Course Business

- **Homework 5 Released**
- Bonus Problem (10 Points)

Homework 4 Statistics

Minimum Value	**26.00**
Maximum Value	**110.00**
Range	**84.00**
Average	**86.16**
Median	**90.00**
Standard Deviation	**19.18**

Cryptography
CS 555

Week 14:

- Digital Signatures Continued
- Multiparty Computation
- Yao's Garbled Circuits

Readings: Katz and Lindell Chapter 10 & Chapter 11.1-11.2, 11.4

Plain RSA Signatures

- Plain RSA
- Public Key (pk): N = pq, e such that $\text{GCD}\big(\text{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$

$$\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m}) = \boldsymbol{m^d \ mod \ N}$$

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if\ m} = [\sigma^{\boldsymbol{e}} \ \boldsymbol{mod\ N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

- Verification Works because

$$\left[\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m})^e \text{ mod N}\right] = [m^{ed} \text{mod N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right] = m$$

No Message Attack

- **Goal:** Generate a forgery using only the public key
  - No intercepted signatures required
- Public Key (pk): $\mathrm{N} = \mathrm{pq}, \mathrm{e}$ such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Pick random $\sigma \in \mathbb{Z}^*_\mathrm{N}$
- Set $\boldsymbol{m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}]$.
- Output (m, σ)

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\ \boldsymbol{m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}\,(m) = H(m)^d \; mod \; N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Remark: The domain of H (e.g.,SHA3) may be shorter than $\mathbb{Z}_N^*$.

Solution: Repeated application of H.

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \to \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}(m) = H(m)^d \ mod\ N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Proof Sketch: Given an RSA-Inversion challenge $\mathrm{c} = r^e \ mod\ N$ we will program the value $\mathrm{H(m)} = c \in \mathbb{Z}_N^*$ into the random oracle to trick the signature attacker into revealing $\mathrm{Sign}_{sk}(m) = r^{ed} = r \ mod\ N$.

Digital Signature Algorithm (DSA)

DSA: $\langle \boldsymbol{g} \rangle$ is subgroup of $\mathbb{Z}_p^*$ of order q

ECDSA: $\langle \boldsymbol{g} \rangle$ is order q subgroup of elliptic curve

- Secret key is x, public key is h=g^x along with generator g (of order q)
- Sign_{sk}(m)
  - Pick random $\left(k \in \mathbb{Z}_q\right)$ and set $r = F\left(g^k\right) = [g^k \: mod \: q]$
  - Compute $s := [k^{-1}(xr + H(m)) \: mod \: q]$
  - Output signature (r,s)
- Vrfy_{pk}(m,(r,s)) check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

Digital Signature Algorithm (DSA)

- $\text{Sign}_{sk}(m)$
  - Pick random $\left(k \in \mathbb{Z}_q\right)$ and set $r = F\left(g^k\right) = [g^k \mod q]$
  - Compute $s \coloneqq [k^{-1}(xr + H(m)) \mod q]$
  - Output signature (r,s)
- $\text{Vrfy}_{pk}(m,(r,s))$ check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

$$= F\left(g^{H(m)k(xr+H(m))^{-1}} g^{xrk(xr+H(m))^{-1}}\right)$$

$$= F\left(g^{(H(m)+xr)k(xr+H(m))^{-1}}\right)$$

$$= F\left(g^k\right) \coloneqq r$$

Digital Signature Algorithm (DSA)

- Secret key is x, public key is $h=g^x$ along with generator g (of order q)
- $\text{Sign}_{sk}(m)$
  - Pick random $(k \in \mathbb{Z}_q)$ and set $r = F(g^k) = [g^k \; mod \; q]$
  - Compute $s \coloneqq [k^{-1}(xr + H(m)) \; mod \; q]$
  - Output signature (r,s)
- $\text{Vrfy}_{pk}(m,(r,s))$ check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

Theorem: If H and F are modeled as random oracles then DSA is secure.

Weird Assumption?

- **Theory:** DSA Still lack compelling proof of security from standard crypto assumptions
- **Practice:** DSA has been used/studied for decades without attacks

Digital Signature Algorithm (DSA)

- Secret key is x, public key is $h=g^x$
- $\mathrm{Sign}_{sk}(m)$
  - Pick random $\left(k \in \mathbb{Z}_q\right)$ and set $r = F\left(g^k\right) = [g^k \; mod \; q]$
  - Compute s $:= [k^{-1}(xr + H(m)) \; mod \; q]$
  - Output signature (r,s)
- $\mathrm{Vrfy}_{pk}(m,(r,s))$ check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

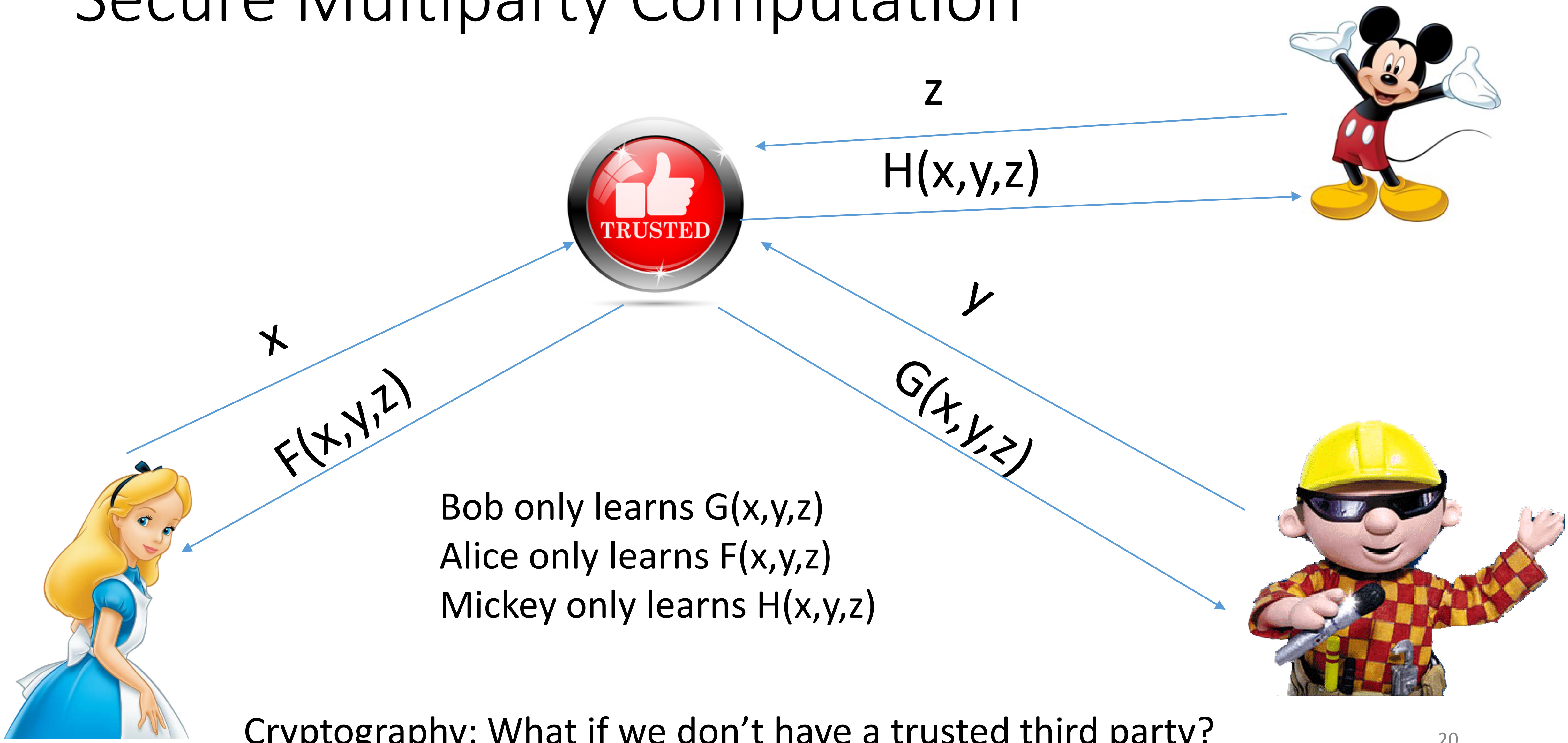
Remark: If signer signs two messages with same random $k \in \mathbb{Z}_q$ then attacker can find secret key sk!

- **Theory:** Shouldn't happen
- **Practice:** Will happen if a weak PRG is used
- Sony PlayStation (PS3) hack in 2010.

Certificate Authority

- Trusted Authority (CA)
  - $m_{CA \rightarrow Amazon}$="Amazon's public key is pk_{Amazon} (date,expiration,###)"
  - $cert_{CA \rightarrow Amazon} = Sign_{SK_{CA}}(m)$
- Delegate Authority to other CA_1
  - Root CA signs m= "CA_1 public key is pk_{CA1} (date,expiration,###) can issue certificates"
  - Verifier can check entire certification chain
- Revocation List Signed Daily
- Decentralized Web of Trust (PGP)

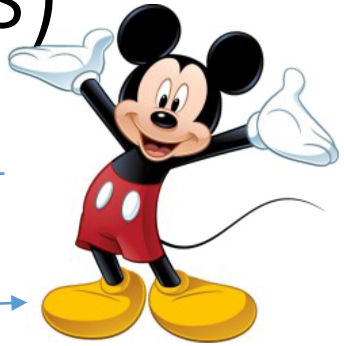
Secure Multiparty Computation



Bob only learns G(x,y,z)
Alice only learns F(x,y,z)
Mickey only learns H(x,y,z)

Cryptography: What if we don't have a trusted third party?

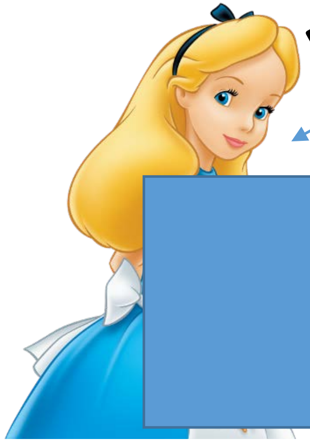
Secure Multiparty Computation (Crushes)



Bob only learns G(x,y,z)
Alice only learns F(x,y,z)
Mickey only learns H(x,y,z)

Alice can infer Y from F(x,y,z) and Bob can infer X from H(x,y,z).
But Alice/Bob cannot infer anything about Z.
Mickey cannot infer y, and learns that x≠ "Mickey"

Secure Multiparty Computation (Crushes)



x="Bob"

F(x,y,z)="match Bo

Bob o

Alice c

Micke

Key Point: The output H(x,y,z) may leak info about inputs. Thus, we cannot prevent Mickey from learning anything about x,y but Mickey should not learn anything else besides H(x,y,z)!

Though Question: How can we formalize this property?

Mickey cannot infer y, and learns that x$\neq$ "Mickey"

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

Computational Indistinguishability

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

$$Adv_{D,n} = \left|Pr_{s\leftarrow X_\ell}[D(s) = 1] - Pr_{s\leftarrow Y_\ell}[D(s) = 1]\right| \leq negl(n)$$

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)

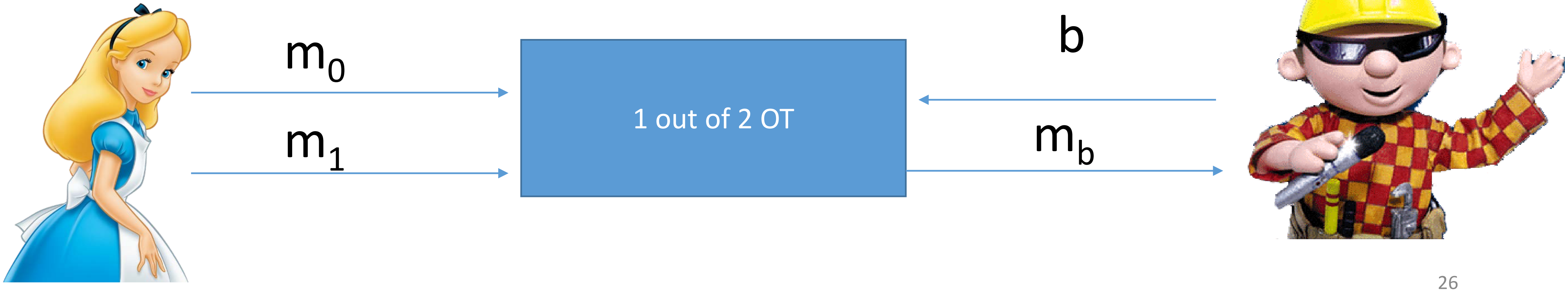
- Let $B_n = trans_B(n, x, y)$ (resp. $A_n = trans_A(n, x, y)$) be the protocol transcript from Bob's perspective (resp. Alice's perspective) when his input is y and Alice's input is x (assuming that Alice follows the protocol).
- **Security**: Assuming that Alice and Bob are both semi-honest (follow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

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

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

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

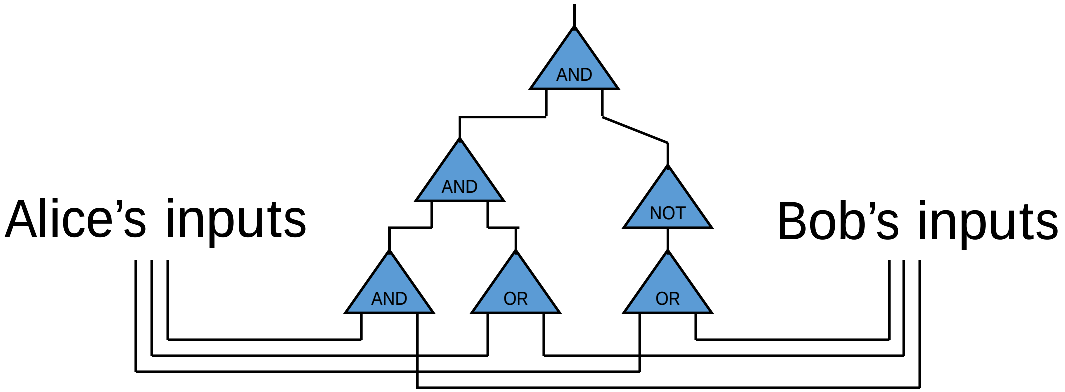


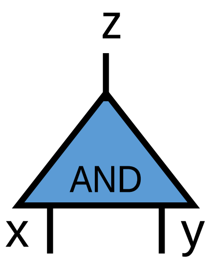
Yao's Protocol

Vitaly Shmatikov

Yao's Protocol

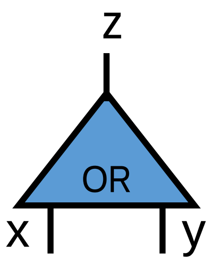
- Compute any function securely
  - … in the semi-honest model
- First, convert the function into a boolean circuit





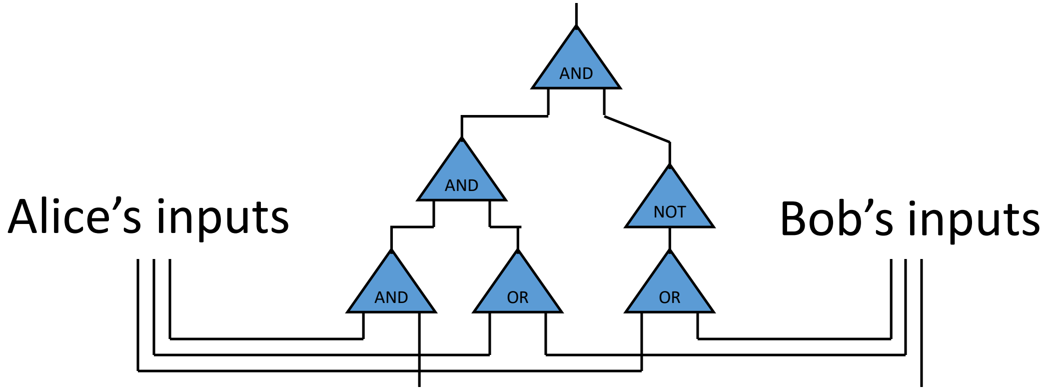
Truth table:

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



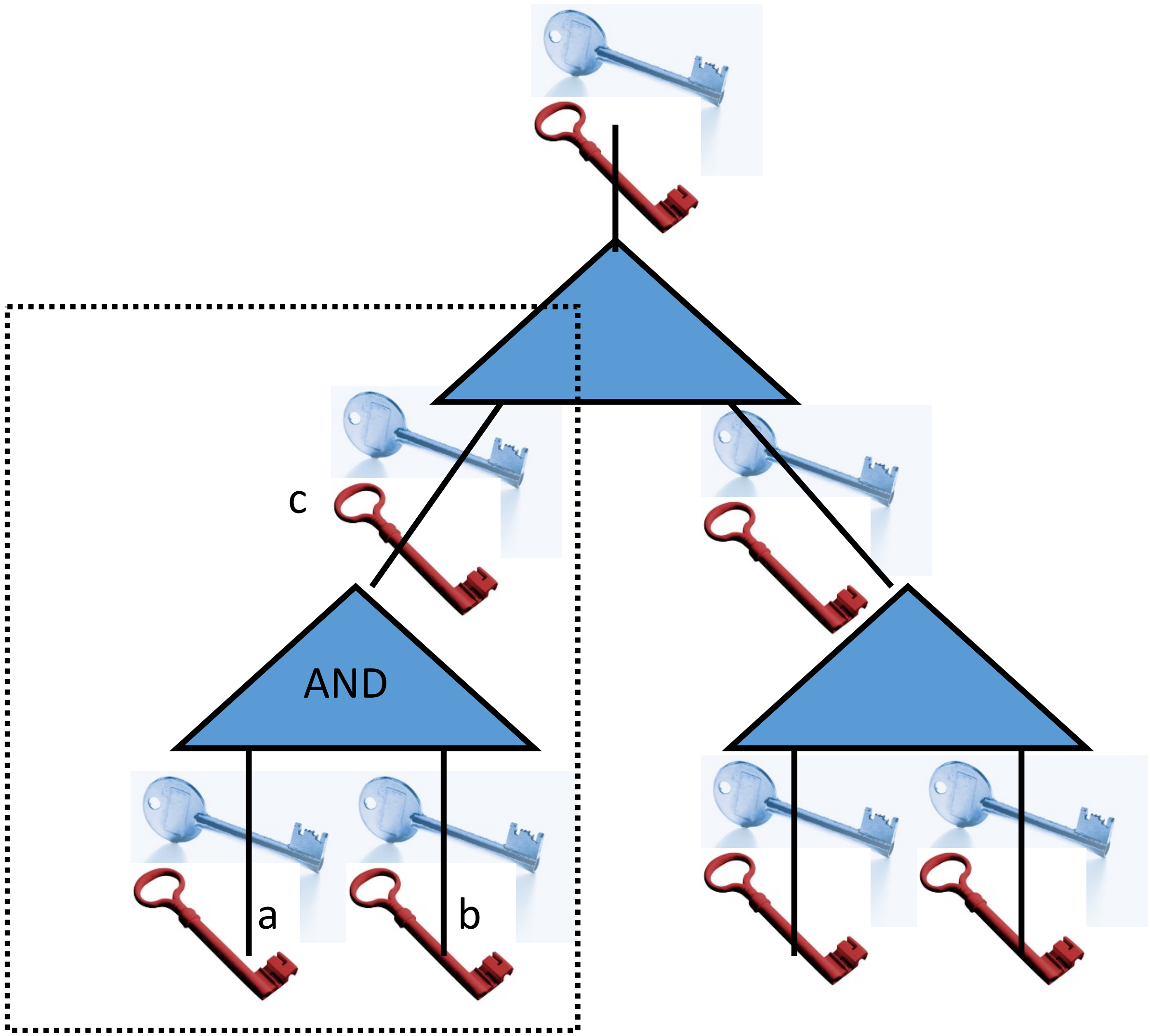
Overview:

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 via OT
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

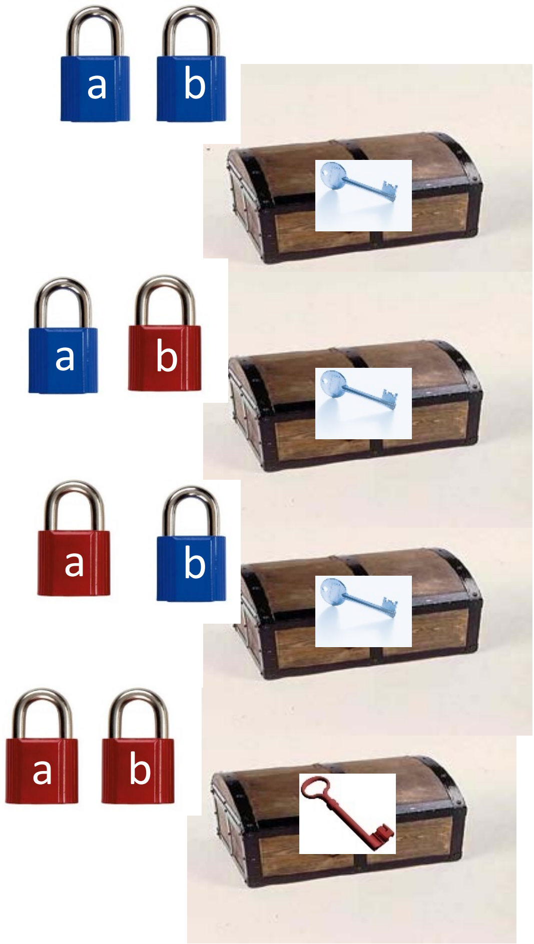
Crucial properties:

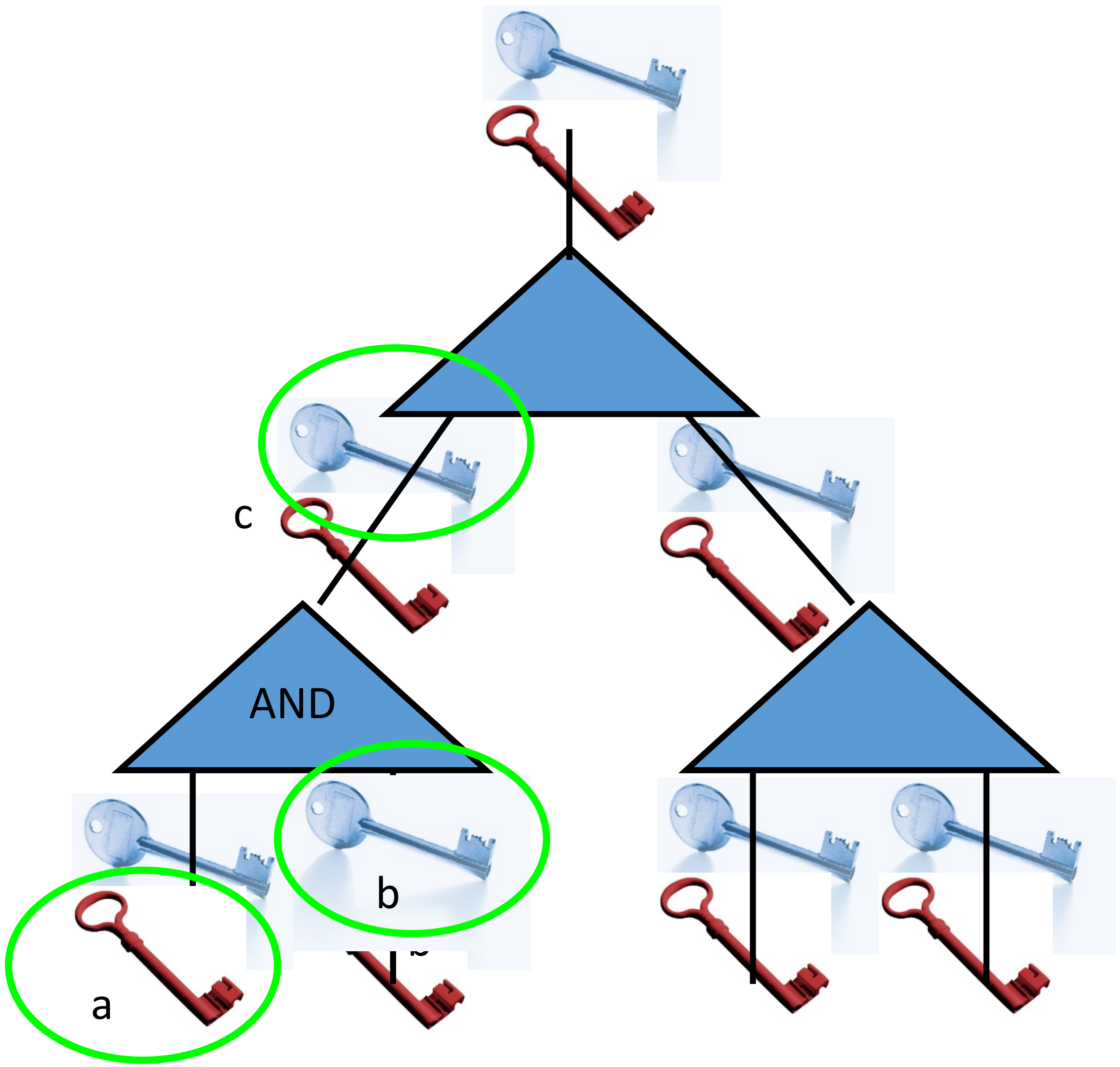
1. Bob never sees Alice's input x in unencrypted form.
2. Bob can obtain encryption of y without Alice learning y.
3. Neither party learns intermediate values.
4. Remains secure even if parties try to cheat.

Intuition



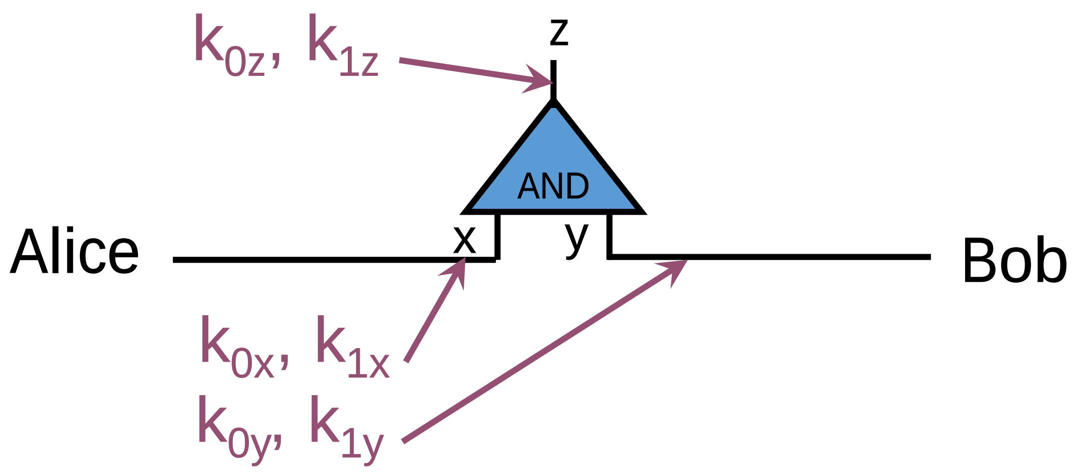
Intuition





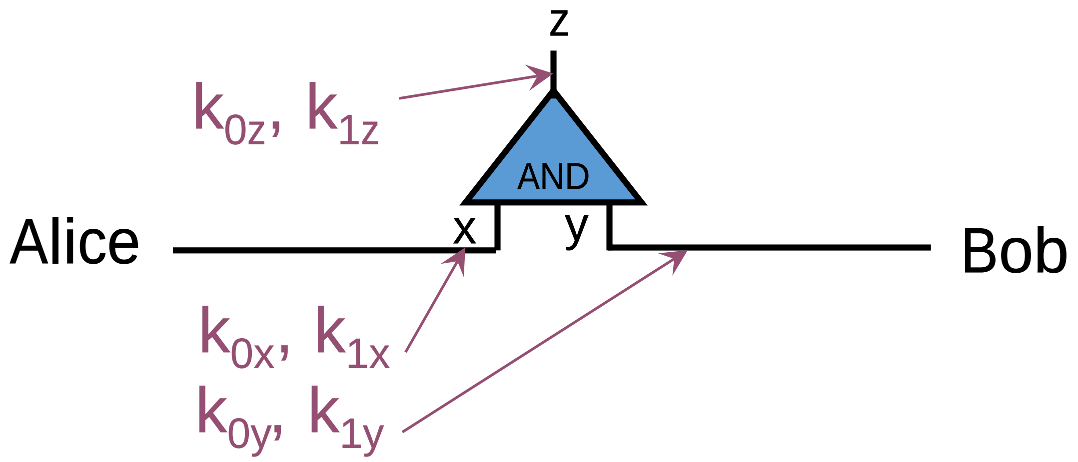
1: Pick Random Keys For Each Wire

- 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:

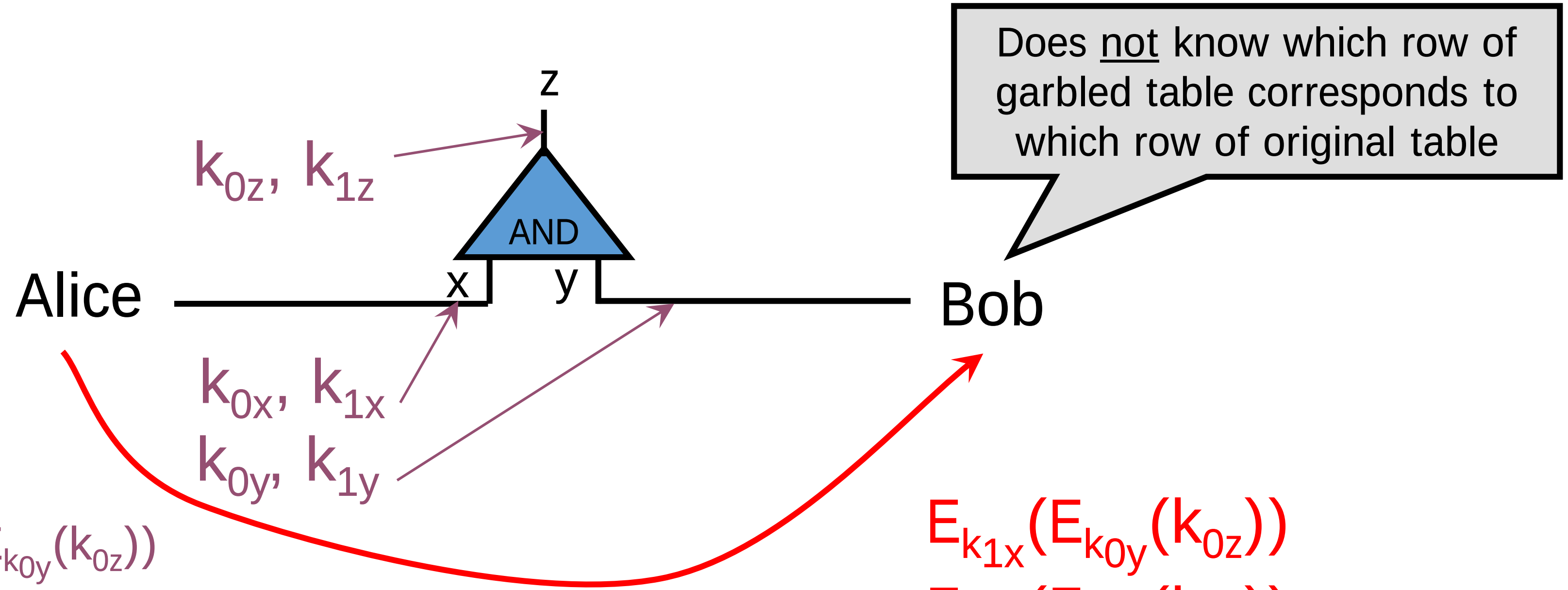
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

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



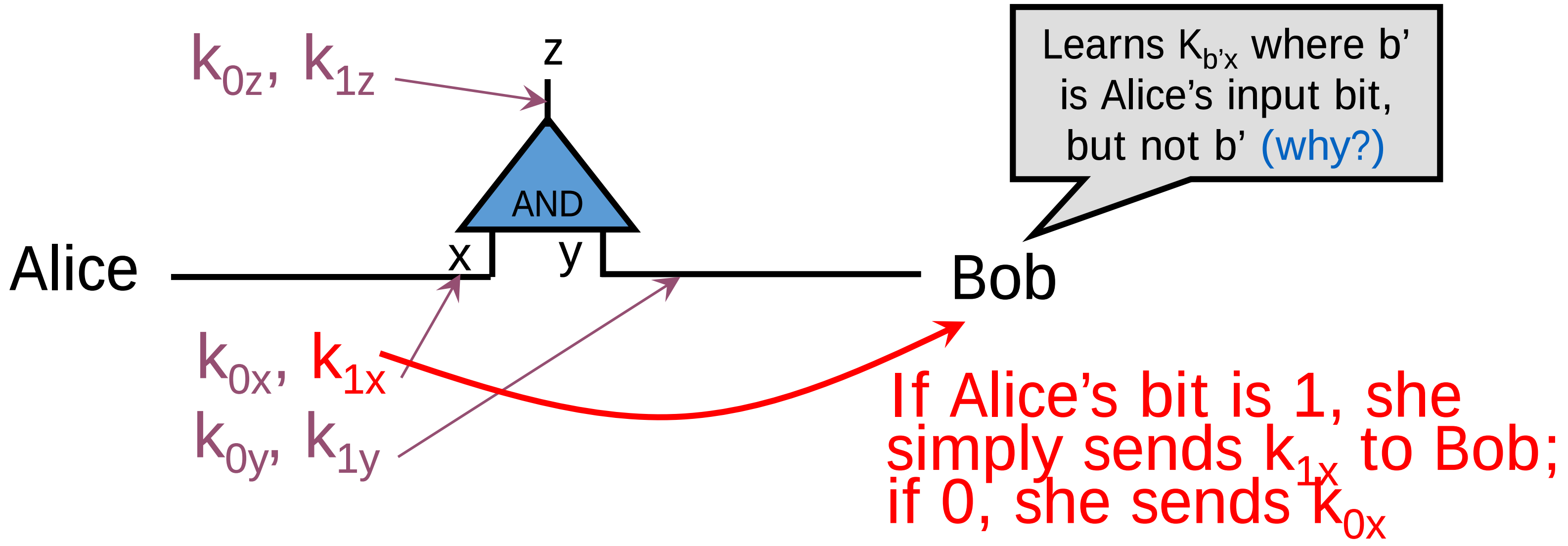
$E_{k0x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k1y}(k_{0z}))$
$E_{k1x}(E_{k0y}(k_{0z}))$
$E_{k1x}(E_{k1y}(k_{1z}))$

Garbled truth table:

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

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

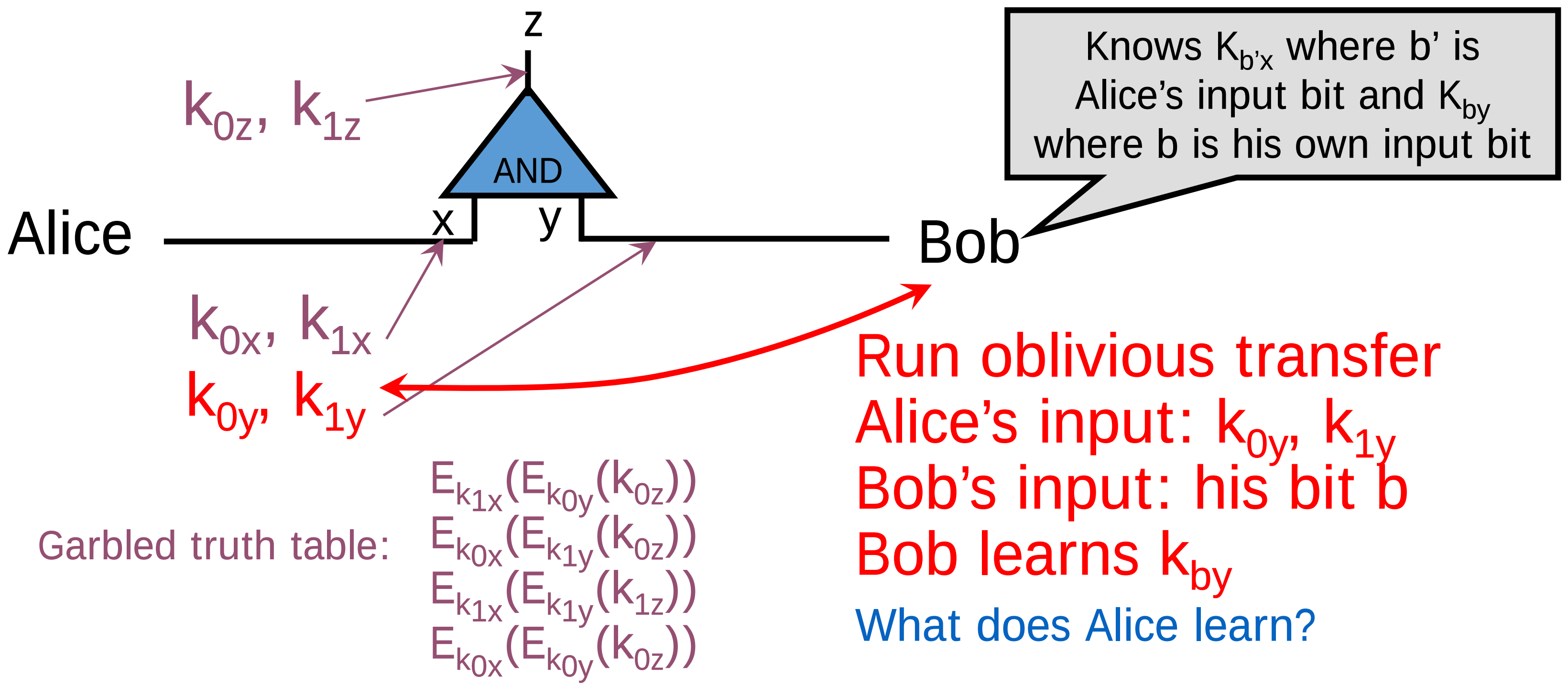


Garbled truth table:

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

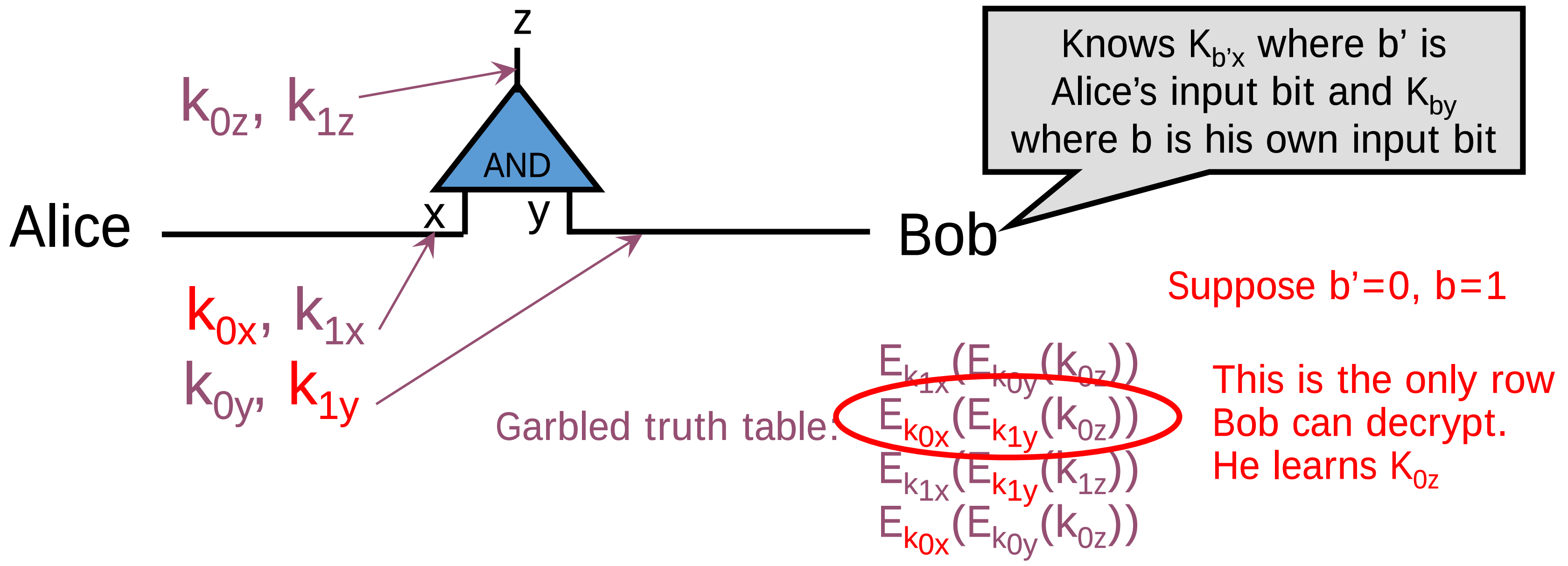
5: Use OT on Keys for Bob’s Input

- 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



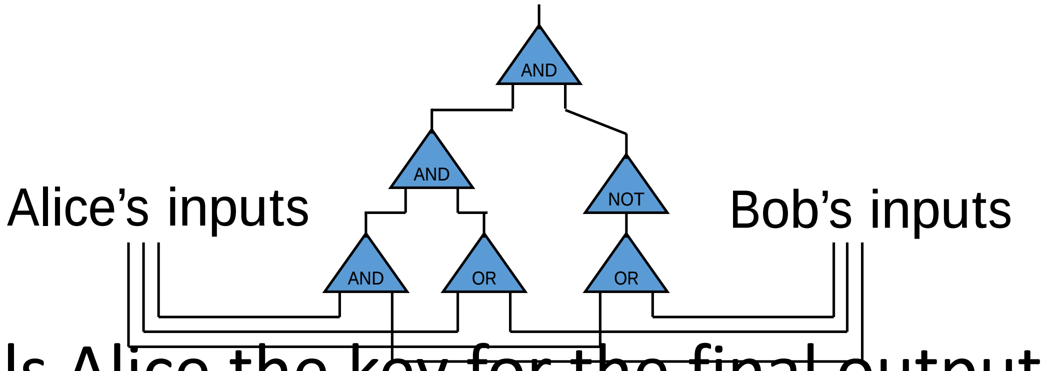
6: Evaluate Garbled Gate

- 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?)

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.

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

- **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.

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!

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(x|r_A) and c_B=com(y|r_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(x|r_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