SEC595
Encrypted Computing

Lecture 16: Oblivious Transfer

Limitations of Yao's Algorithms

- Previous solutions to Yao’s problem uses RSA encryption scheme
  - Assume RSA public key scheme is adopted
  - Assume $O_1 \leq A, B \leq O_2$
  - Finds whether A>B

- For a general version of Yao's Millionaire problem that works for any function, we need to introduce two concepts
  - **Oblivious Transfer:** to securely selecting a value
  - **Garbled Circuits:** to represent any arithmetic function F

1-out-of-2 Oblivious Transfer (OT)

- A Fundamental SMC primitive
- Sender has two messages m_0 and m_1
- Receiver has a bit b, and the receiver wishes to receive m_b, without the sender learning b
- Sender wants to ensure that the receiver receives only one of the two messages

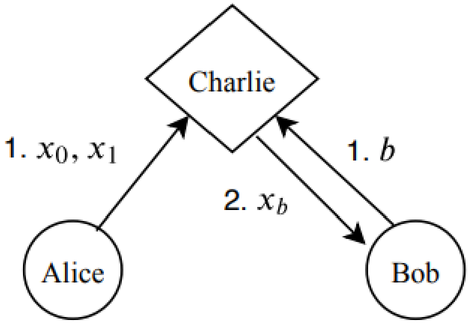


FIGURE 6.1: 1-2 Oblivious Transfer using trusted party

- Inputs
  - Sender has two messages m_0 and m_1
  - Receiver has a single bit $b \in \{0,1\}$
- Outputs
  - Sender receives nothing
  - Receiver obtain m_b and learns nothing of m_{1-b}

1-2 OT with Trusted third party during Setup

1. Bob sends its input $b \in \{0, 1\}$ to Charlie.

2. Charlie generates two public-private key pairs say using El-Gamal's public key encryption scheme described above. Let these be (pk_0, sk_0) and (pk_1, sk_1). Charlie then sends sk_b to Bob and pk_0, pk_1 to Alice.

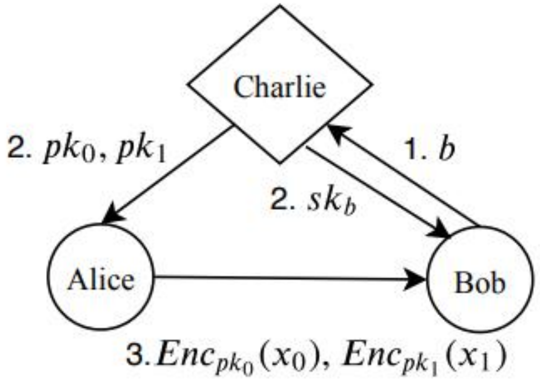


FIGURE 6.2: 1-2 Oblivious Transfer with trusted party during Setup phase.

1. Alice sends $(c_1, c_2) = (\mathsf{Encrypt}(pk_0, x_0), \mathsf{Encrypt}(pk_0, x_0))$ to Bob.

2. Bob on receiving (c_1, c_2) tries to decrypt both using sk_b and the keeps the plain text that she can correctly decrypt.

1-2 OT without a third party??

Naor, Moni, and Benny Pinkas. "Oblivious transfer and polynomial evaluation." In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 245-254. 1999.
Beaver, Donald. "Precomputing oblivious transfer." In *Annual International Cryptology Conference*, pp. 97-109. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.

A 1-out-of-2 Oblivious Transfer Protocol

Step 1 Alice	Alice generates 1- an RSA key pair $\mathrm{PK} = (\mathrm{n}, \mathrm{e})$ and $\mathrm{SK} = (\mathrm{d})$ 2- two random values, r_0 and r_1, and sends them to Bob along with her PK
Bob picks a bit b to be either 0 or 1, and selects r_b	Step 2 Bob
Bob generates a random value k and blinds it with r_b by computing $\mathrm{v} = \mathrm{r}_\mathrm{b} + \mathrm{k}^\mathrm{e} \bmod \mathrm{n}$ and sends it to Alice	Step 3 Bob
Step 4 Alice	Alice doesn't know which of r_0 and r_1 Bob chose. Alice computes $\mathrm{k}_0 = (\mathrm{v} - r_0)^\mathrm{d} \bmod \mathrm{n}$ and $\mathrm{k}_1 = (\mathrm{v} - r_1)^\mathrm{d} \bmod \mathrm{n}$
Step 5 Alice	Alice combines the two secret messages with each of the possible keys, i.e. $\mathrm{m}_0' = \mathrm{m}_0 + \mathrm{k}_0$ and $\mathrm{m}_1' = \mathrm{m}_1 + \mathrm{k}_1$, and sends them both to Bob
Bob knows which of the two messages can be unblinded with k, so he is able to compute exactly one of the messages $\mathrm{m}_\mathrm{b} = \mathrm{m}_\mathrm{b}' - \mathrm{k}$	Step 6 Bob

A 1-out-of-2 Oblivious Transfer Protocol

	Sender (Bob)			Receiver (Alice)	
Input messages	m_0, m_1			$b \quad k$	Choice bit b, random k
RSA key pair	d	n, e	$\longrightarrow$	n, e	
Random strings		r_0, r_1	$\longrightarrow$	r_0, r_1	
		v	$\longleftarrow$	$v = (r_b + k^e) \bmod n$	

$k_0 = (v - r_0)^d \bmod N$

$k_1 = (v - r_1)^d \bmod N$

$m'_0 = m_0 + k_0$	$\longrightarrow$	m'_0
$m'_1 = m_1 + k_1$		m'_1

$m_b = m'_b - k$

1-out-of-n OT?

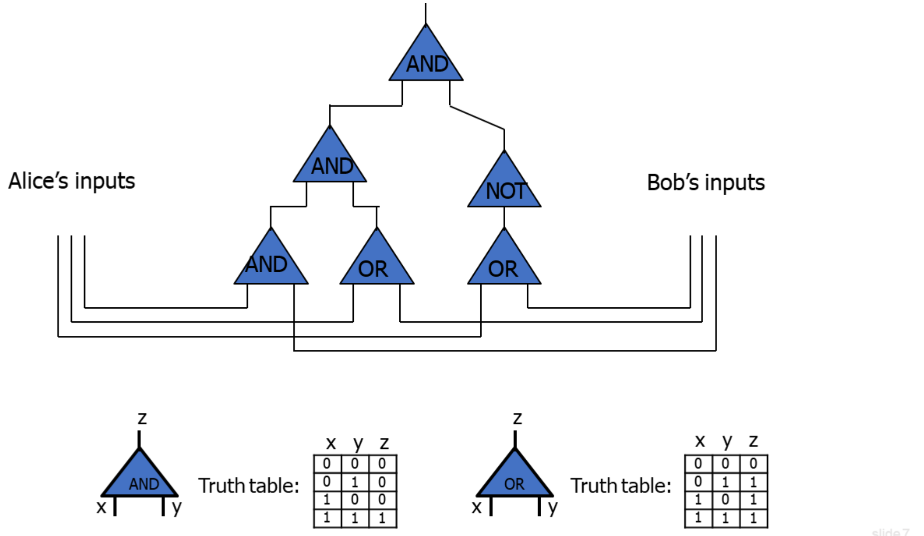
- Sender has n messages m_0 through m_{n-1}
- Receiver has an index b, and the receiver wishes to receive m_b, without the sender learning b
- Sender wants to ensure that the receiver receives only m_b and learns nothing about the other $n-1$ messages
- How can we achieve this using 1-out-of-2 OT?

https://github.com/wyatt-howe/1-out-of-n

https://www.cs.umd.edu/~jkatz/gradcrypto2/f13/lecture4.pdf

Overview of Yao's Garbled Circuit Protocol

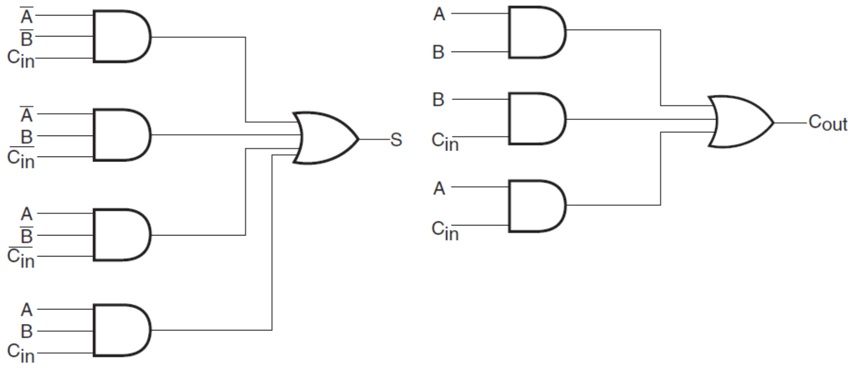
- Assume that Alice has number X, Bob has number Y, Alice learns $F(X,Y)$, and Bob learns nothing
  - Represent $F(X,Y)$ using a Boolean circuit
  - Alice creates and encrypts the circuit. Each wire in in the circuit is associated with two random values. Alice sends the "garbled" circuit to Bob
  - Alice sends the values corresponding to her input bits
  - Bob uses OT to obtain values for his bits
  - Bob evaluates the circuits and send the result to Alice



Circuit Computation

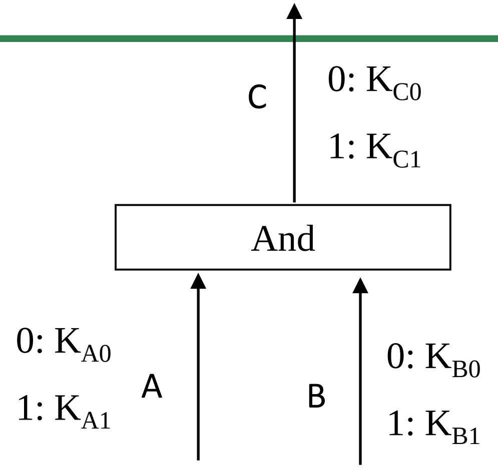
- The design of Yao’s protocol is based on circuit computation

- Any (efficiently) computable function can be represented as a family of (polynomial-size) Boolean arithmetic circuits
  - Such a circuit consists of **AND**, **OR**, and **NOT** gates



Garbled Circuit

- We can represent Alice's circuit with a garbled circuit so that evaluating it does not leak information about intermediate results
- For each edge in the circuit, we use two random keys to represent 0 and 1, respectively
- We represent each gate with 4 ciphertexts, for input (0,0), (0,1), (1,0), (1,1), respectively
- The ciphertext for input (A, B) is the key representing the output $Gate(A, B)$ encrypted by the keys representing A and B
- The entries of the truth table can be permutated
- Alice keeps the mapping between edges and random keys to her self



A	B	C	
0	1	0	$E_{K_{A0}}(E_{K_{B1}}(K_{C0}))$
1	1	1	$E_{K_{A1}}(E_{K_{B1}}(K_{C1}))$
1	0	0	$E_{K_{A1}}(E_{K_{B0}}(K_{C0}))$
0	0	0	$E_{K_{A0}}(E_{K_{B0}}(K_{C0}))$

Evaluation of Garbed Circuit

- Given the keys representing the inputs of a gate, we can easily obtain the key representing the output of the gate
  - Only need to decrypt the corresponding entry
  - But we do not know which entry it is?
  - We need decrypt all entries
  - To get a successful decryption, we append few zeroes when encrypting output key
    - Example, $K'_{C0} = K_{C0}00000$ and $K'_{C1} = K_{C1}00000$

A	B	C	
0	1	0	$E_{K_{A0}}(E_{K_{B1}}(K_{C0}))$
1	1	1	$E_{K_{A1}}(E_{K_{B1}}(K_{C1}))$
1	0	0	$E_{K_{A1}}(E_{K_{B0}}(K_{C0}))$
0	0	0	$E_{K_{A0}}(E_{K_{B0}}(K_{C0}))$

Translating Input

- Alice sends the garbled circuit, and the keys corresponding to her input. (How about Bob's input?)
- So, we know that, given the keys representing Bob’s private input, we can evaluate the garbled circuit.
  - Then Bob can evaluate the garbled circuit if he knows how to translate his input to the keys
- But Alice can’t give the translation table to Bob.
  - Otherwise, Bob can learn information during evaluation
- A solution to this problem is 1-out-of-2 OT for each input bit
  - Alice sends the keys representing 0 and 1;
  - Bob chooses to receive the key representing his input at this bit
  - Clearly, Bob can’t evaluate the circuit at any other input

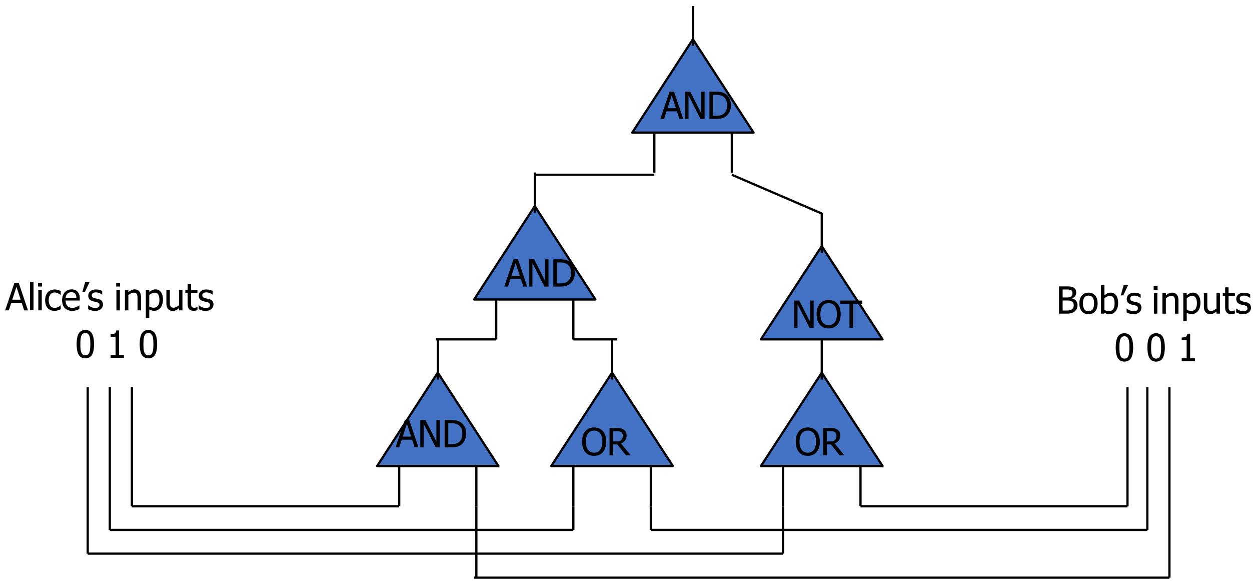
Finishing the Evaluation

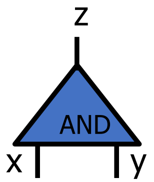
- At the end of evaluation, Bob gets the keys representing the output bits of circuit

- Alice sends Bob a table of the keys for each output bit

- Bob translates the keys back to the output bits

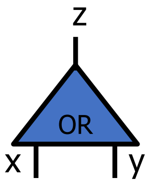
Example





Truth table:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1



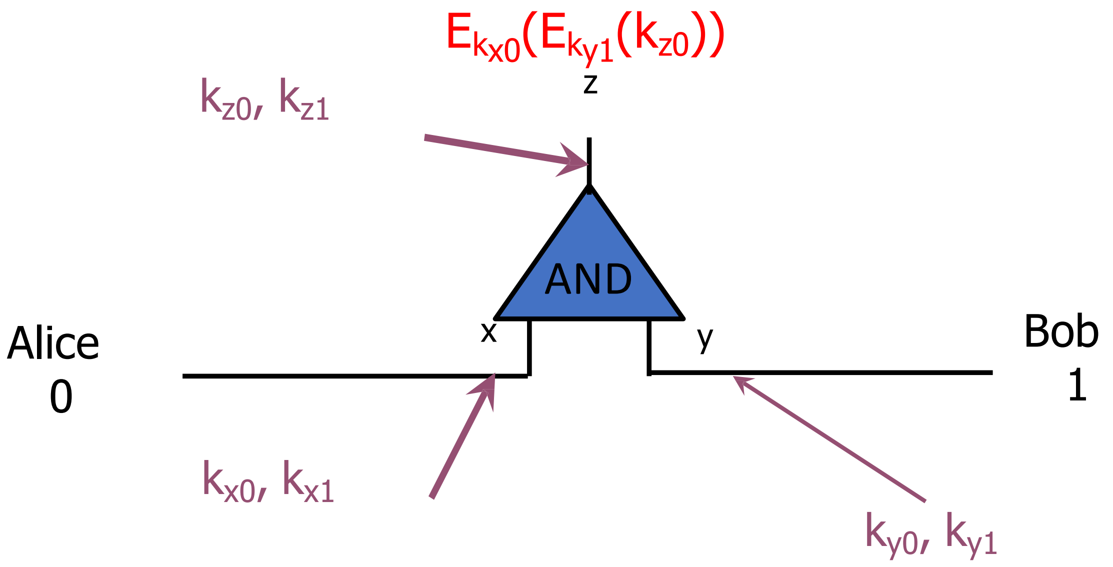
Truth table:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

slide 7

Step1: Pick Random Keys For Each Wire

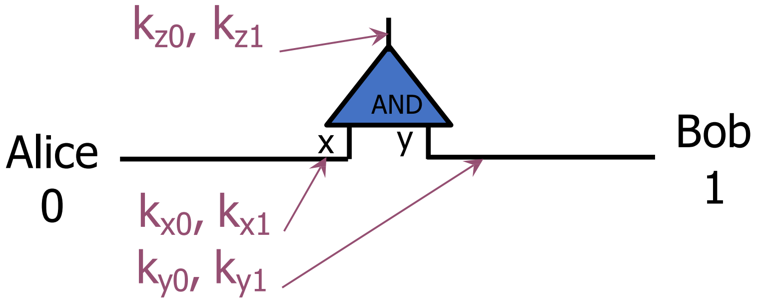
- Alice picks two random keys for each wire
  - One key corresponds to "0", the other to "1"
  - 6 keys in total for a gate with 2 input wires



slide 8

Step 2: Encrypt Truth Table

- Alice encrypts each row of the truth table by encrypting the output-wire key with the corresponding pair of input-wire keys z



Original truth table:

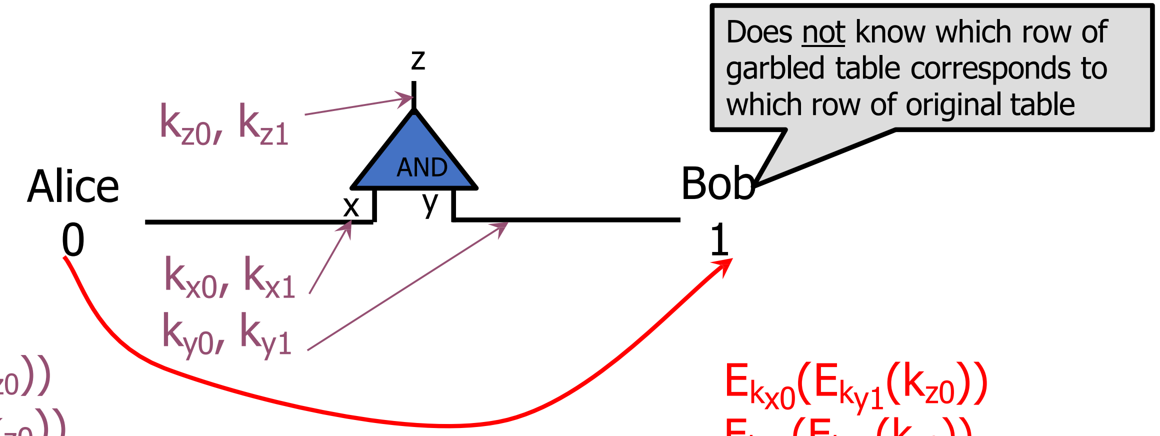
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Encrypted truth table:

$E_{k_{x0}}(E_{k_{y0}}(k_{z0}))$
$E_{k_{x0}}(E_{k_{y1}}(k_{z0}))$
$E_{k_{x1}}(E_{k_{y0}}(k_{z0}))$
$E_{k_{x1}}(E_{k_{y1}}(k_{z1}))$

Step 3: Send Garbled Truth Table

- Alice randomly permutes (“garbles”) encrypted truth table and sends it to Bob



$E_{k_{x0}}(E_{k_{y0}}(k_{z0}))$
$E_{k_{x0}}(E_{k_{y1}}(k_{z0}))$
$E_{k_{x1}}(E_{k_{y0}}(k_{z0}))$
$E_{k_{x1}}(E_{k_{y1}}(k_{z1}))$

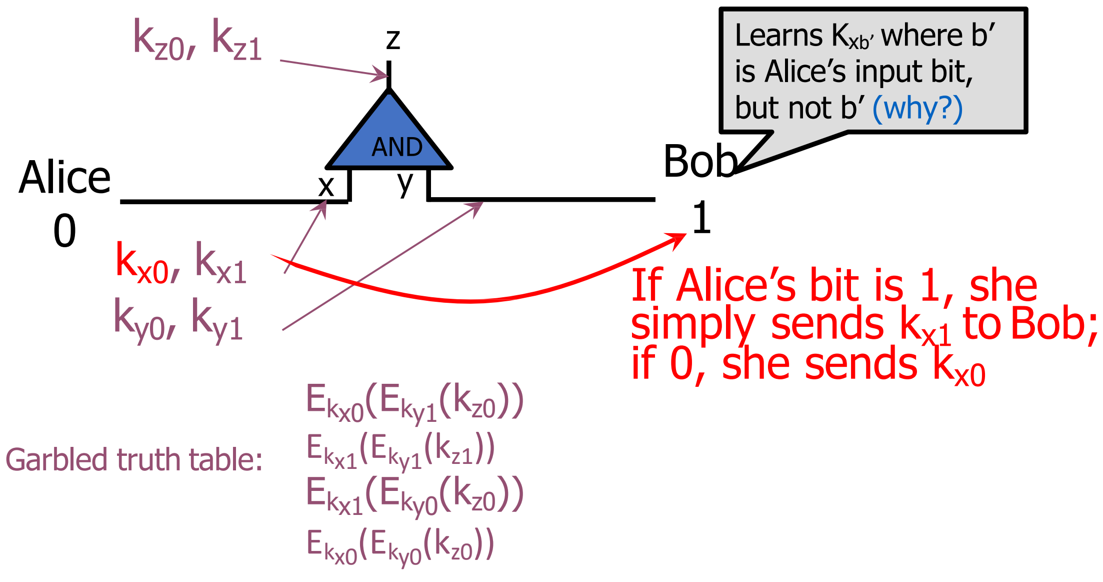
Garbled truth table:

$E_{k_{x0}}(E_{k_{y1}}(k_{z0}))$
$E_{k_{x1}}(E_{k_{y1}}(k_{z1}))$
$E_{k_{x1}}(E_{k_{y0}}(k_{z0}))$
$E_{k_{x0}}(E_{k_{y0}}(k_{z0}))$

slide 10

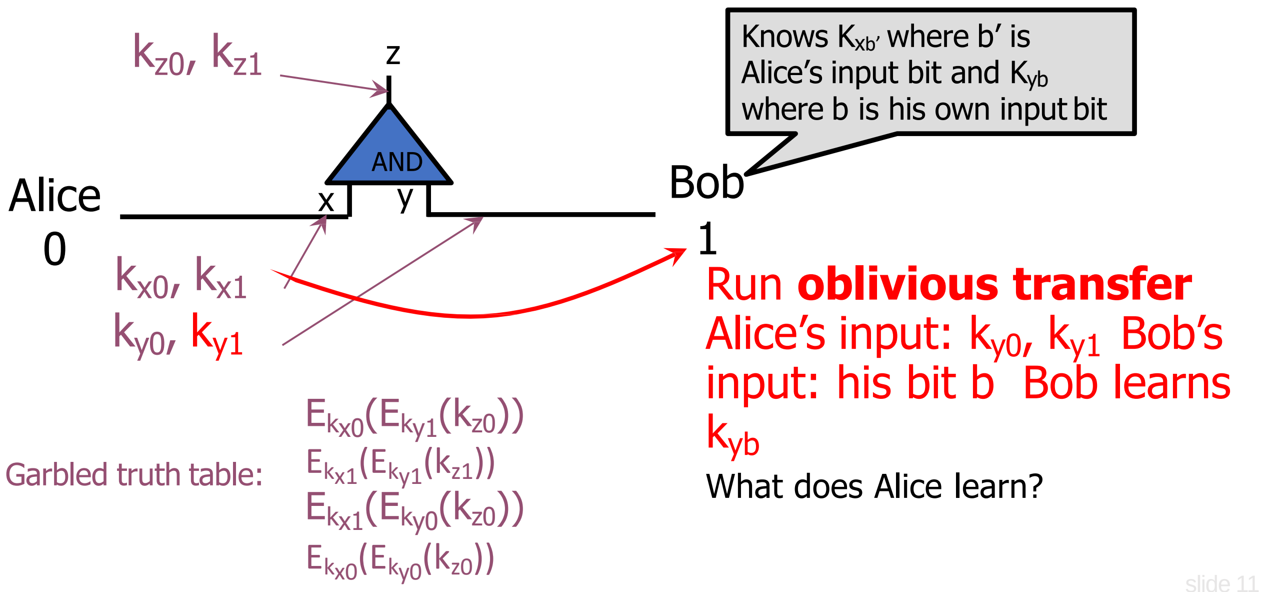
Step 4: Send Keys For Alice's Inputs

- Alice sends the key corresponding to her input bit
- Keys are random, so Bob does not learn what this bit is



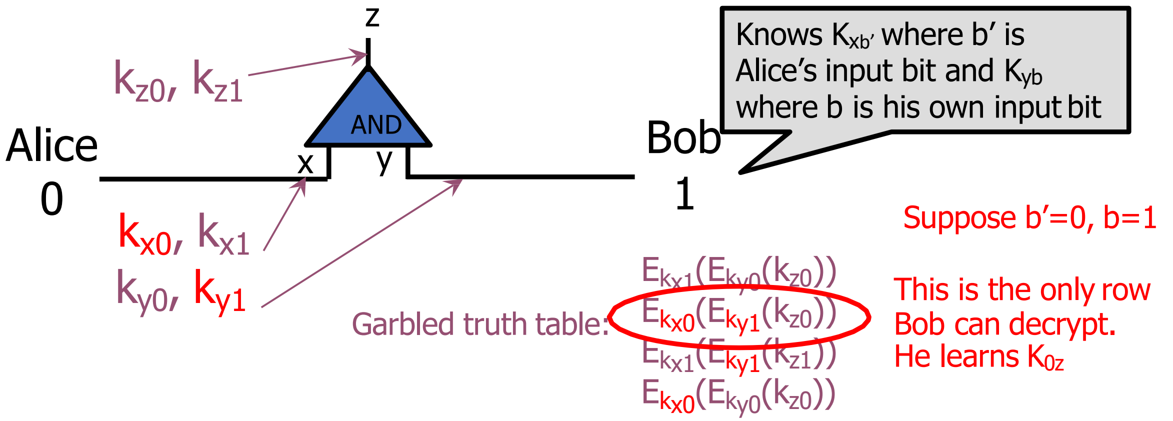
Step 5: Use OT on Keys for Bob's Input

- Alice and Bob run oblivious transfer protocol
  - Alice's input is the two keys corresponding to Bob's wire
  - Bob's input into OT is simply his 1-bit input on that wire



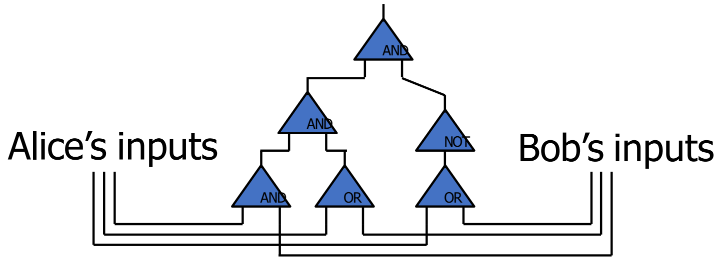
Step 6: Evaluate One Garbled Gate

- Using the two keys that he learned, Bob decrypts exactly one of the output-wire keys
  - Bob does not learn if this key corresponds to 0 or 1
  - Why is this important?



Step 7: Evaluate Entire Circuit

- In this way, Bob evaluates entire garbled circuit
  - For each wire in the circuit, Bob learns only one key
  - It corresponds to 0 or 1 (Bob does not know which)
    - Therefore, Bob does not learn intermediate values (why?)
- Bob tells Alice the key for the final output wire so she can determine if it corresponds to 0 or 1



- Bob does not tell her intermediate wire keys (why?)

Example