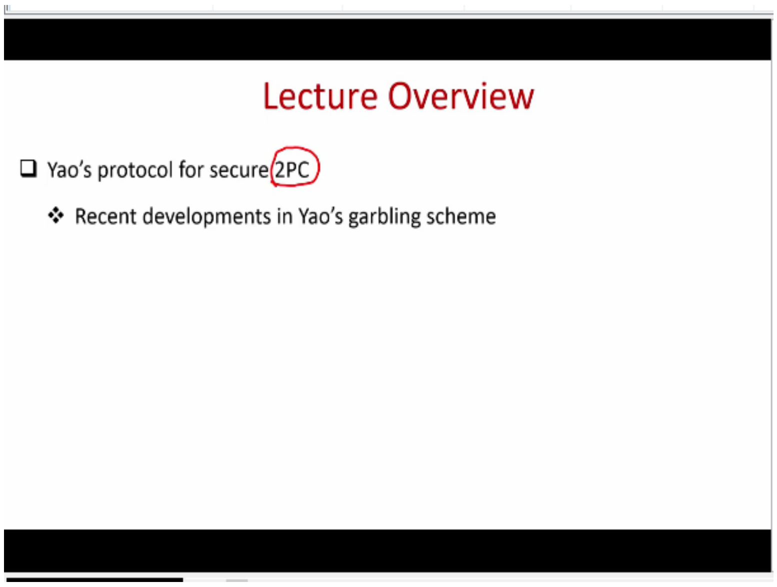
# Secure Computation: Part I

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

## Lecture-50
## Yao's Protocol for Secure 2PC

**(Refer Slide Time: 00:31)**

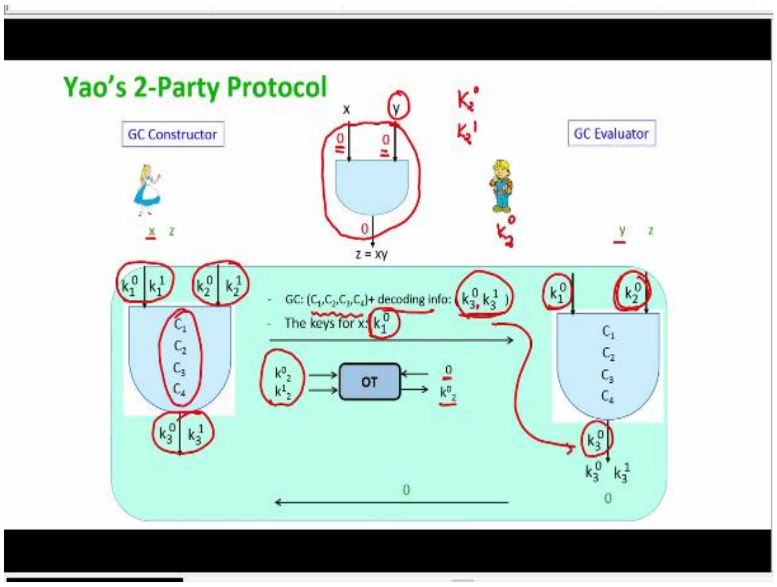


### Lecture Overview

- ❑ Yao's protocol for secure 2PC
    - ❖ Recent developments in Yao's garbling scheme

Hello everyone, welcome to this lecture. So, we had seen in the last lecture what Yao's garbling scheme is. So, we will now see the full-fledged secure 2PC protocol using the Yao's garbling scheme and then we will see some of the recent developments which has happened in the context of Yao's garbling scheme.

**(Refer Slide Time: 00:50)**



### Yao's 2-Party Protocol

- GC: $(C_1, C_2, C_3, C_4)$+ decoding info: $(k_3^0, k_3^1)$
- The keys for x: $k_1^0$

Here is the full Yao's 2-party protocol. To make it very simple, we consider a very simple function namely, a 2-party AND function of course, there could be a function which is enormously large, enormously complex and represented by a very complex Boolean circuit, but let us make it first very simple. And we consider the case where we have Alice and Bob with private bits $x$ and $y$ respectively and they want to obtain output $z$.

Right now, you might be wondering that we have already proved that it is impossible to compute securely the AND function, but that was in the context of perfect security where one of these 2 parties could be corrupted by a computationally unbounded adversary. Now, we are in the cryptographically secure world where the corrupt party is only polynomially time bounded. Now, we will see how, using the Yao's 2-party protocol, we can allow Alice and Bob to securely computer and have their private bits.

So, as part of the protocol description, we can assign the role of garbled circuit constructor to one of these 2 parties and a garbled circuit evaluator to the other party. So, for simplicity, let us make Alice the constructor and Bob as the evaluator. So, what does Alice have to do? Alice has to first garble the wires. So, how she can garble the wires? For the $x$ wire, she will assign 2 indistinguishable encryption keys.

For the $y$ wire, she will assign 2 indistinguishable keys, for the $z$ wire, she will assign 2 indistinguishable encryption keys and only she will know the labels of the assigned keys. And for simplicity, let us take a concrete instantiation of the protocol where the values of $x$ and $y$ are both 0 and hence output will be 0. So, the security property that we will require here is that if Alice is corrupt, anyhow, she knows that her input is 0.

So the final output she will know is 0, but we require that she should not learn anything about $y$. If Bob is corrupt similarly, he should not learn anything, whether $x$ was 0 or X was 1. So, this is how Alice will do the garbling of the wires. And now she will prepare the double encryptions by encrypting the keys $k_3^0$, $k_3^1$, appropriately using the key combinations assigned to the $x$ wire and $y$ wire and randomly shuffling the 4 cipher texts.

What she will give to the evaluator? She will give to the evaluator, the garbled circuit. In this example the circuit consists of just a single gate. So, she will give Bob all the 4 cipher texts associated with this AND gate. Remember, these ciphertexts are shuffled that means it is not the case that $c_1$ is for the first row of the truth table of the AND function or $c_4$ is for the last row of the truth table of the AND function.

And it also gives Bob the decoding information. So, what is the decoding information? The labels of the keys assigned to the $z$ wire. So, $k_3^0$, $k_3^1$; both of them will be given to Bob along with the label that the first key is for $z = 0$, the second key is for $z = 1$. As of now, nothing about $x$ and $y$ have been exchanged. Now, since the input $x$ belongs to Alice and the value of $x$ that we are taking in this execution is 0. What Alice will provide to Bob is the following. It will provide the encoded $x$ since $x$ is 0, encoded $x$ is nothing but the key corresponding to $x$ being 0.

Now Bob will be just seeing the key, from the key it cannot figure out whether it is seeing the key for $x$ being 0 or whether it is seeing the key for $x$ being 1. That preserves the privacy of Alice's key. And now, we have to also provide Bob's encoded $y$. Here, encoded $y$ means the key corresponding to the bit $y$ and $y$ in this case is 0. So, somehow Alice should provide Bob, the key corresponding to the $y$ wire or the second wire where $y$ is 0.

But how can Alice provide that key to Bob? Because Bob cannot go and directly tell Alice that his input is 0, and request her to give the key for, $y$ being 0, because that breaches the privacy of Bob. And on the other hand, we cannot ask Alice to give both the keys to Bob, corresponding to $y$, because that will be a security breach. Because remember, for security purposes, we need the invariant that over each wire, only one of the keys should be available to Bob, or to the evaluator.

If Bob is given both the keys corresponding to $k_2$ being 0 and $k_2$ being 1, then he will be able to decrypt more than 1 double encryption here. And then comparing it later with the decoding information, it can figure out whether it has participated in input configuration, $x$ being 0 and $y$ being 0 or $x$ being 1 and $y$ being 0. So, that is fine for the security purpose, he cannot directly go and give Bob both the keys for the bit $y$. So, now we are now faced with a problem.

Bob wants only one of the 2 keys. So, there are 2 keys assigned to the value $y$, Bob wants only one of them without actually telling which of the 2 keys he is interested in. But it turns out that we already have seen mechanism to get around this problem and the mechanism is to execute an oblivious transfer protocol. So, Bob will participate as a receiver in an instance of oblivious transfer protocol. The inputs for the Alice as a sender in the OT instance will be the keys corresponding to the $y$ wire.

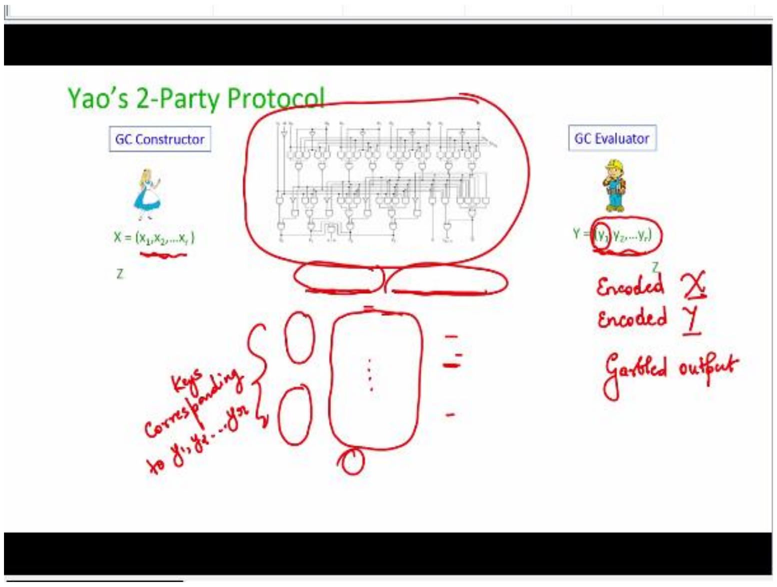
And the input as a receiver for Bob will be his value of $y$. From the correctness property of the oblivious transfer, Bob will obtain only the key corresponding to $y$ = 0, it will not learn anything about the other key. So, now Bob will start evaluating the circuit, he has only one key for $x$, one key for $y$.

So, he knows that the key that it has got for the $y$ wire is actually for $y = 0$ that is not a breach of privacy, because anyhow, Bob himself is the owner of the $y$ input, but the privacy of the $x$ input is preserved because for the $x$ wire, Bob is not aware of the label of the key. And now he has 4 cipher texts here. Assuming this encryption scheme, which has been used here has special correctness property, it is only one of the cipher texts which he will be able to decrypt.

In this case, he will be only able to decrypt the cipher text which has been encrypted using the key combination $k_1^0$, $k_2^0$ and after decrypting that ciphertext he will obtain the message which has been encrypted. In that ciphertext the message which has been encrypted in that ciphertext will be $k_3^0$, because that is how Alice would have prepared the double encryptions. And now, by comparing this $k_3^0$ with the decoding information that Bob has, it will be able to figure out that output of the function is 0 and it can tell the result to Alice.

So, now you can see that we require only a fixed amount of interaction between Alice and Bob. You might be wondering that what if we have more than one gate in the circuit? Does the Yao protocol ensure that still Alice and Bob have to interact only a fixed number of times? And answer is yes.

**(Refer Slide Time: 09:50)**



So, now imagine that there is a very big function, very complex function, where there are multiple input bits for Alice and multiple input bits for Bob. For simplicity, I am assuming that they have the same number of inputs odd number of inputs, but that may not be the case. It might be case that Alice has say 100 input bits, Bob has 200 input bits and so on. And there is a huge Boolean circuit, which they want to securely evaluate.

Yao's 2 party protocol will still require Alice and Bob to interact constant number of times. So, here is how the protocol will proceed. Alice will garble the entire circuit; garbling the entire circuit means for each wire assigning 2 indistinguishable keys for each gate prepare 4 double encryption. And all those double encryptions are randomly permuted for each gate and given to Bob plus the decoding information.

Namely for the output wires of the function or the circuit, give Bob both the keys along with the labels. And for her own input bits, $x_1, x_2, \dots, x_r$, the keys corresponding to those bits, but without the labels, are given by Alice to Bob and this requires only one round of interaction. For doing this, Alice has to go over the entire circuit once. Now, it is the Bob's turn to get the keys corresponding to his input bits.

So, I am assuming here that Bob has odd number of input bits, so, we have to execute $r$ instances of oblivious transfer protocol here. Alice will be acting as the sender in all these OT instances. For the first OT instance, she will participate with the keys$y_1^0$, $y_1^1$. for the second wire and Bob will be participating with choice bit being $y_1$ and as a result it will obtain only the key corresponding to $y_1$.

Bob will obtain the key corresponding to $y_1$ depending upon the value of his choice bit $y_1$. In the same way for the $r$th OT instance, Alice will participate as sender with the 2 indistinguishable keys that it has assigned for $y_r$. So, all these are keys corresponding to the bits $y_1, y_2, .., y_r$ and Bob is participating as a receiver in this OT instances with the exact value of his input bits.

And depending upon the exact value of his input bits Bob receives the corresponding keys and will now be having encoded $x$,and encoded $y$. And all these OT instances can be executed in parallel. And OT can be executed in constant number of rounds. So, all these OT instances can be executed in 1 shot. And now once Bob has encoded $x$, encoded $y$ plus the garbled circuit, it can start evaluating the circuit gate by gate.

For each gate it has to try opening all the 4 ciphertexts, one of the ciphertexts will be opened or decrypted, it will obtain the corresponding message and assign it as the key to the output of that gate and so on and once that has obtained all the garbled output, the garbled output will be decoded using the decoding information that will help Bob to find out the exact value of the function output which it can now communicate back to Alice.

And hence Alice and Bob both learn the function output. So, that is how the entire protocol will work. And you can now see that irrespective of how many gates are there, what is the multiplicative depth of the circuit? It requires only a constant number of interactions between Alice and Bob here. So, proving the security of Yao's protocol is slightly challenging here due to interest of time I am not going into the full formal details, but I have already discussed a reference to see the full security proof here.

Intuitively, let us try to understand why exactly this whole protocol is secure. Consider the case when Alice is corrupt. When Alice is corrupt, does she learn anything about Bob's input? And the answer is no, because that boils down to basically the receiver's privacy from the OT instances. Because after these OT instances, there is no interaction between Alice and Bob, it is only the clear function output which is given back from Bob to Alice.

And based on $x$ and $z$, whatever Alice can infer about $y$ that is anyhow being allowed to be inferred. So, the only scope for potentially corrupt Alice to learn anything about $y$ is during the OT instances, but OTs ensure that receiver's privacy is maintained and hence nothing about $y$ is revealed to a corrupt Alice. Now, consider the case when Bob is corrupt. And when Bob is corrupt, we have to argue whether it learns anything about $x$ or not.

Well, it sees the encoded $x$ and encoded $x$ is basically a bunch of unlabeled keys, those keys are indistinguishable keys and hence Bob cannot figure out whether it is seeing the keys for $x_1$ being 0, $x_2$ being 1 and so on. But, it is also obtaining the garbled circuit and in the garbled circuit it is seeing 4 double encryptions. One of those double encryptions he will be able to open and we have to argue that it does not learn anything about what ciphertext it has opened.

So, for instance, if I consider a garbled AND gate where the 4 double encryptions are randomly permuted; even though they are randomly permuted, Bob knows that 3 of these ciphertexts encode or encrypt the output key corresponding to 0. And only one of the ciphertext encrypt the output key corresponding to 1, that much information Bob has because that is the property of the AND gate.

Now, suppose it is able to find out somehow it gets a sense that out of this $c_1, c_2, c_3, c_4$ it figures out that $c_1, c_3$ and $c_4$ are encryptions of the same message, and $c_2$ is an encryption of a different message. Suppose it gets a hint of that based on the garbling function and so on, then that itself is a breach of privacy. Because even if it learns that out of these 4 ciphertext, a subset of ciphertext, encrypt the same output message and another subset of ciphertext encrypt another message.

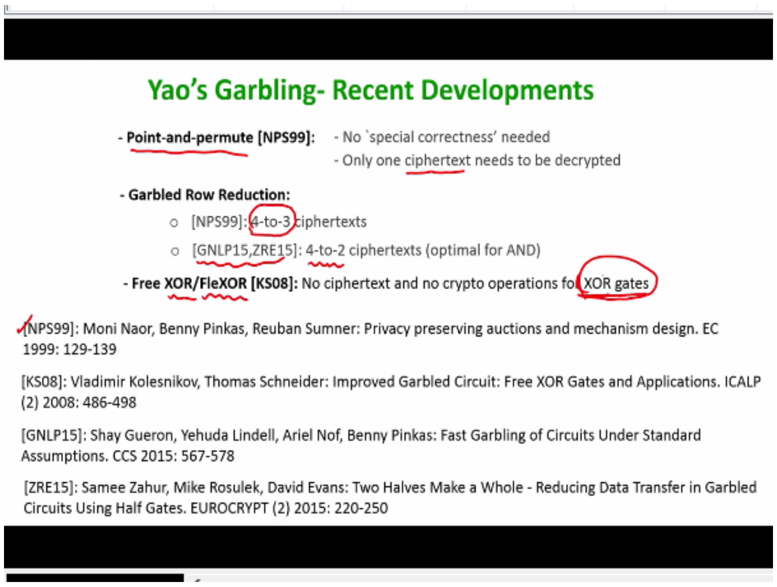
Then that is also a breach of privacy because ideally, we require that Bob should not learn anything regarding the position of the rows of the truth table and the labels of the output keys and so on. And arguing it formally based on the fact that for the input of this garbled AND gate Bob will be having only 1 key for both the input wires is slightly challenging him. But we can still formally prove that even though Bob has $c_1, c_2, c_3 c_4$.

If I consider say for instance the AND input gates to be $a$, $b$ and output to be $c$ and we have the key pairs$(k_a^0, k_b^0)$, $(k_a^0, k_b^1)$, $(k_a^1, k_b^0)$ and $(k_a^1, k_b^1)$. And say Bob gets the keys $k_a^0$ and $k_b^1$, then even though the ciphertexts which are prepared here are using these 4 keys. For one of the ciphertext Bob will be having both the keys, say it is $k_a^0$ and $k_b^1$. So, it can be completely decrypted by Bob.

But for the other ciphertext Bob does not have both the keys but he has one of the keys. So, for instance, if I take the first ciphertext, it is in encrypted using $k_a^0$ and $k_b^0$. So, $k_a^0$ is available with Bob but he does not have $k_b^0$. So, we have to formally argue that even though he has one of the 2 keys with which first ciphertext is prepared, it does not learn anything about what is kept inside and does not learn what anything about what is kept inside.

By that I formally mean even the fact that whether it is the same thing which is encrypted in the other ciphertext or not, no, not even the position. So, arguing it formally is slightly challenging, but we can prove that indeed if our symmetric encryption scheme has say CPA security property and the special correctness property then a corrupt Bob, who for each of the gates has only just a single pair of key corresponding to the 2 input wires cannot learn anything about what is encrypted in the remaining three cipher texts. So, that intuitively proves the privacy for Alice from a potentially corrupt power.

**(Refer Slide Time: 20:55)**



## Yao's Garbling- Recent Developments

- **Point-and-permute [NPS99]:** - No `special correctness' needed
  - Only one ciphertext needs to be decrypted
- **Garbled Row Reduction:**
  - [NPS99]: 4-to-3 ciphertexts
  - [GNLP15,ZRE15]: 4-to-2 ciphertexts (optimal for AND)
- **Free XOR/FleXOR [KS08]:** No ciphertext and no crypto operations for XOR gates

✓[NPS99]: Moni Naor, Benny Pinkas, Reuban Sumner: Privacy preserving auctions and mechanism design. EC 1999: 129-139

[KS08]: Vladimir Kolesnikov, Thomas Schneider: Improved Garbled Circuit: Free XOR Gates and Applications. ICALP (2) 2008: 486-498

[GNLP15]: Shay Gueron, Yehuda Lindell, Ariel Nof, Benny Pinkas: Fast Garbling of Circuits Under Standard Assumptions. CCS 2015: 567-578

[ZRE15]: Samee Zahur, Mike Rosulek, David Evans: Two Halves Make a Whole - Reducing Data Transfer in Garbled Circuits Using Half Gates. EUROCRYPT (2) 2015: 220-250

So, now, let me quickly go through some of the recent developments, which has happened in the regime of Yao's garbling, because Yao's protocol is very fundamental, it gives you very strong guarantees, because it requires a constant number of rounds here and hence, it might be very useful in practical scenarios, where the parties do not want to talk too many times. So, there are various interesting optimizations which have been proposed.

So, one of these optimizations is called as point and permute technique, which we will see in the next lecture introduced in this paper. And this technique ensures that you do not require any special correctness property from your underlying symmetric encryption scheme. So, even if you take a regular CPA secure encryption scheme based on AES and based on modes of operations, even without requiring special correctness property, we can ensure that the evaluator opens the right ciphertext for each of the gates without actually knowing what ciphertext it is opening, whether it is corresponding to the first row of the truth table or second row of the truth table, third row of the truth table and so and more importantly, the evaluator does not have to try decrypting or opening all the 4 ciphertext. Magically it will know that this is the ciphertext which I have to decrypt without knowing anything regarding the position of that ciphertext out of the 4 ciphertext.

And we have another technique called garbled row reduction method proposed in the same paper, which reduces the size of the garbled gates from 4 ciphertexts to 3 ciphertexts. And this is a significant saving, because if you see closely, even though Yao's protocol is a constant round protocol for each gate, we have to prepare 4 ciphertext and communicate it to the evaluator.

So, if you have say a million number of gates, we are actually creating 4 million ciphertexts and the cipher text size will be tremendously large even if we are using symmetric encryption and say each cipher text size is 256 bits. Computing and communicating 4 million ciphertexts of size 256 bits is an enormous amount of computation and communication. So, even if we can reduce the size of each garbled gate by 1 ciphertext that is a tremendous saving practically.

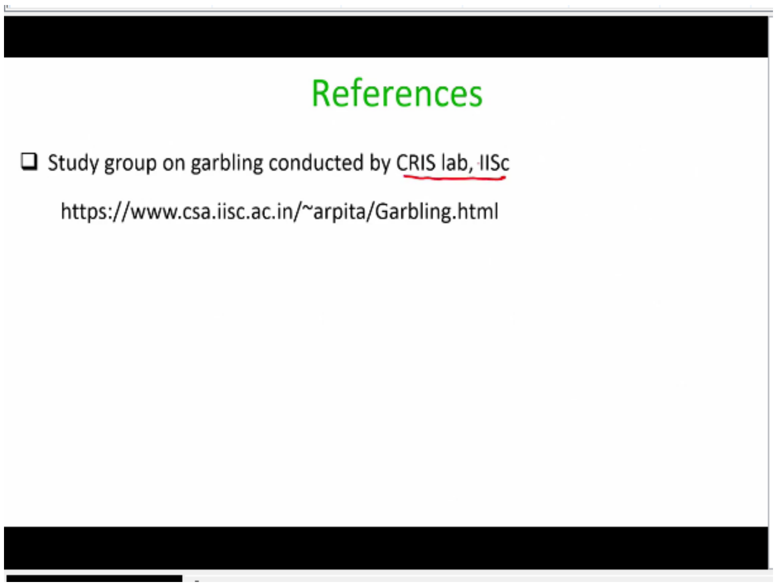
So, this garbled row reduction method enables one to garble the gates in such a way that instead of 4 ciphertext, we require only 3 ciphertext for each gate. Then there is a series of work which tries to further reduce the number of ciphertext for each garbled gate. So, these papers propose garbling techniques we are for each gate, only 2 ciphertexts have to be computed as part of garbling.

And then there is this very nice cute method which is called as free XOR method which allows to garble the XOR gates in such a way that you do not require any ciphertext to be computed as part of garbling. So, this is something equivalent to your linearity property where the garbling will be done in such a way that for garbling the XOR gate, no ciphertext have to be computed and communicated.

And magically when evaluator is evaluating a XOR gate based on the keys which it possesses for the inputs of the XOR gates it can automatically obtain the right key corresponding to the output of the XOR gate and this is a very, very significant reduction because now you can imagine that in your circuit if you have a vast amount of XOR gates then you can evaluate them freely.

The garblers do not have to do anything to garble those XOR gates, the evaluator do not have to do any decryption operation for evaluating those garbled circuits. So, that is why this is a very, very important optimization.

**(Refer Slide Time: 25:32)**



## References

- Study group on garbling conducted by CRIS lab, IISc

  https://www.csa.iisc.ac.in/~arpita/Garbling.html

So, there are lots of references which you can find in the internet regarding the Yao's garbling scheme, recent optimizations and so on. So, at CRIS lab, Indian Institute of Science, some time back, we conducted a study group specifically to discuss the garbling scheme starting from the historical result and discussing all the way the state of the odd results. So, you are referred to this link; there you can get all the important and relevant references. Thank you.