involved in an extensive way. So, after the work of Ishai et al, several significant improvements have been done, very nice results have been obtained in the field of OT extension. And I have listed down here few of the important works. Thank you.