Course Business

- I am traveling April 25-May 3rd
  - Will still be available by e-mail to answer questions
- Final Exam Review on Monday, April 24th
- Guest Lectures on April 26 and 28 (TBD)

- Final Exam on Monday, May 1st (in this classroom)
  - Adib will proctor
- Practice Final Exam released soon

Cryptography
CS 555

Topic 37: Yao's Garbled Circuits

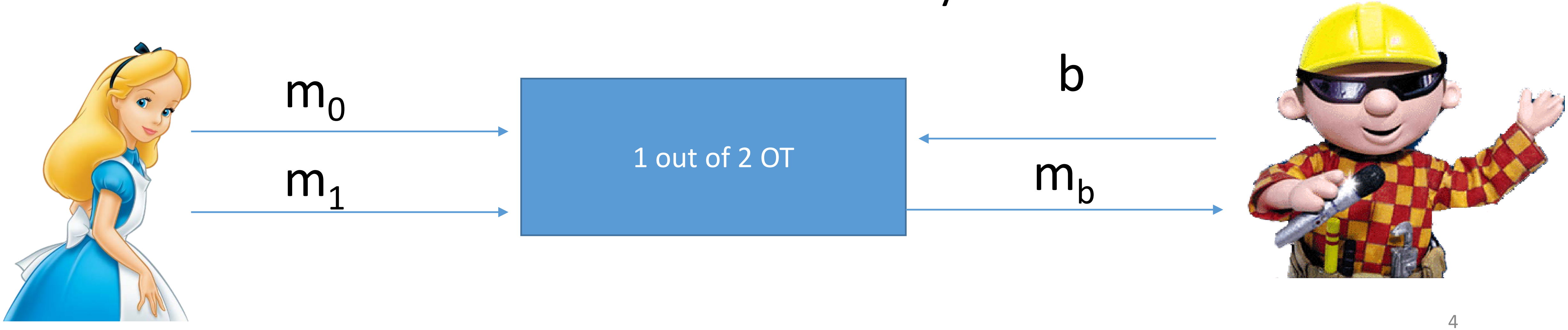
Credit: Some slides from Vitaly Shmatikov

Recap

- Zero-Knowledge Proofs
- Commitment Schemes
- Oblivious Transfer
- ~~Secure Multiparty Computation (Security Models)~~

Recap: Oblivious Transfer (OT)

- 1 out of 2 OT
  - Alice has two messages m_0 and m_1
  - At the end of the protocol
    - Bob gets exactly one of m_0 and m_1
    - Alice does not know which one
- Oblivious Transfer with a Trusted Third Party

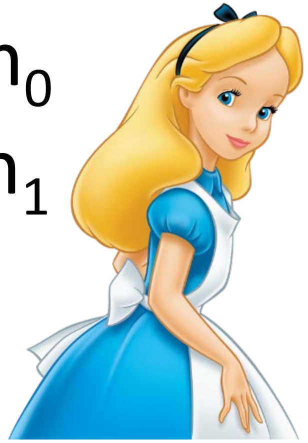


Recap: Bellare-Micali 1-out-of-2-OT protocol

- Oblivious Transfer without a Trusted Third Party
  - g is a generator for a prime order group G_q in which CDH problem is hard

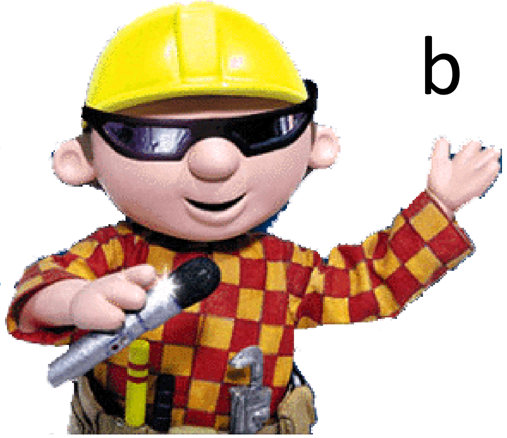
m_0

m_1



$c \leftarrow_R G_q$

b



$k \leftarrow_R Z_q$

$z_b = g^k, z_{1-b} = cg^{-k}$

c

z_0, z_1

$$C_0 = \left[g^{r_0}, H\left(z_0^{r_0}\right) \oplus m_0\right]$$
$$C_1 = \left[g^{r_1}, H\left(z_1^{r_1}\right) \oplus m_1\right]$$

Alice must check that

$$z_1 = c(z_0)^{-1}$$

Bob can decrypt C_b

$$z_b^{r_b} = g^{kr_b}$$

Secure Multiparty Computation (Adversary Models)

- Semi-Honest (“honest, but curious”)
  - All parties follow protocol instructions, but…
  - dishonest parties may be curious to violate privacy of others when possible
- Fully Malicious Model
  - Adversarial Parties may deviate from the protocol arbitrarily
    - Quit unexpectedly
    - Send different messages
  - It is much harder to achieve security in the fully malicious model
- Convert Secure Semi-Honest Protocol into Secure Protocol in Fully Malicious Mode?
  - Tool: Zero-Knowledge Proofs

Voting in the Semi-Honest Model

Question: is cryptography awesome?

y=1 (yes)

$$m_6 = m_5 - R_{Alice,Bob} \ \ mod\ 5 = x + y + z$$

$$m_3 = m_2 + R_{Alice,Bob} \ \ mod\ 5$$

x=1 (yes)

$$m_1 = x + R_{Bob,Devil} \ \ mod\ 5$$

$$m_4 = m_3 - R_{Bob,Devil} \ \ mod\ 5$$

$$m_6 = x + y + z = 2$$

z=0 (no)

$$m_2 = z + m_1 + R_{Devil,Alice} \ mod\ 5$$

$$m_5 = m_5 - R_{Devil,Alice} \ mod\ 5$$

Malicious Model?

Question: is cryptography awesome?

y=1 (yes)

$m_6 = m_5 - R_{Alice,Bob} \ \ mod\ 5 = x + y + z - 2$

$m_3 = m_2 + R_{Alice,Bob} \ \ mod\ 5$

x=1 (yes)

$m_1 = x + R_{Bob,Devil} \ \ mod\ 5$

$m_4 = m_3 - R_{Bob,Devil} \ \ mod\ 5$

$m_6 = x + y + z - 2 = 0$

z=0 (no)

$m_2 = z - 2 + m_1 + R_{Devil,Alice} \ mod\ 5$

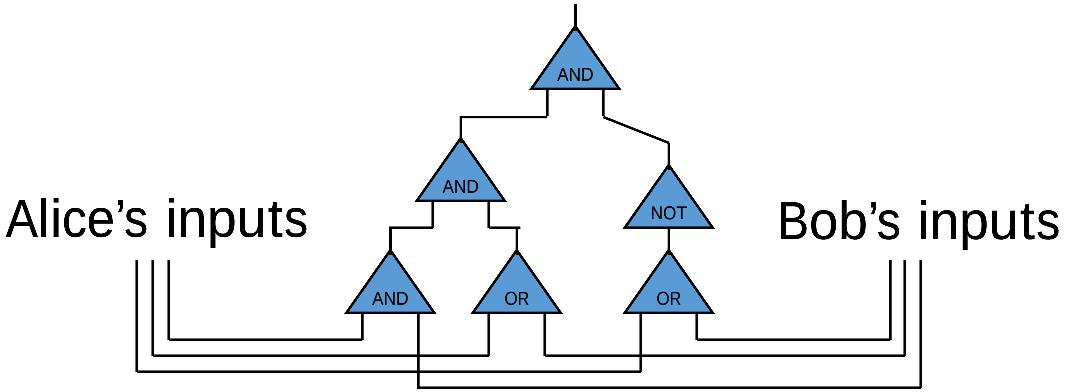
$m_5 = m_5 - R_{Devil,Alice} \ mod\ 5$

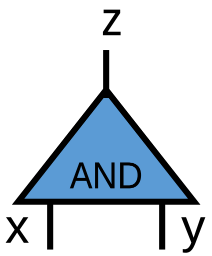
Yao's Protocol

Vitaly Shmatikov

Yao's Protocol

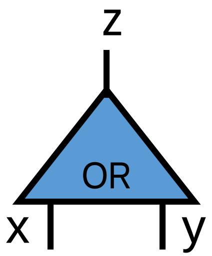
- Compute any function securely
  - … in the semi-honest model
- First, convert the function into a boolean circuit





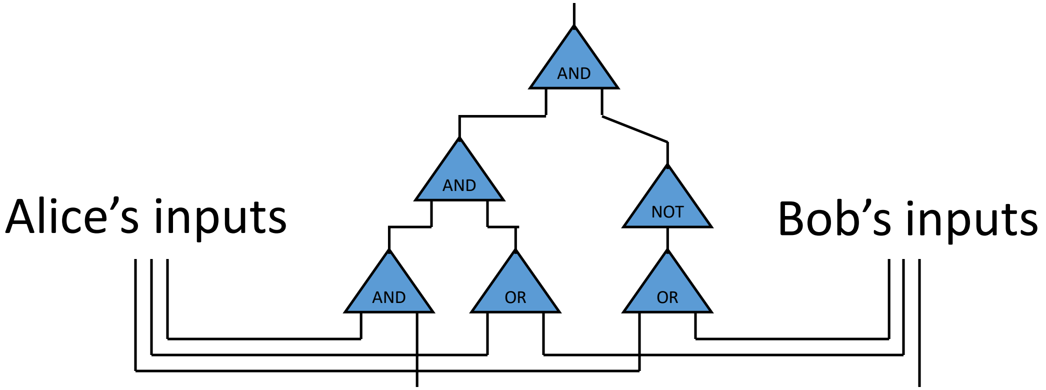
Truth table:

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



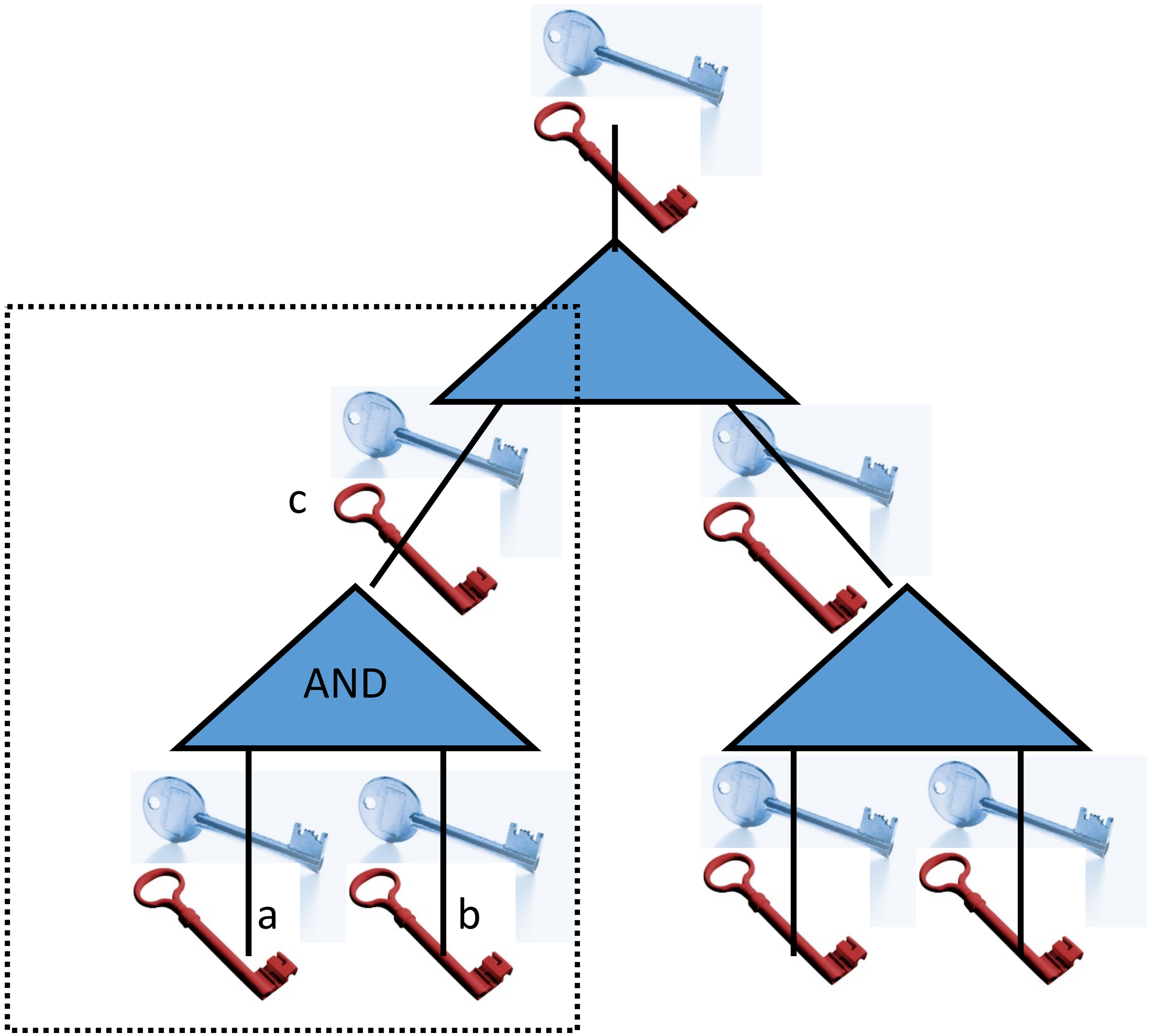
Overview:

1. Alice prepares "garbled" version **C'** of C
2. Sends "encrypted" form **x'** of her input x
3. Allows bob to obtain "encrypted" form **y'** of his input y
4. Bob can compute from **C'**,**x'**,**y'** the "encryption" **z'** of z=C(x,y)
5. Bob sends **z'** to Alice and she decrypts and reveals to him z

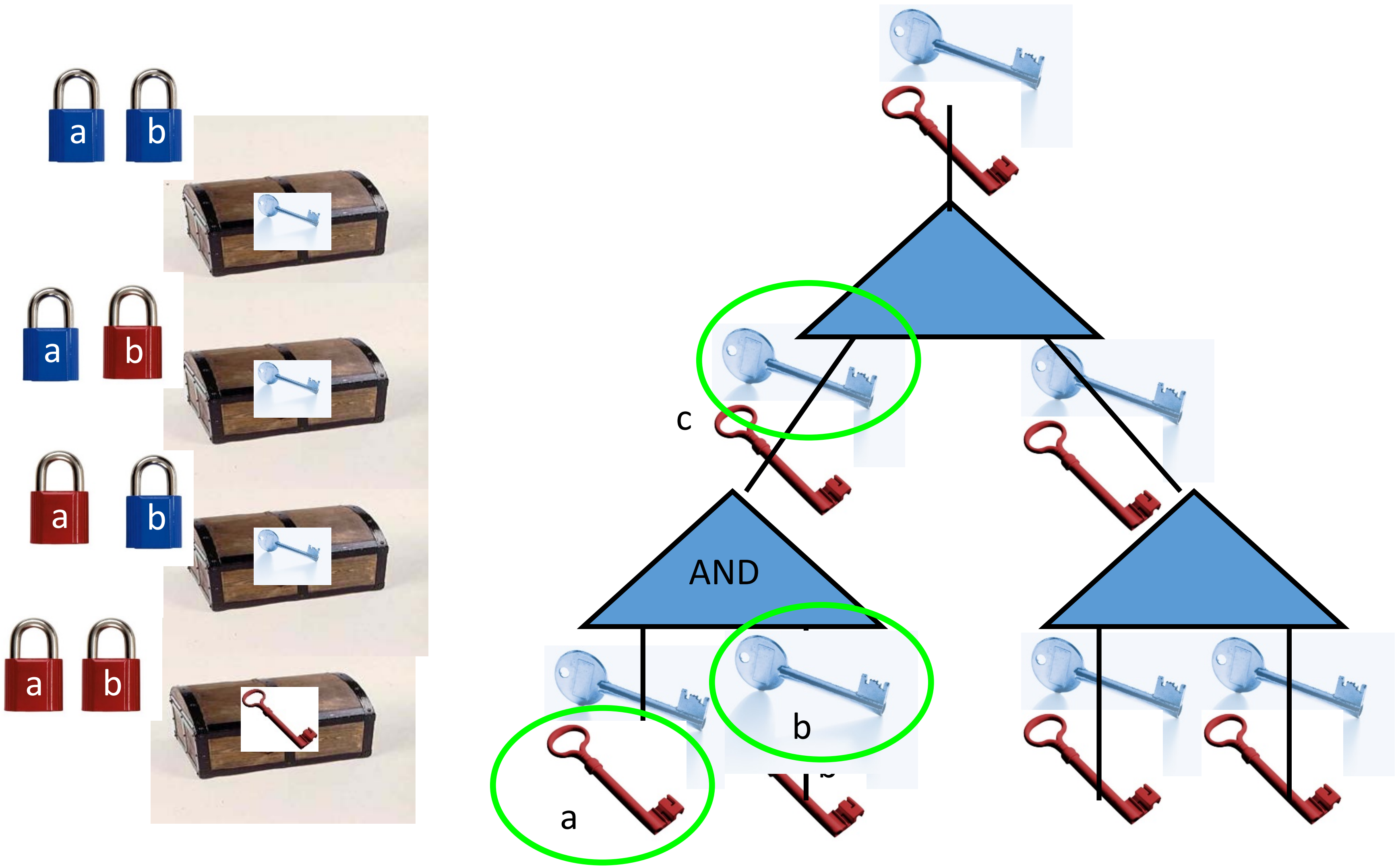
Crucial properties:

1. Bob never sees Alice's input x in unencrypted form.
2. Bob can obtain encryption of y without Alice learning y.
3. Neither party learns intermediate values.
4. Remains secure even if parties try to cheat.

Intuition

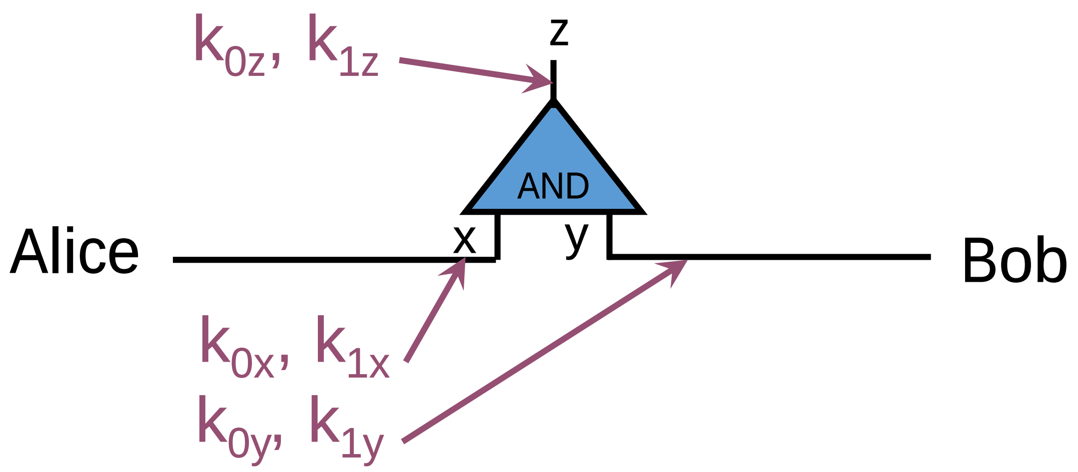


Intuition



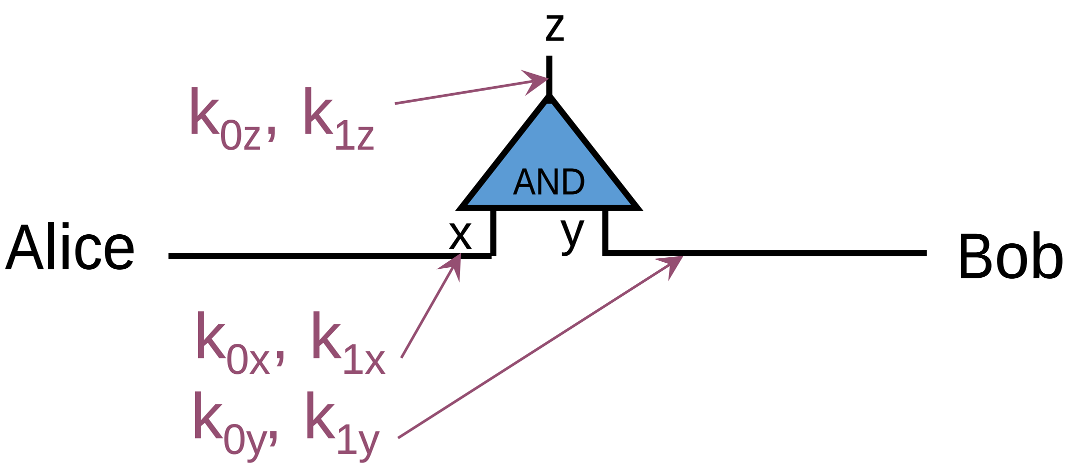
1: Pick Random Keys For Each Wire

- Next, evaluate one gate securely
  - Later, generalize to the entire circuit
- 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



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



Original truth table:

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

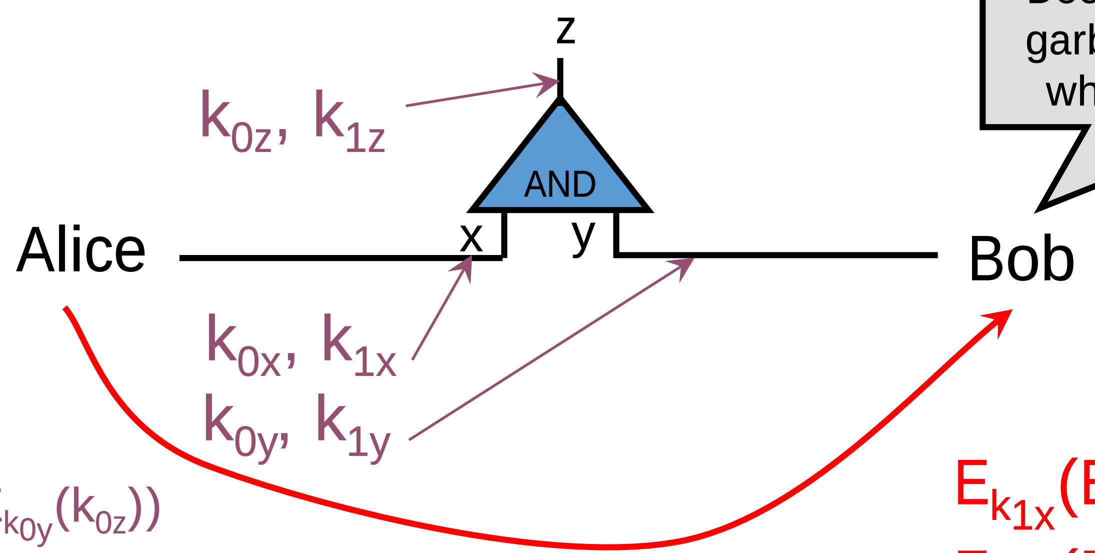
Encrypted truth table:

$$E_{k0x}(E_{k0y}(k_{0z}))$$
$$E_{k0x}(E_{k1y}(k_{0z}))$$
$$E_{k1x}(E_{k0y}(k_{0z}))$$
$$E_{k1x}(E_{k1y}(k_{1z}))$$

3: Send Garbled Truth Table

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

Does not know which row of garbled table corresponds to which row of original table



$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$

$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$

$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

Garbled truth table:

$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$

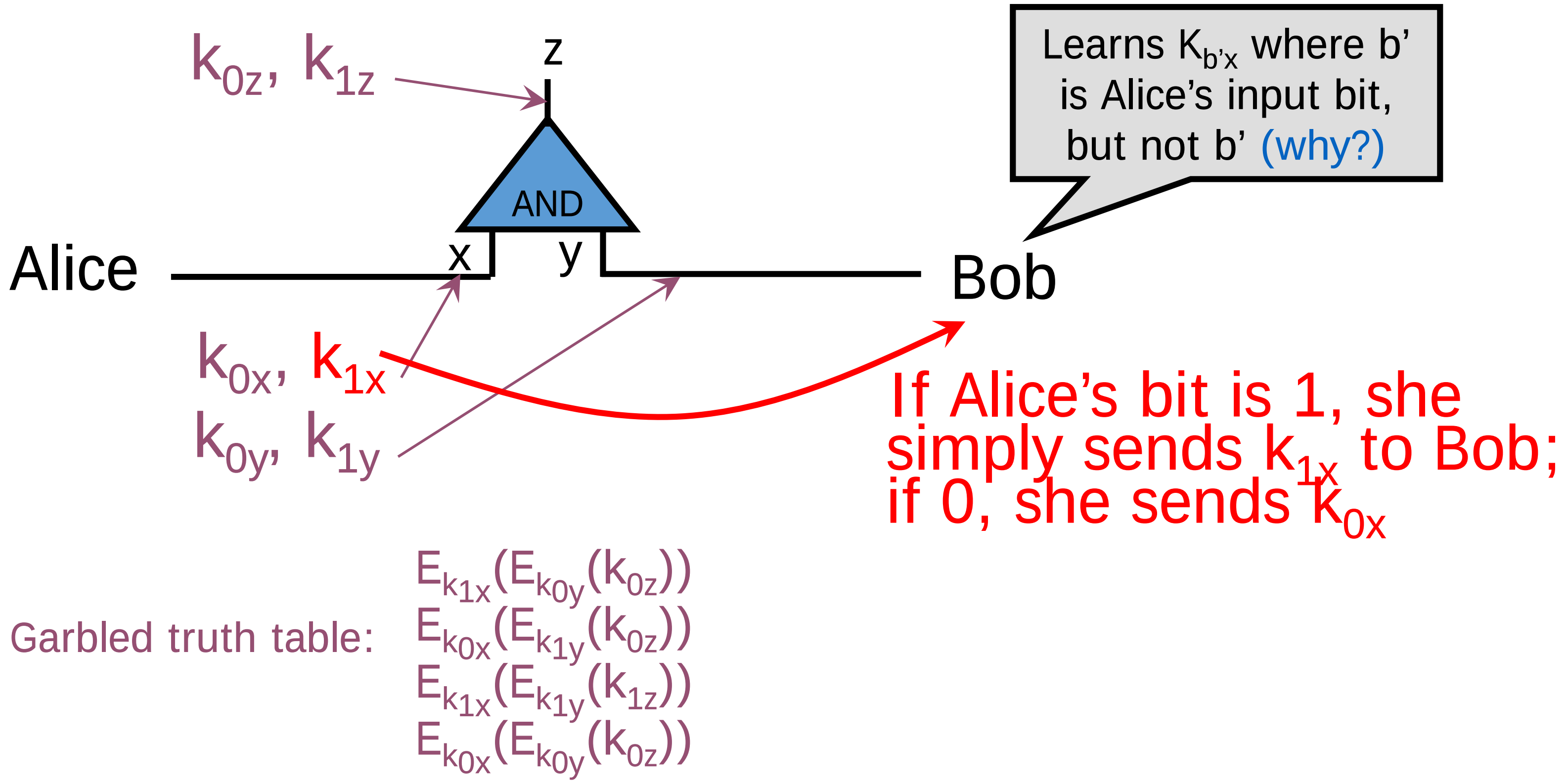
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$

$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

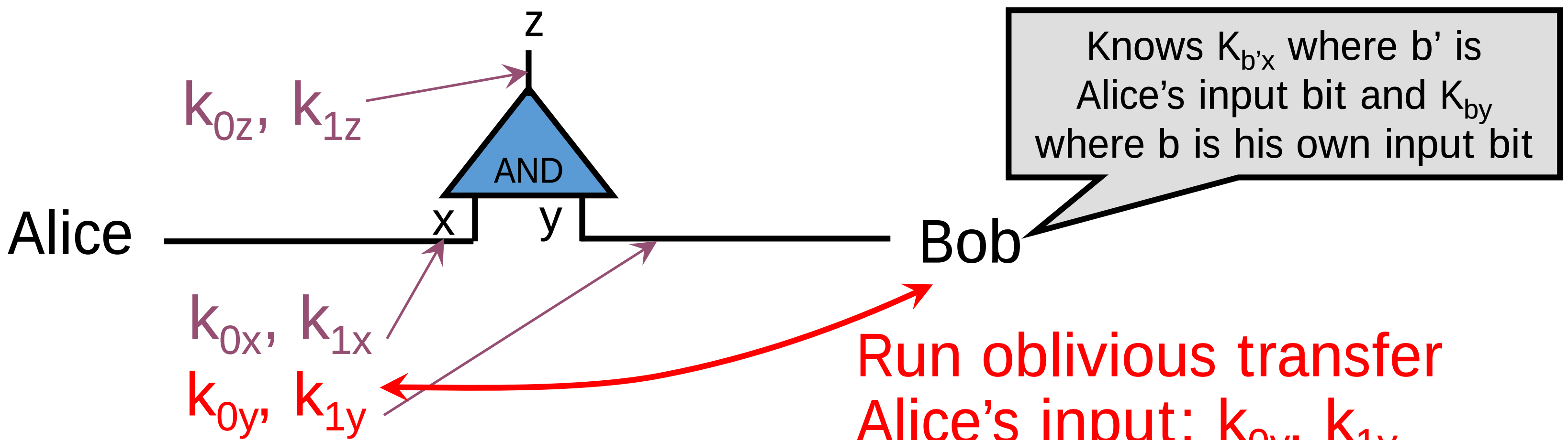
4: Send Keys For Alice's Inputs

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



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

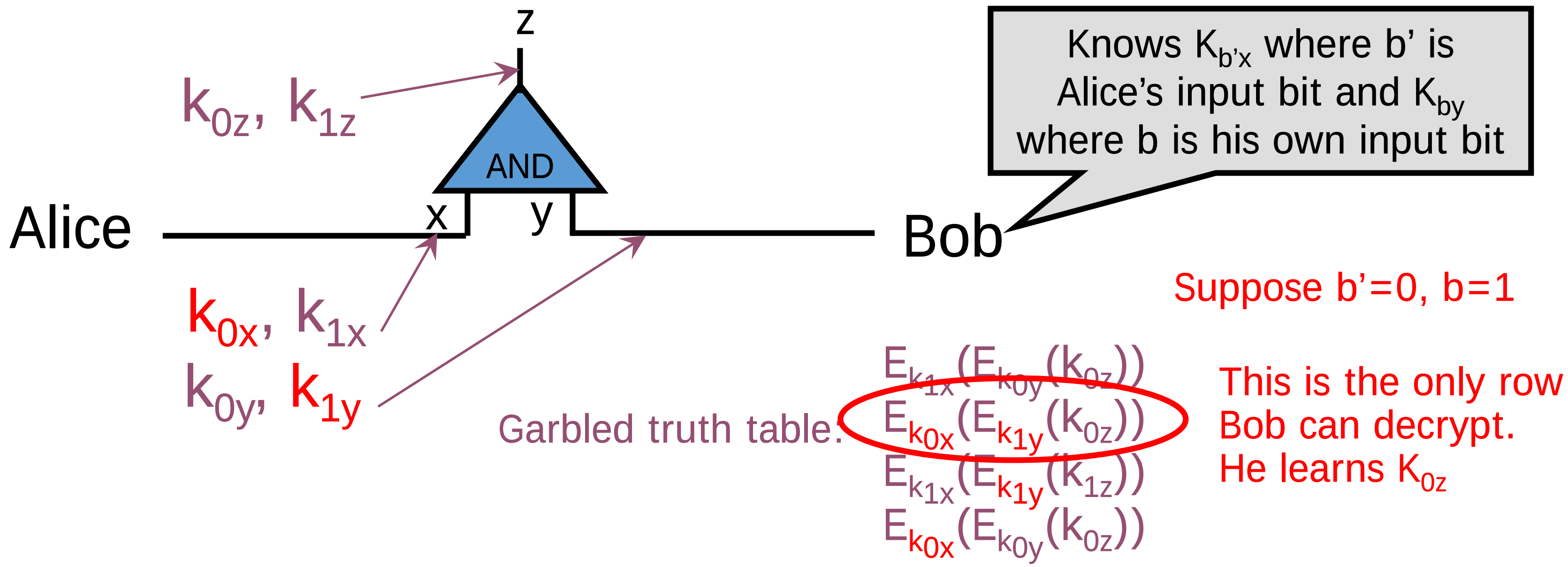


Garbled truth table:
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

Run oblivious transfer
Alice's input: k_{0y}, k_{1y}
Bob's input: his bit b
Bob learns k_{by}
What does Alice learn?

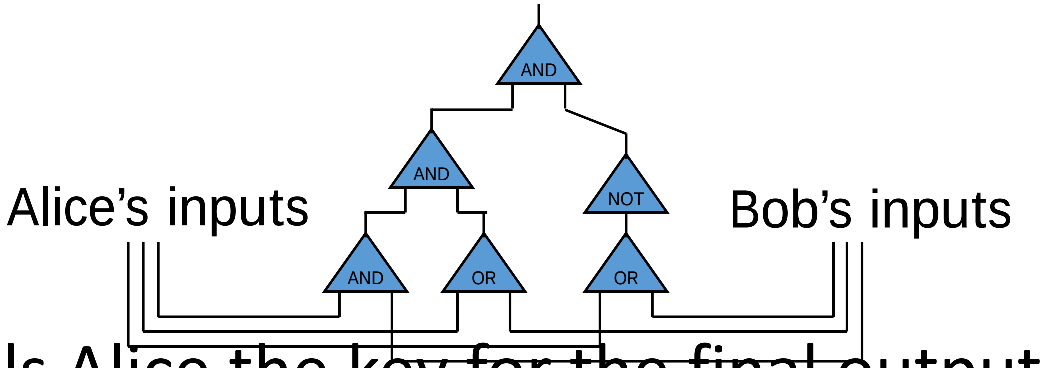
6: Evaluate Garbled Gate

- 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?



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 and she tells him if it corresponds to 0 or 1
  - Bob does <u>not</u> tell her intermediate wire keys (why?)

Brief Discussion of Yao's Protocol

- Function must be converted into a circuit
  - For many functions, circuit will be huge
- If m gates in the circuit and n inputs from Bob, then need 4m encryptions and n oblivious transfers
  - Oblivious transfers for all inputs can be done in parallel
- Yao's construction gives a constant-round protocol for secure computation of any function in the semi-honest model
  - Number of rounds does not depend on the number of inputs or the size of the circuit!

Computational Indistinguishability

Definition: We say that an ensemble of distributions $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for all PPT distinguishers D, there is a negligible function negl(n), such that we have

$$Adv_{D,n} = \left|Pr_{s\leftarrow \mathrm{X}_\ell}[D(s)=1] - Pr_{s\leftarrow \mathrm{Y}_\ell}[D(s)=1]\right| \le negl(n)$$

Notation: $\{X_n\}_{n\in\mathbb{N}} \equiv_C \{Y_n\}_{n\in\mathbb{N}}$ means that the ensembles are computationally indistinguishable.

Security (Semi-Honest Model)

- Let $B_n = trans_B(n, x, y)$ (resp. $A_n = trans_A(n, x, y)$) be the protocol transcript from Bob's perspective (resp. Alice's perspective) when his input is x and Alice's input is y (assuming that Alice follows the protocol).
- **Security**: Assuming that Alice and Bob are both semi-honest (follow the protocol) then there exist PPT simulators S_A and S_B s.t.

$$\{A_n\}_{n\in\mathbb{N}} \equiv_C \{S_A(n, f_A(x, y)\ )\}_{n\in\mathbb{N}}$$
$$\{B_n\}_{n\in\mathbb{N}} \equiv_C \{S_B(n, f_B(x, y))\}_{n\in\mathbb{N}}$$

- **Remark**: Simulator S_A is only shown Alice's output $f_A(x, y)$ (similarly, S_B is only shown Bob's output $f_B(x, y)$)

Security (Semi-Honest Model)

- **Security**: Assuming that Alice and Bob are both semi-honest (follow the protocol) then there exist PPT simulators S_A and S_B s.t.

$$\{A_n\}_{n\in\mathbb{N}} \equiv_C \{S_A(n, x, f_A(x, y)\ )\}_{n\in\mathbb{N}}$$
$$\{B_n\}_{n\in\mathbb{N}} \equiv_C \left\{S_B\left(n, y, f_B(x, y)\right)\right\}_{n\in\mathbb{N}}$$

- **Remark**: Simulator S_A is only shown Alice's output $f_A(x, y)$ (similarly, S_B is only shown Bob's output $f_B(x, y)$)

Theorem (informal): If the oblivious transfer protocol is secure, and the underlying encryption scheme is CPA-secure then Yao's protocol is secure in the semi-honest adversary model.

Fully Malicious Security?

1. Alice could initially garble the wrong circuit C(x,y)=y.
2. Given output of C(x,y) Alice can still send Bob the output f(x,y).
3. Can Bob detect/prevent this?

Fix: Assume Alice and Bob have both committed to their input: c_A=com(xlr_A) and c_B=com(ylr_B).

- Alice and Bob can use zero-knowledge proofs to convince other party that they are behaving honestly.
- **Example**: After sending a message A Alice proves that the message she just sent is the same message an honest party would have sent with input x s.t. c_A=com(xlr_A)
- Here we assume that Alice and Bob have both committed to correct inputs (Bob might use y which does not represent his real vote etc… but this is not a problem we can address with cryptography)

Fully Malicious Security

- Assume Alice and Bob have both committed to their input: c_A=com(x|r_A) and c_B=com(y|r_B).
  - Here we assume that Alice and Bob have both committed to correct inputs (Bob might use y which does not represent his real vote etc… but this is not a problem we can address with cryptography)
  - Alice has c_B and can unlock c_A
  - Bob has c_A and can unlock c_B

1. Alice sets C_f = GarbleCircuit(f,r).
   1. Alice sends to Bob.
   2. Alice convinces Bob that C_f = GarbleCircuit(f,r) for some r (using a zero-knowledge proof)
2. For each original oblivious transfer if Alice's inputs were originally x_0,x_1
   1. Alice and Bob run OT with y_0,y_1 where y_i=$Enc_K(x_i)$
   2. Bob uses a zero-knowledge proof to convince Alice that he received the correct y_i (e.g. matching his previous commitment c_B)
   3. Alice sends K to Bob who decrypts y_i to obtain x_i

Next Class: Differential Privacy

- No Reading ☺