Cryptography
CS 555

Week 13:

- More Plain RSA Attacks
- Secure Multi-Party Computation (Garbled Circuits)

Reminder: Quiz 5 due tonight (4/14) at 11:30PM on Brigthspace

Readings: Chapter 11.1-11.2, 11.4

Spring 2021

Plain RSA Attacks: Related Messages

- Sender encrypts m and $m + \delta$, where offset δ is known to attacker

- Attacker intercepts

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \; mod \; N$$

and

$$c_2 = \mathrm{Enc}_{pk}(m + \delta) = (m + \delta)^e \; mod \; N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1 \, mod \; N$$

and

$$f_2(x) = (x + \delta)^e - c_2 \, mod \; N$$

More Attacks: Encrypting Related Messages

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \ mod\ N$$
$$c_2 = \mathrm{Enc}_{pk}(m+\delta) = (m+\delta)^e \ mod\ N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1\, mod\ N$$

and

$$f_2(x) = (x+\delta)^e - c_2\, mod\ N$$

- Both polynomials have a root at x=m, thus (x-m) is a factor of both polynomials
- The GCD operation can be extended to operate over polynomials ☺
  - Polynomial time in log N and degree e
  - Attack on Plain RSA only works when e is small (often true in practice)
- GCD($f_1(x)$, $f_2(x)$) reveals the common factor (x-m)
  - Can easily extract m from g(x)=(x-m)= GCD($f_1(x)$, $f_2(x)$)

Factor N given $\phi(N)$

- Suppose we are given $N = pq$ and $\phi(N) = (p-1)(q-1)$

- **Idea:** Solve for p using quadratic formula!

$$\phi(N) = (p-1)(q-1) = (p-1)\left(\frac{N}{p} - 1\right)$$

$$p\phi(N) = (p-1)(N-p) \qquad \text{(Multiply by p)}$$

$$p^2 + p(\phi(N) - 1 - N) + N = 0 \qquad \text{(Algebra)}$$

Factor N given $\phi(N)$

- Suppose we are given $N = pq$ and $\phi(N) = (p-1)(q-1)$

- **Idea:** Solve for p using quadratic formula!

$$p^2 + p(\phi(N) - 1 - N) + N = 0 \qquad \text{(Algebra)}$$

$$p = \frac{-(\phi(N) - 1 - N) \pm \sqrt{(\phi(N) - 1 - N)^2 - 4N}}{2}$$

$$\text{(Quadratic Formula)} \quad a = 1, b = (\phi(N) - 1 - N), c = N$$

Dependent Keys Part 1

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.

- **Question**: Is this secure?

- **Answer**: No, given $e_i d_i$ employee i can factor N (and then recover everyone else's secret key).
- See Theorem 8.50 in the textbook

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $\mathrm{e_i d_i}=1 \bmod \phi(N)$.

- Suppose that each employee is trusted (so it is ok if employee i factors N)

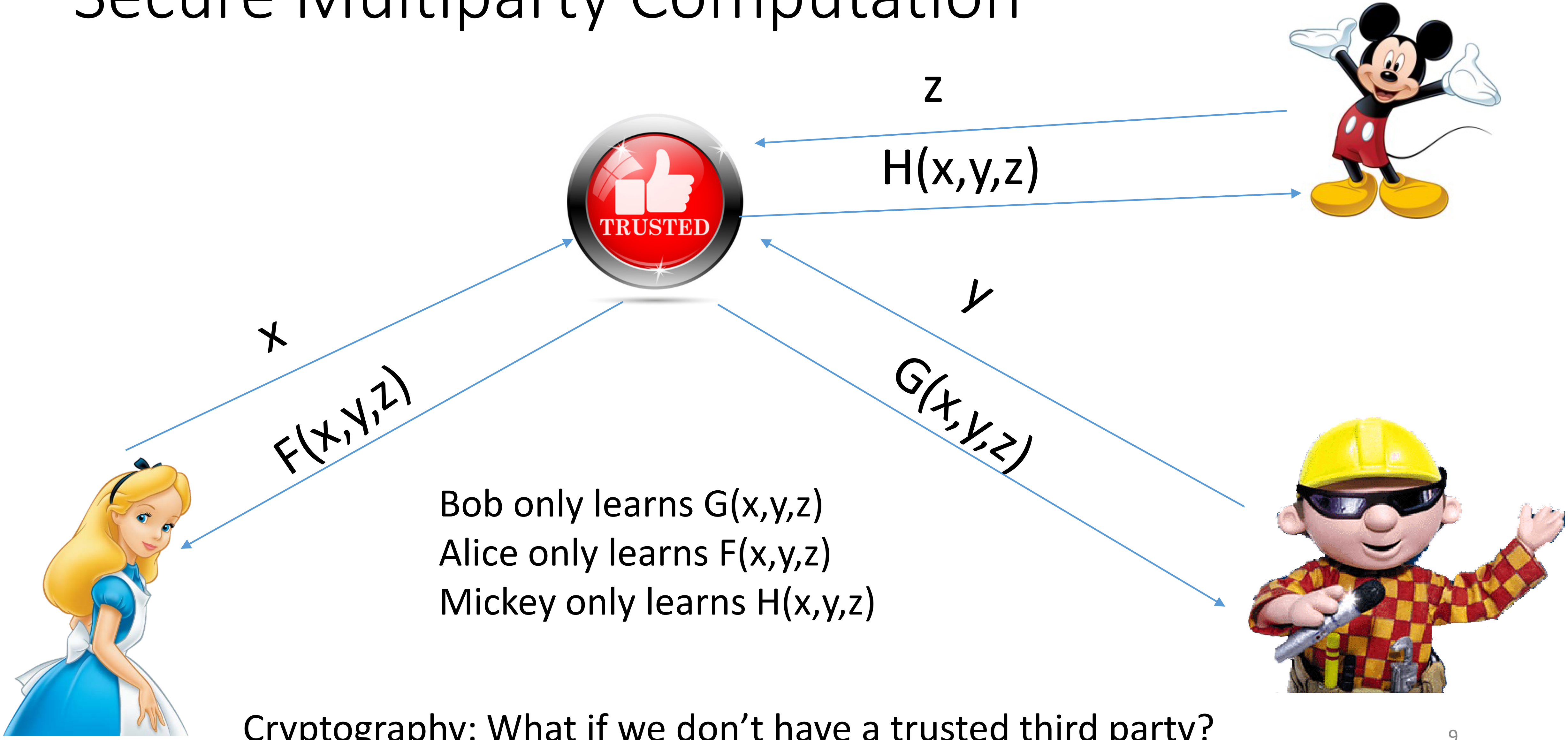
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1= [m^{e_1}\ mod\ N]$ and $c_2= [m^{e_2}\ mod\ N]$

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1 = [m^{e_1} \bmod N]$ and $c_2 = [m^{e_2} \bmod N]$
- If **gcd**$(e_1,e_2)=1$ (which is reasonably likely) then attacker can run extended GCD algorithm to find X,Y such that $Xe_1+Ye_2=1$.

$$[c_1{}^X c_2{}^Y \bmod N] = [m^{Xe_1} m^{Ye_2} \bmod N] = [m^{Xe_1+Ye_2} \bmod N] = m$$

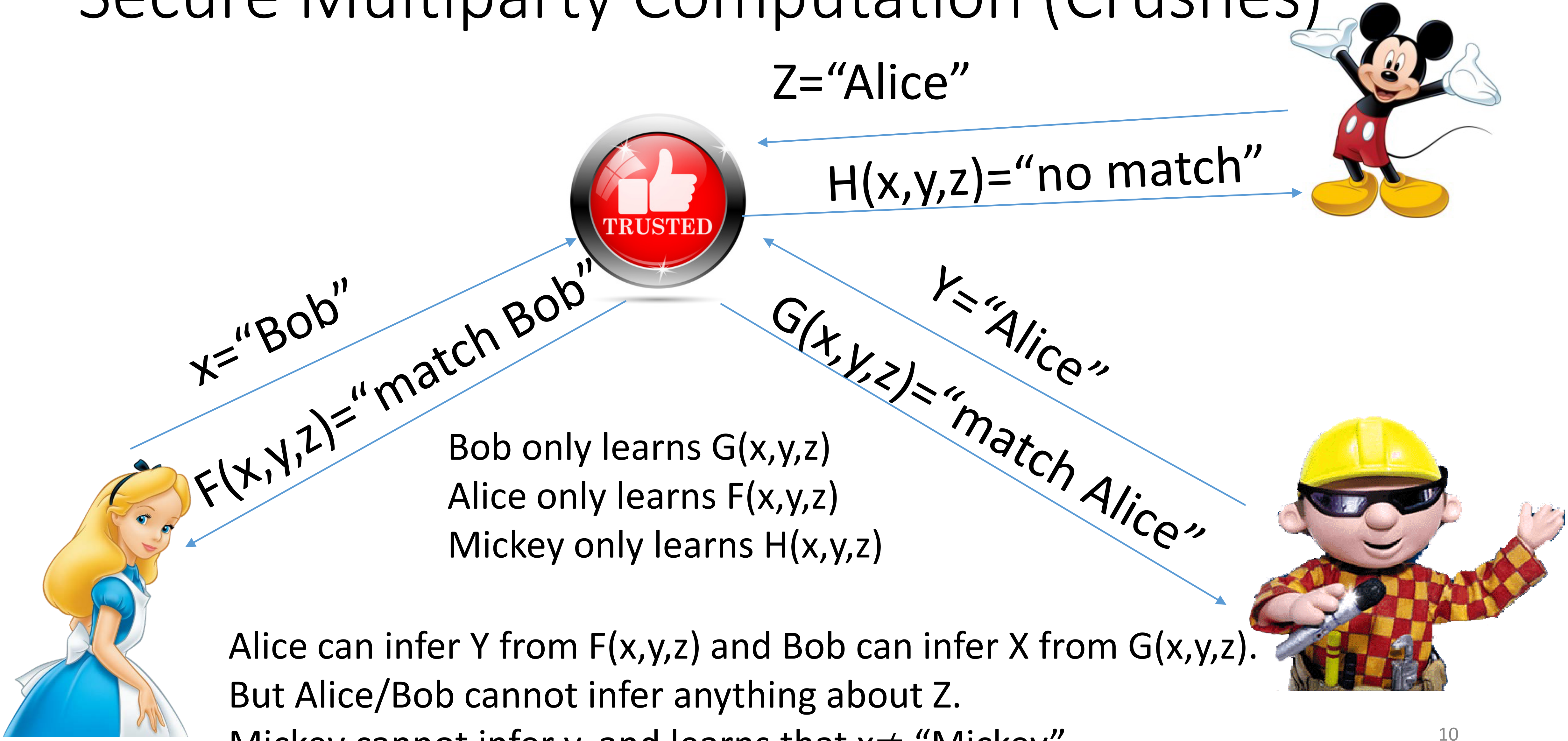
Secure Multiparty Computation



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)



Z="Alice"

H(x,y,z)="no match"

TRUSTED

X="Bob"

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

Y="Alice"

G(x,y,z)="match Alice"

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 G(x,y,z).
But Alice/Bob cannot infer anything about Z.
Mickey cannot infer y, and learns that x≠ "Mickey"

Secure Multiparty Computation (Crushes)

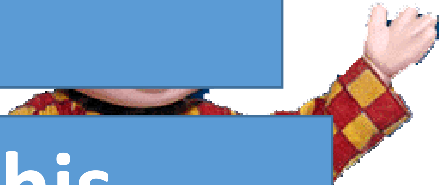
x="Bob"

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

Bob o

Alice c

Mickey only learns H(x,y,z)



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≠ "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
- Current Focus: Semi-Honest Protocols

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

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

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)

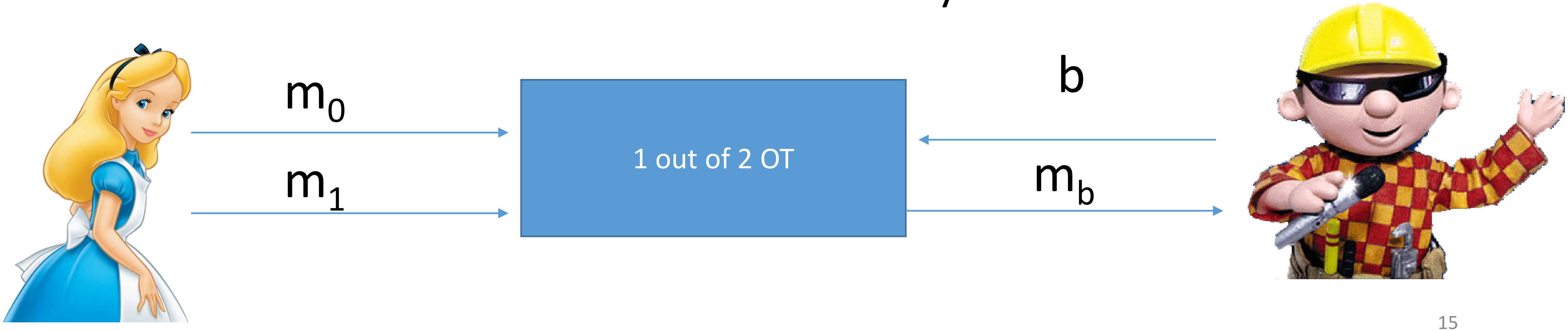
- Let $B_n = trans_B(n, x, y)$ (resp. $A_n = trans_A(n, x, y)$) be the protocol transcript from Bob's perspective (resp. Alice's perspective) when his input is y and Alice's input is x (assuming that Alice follows the protocol).
- **Security**: Assuming that Alice and Bob are both semi-honest (follow 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 $f_B(x, y)$)

Building Block: 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, and Bob learns nothing about other message
- Oblivious Transfer with a Trusted Third Party



m_0

m_1

1 out of 2 OT

b

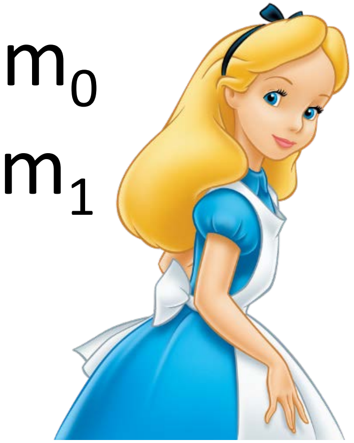
m_b

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



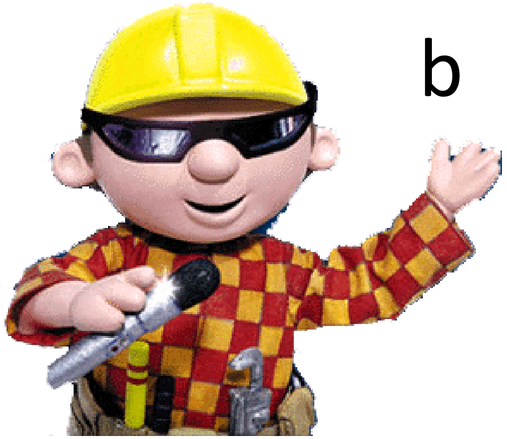
$c \leftarrow_R G_q$

$$\xrightarrow{\quad c \quad}$$

$$\xleftarrow{\quad z_0, z_1 \quad}$$

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

b



$k \leftarrow_R Z_q$

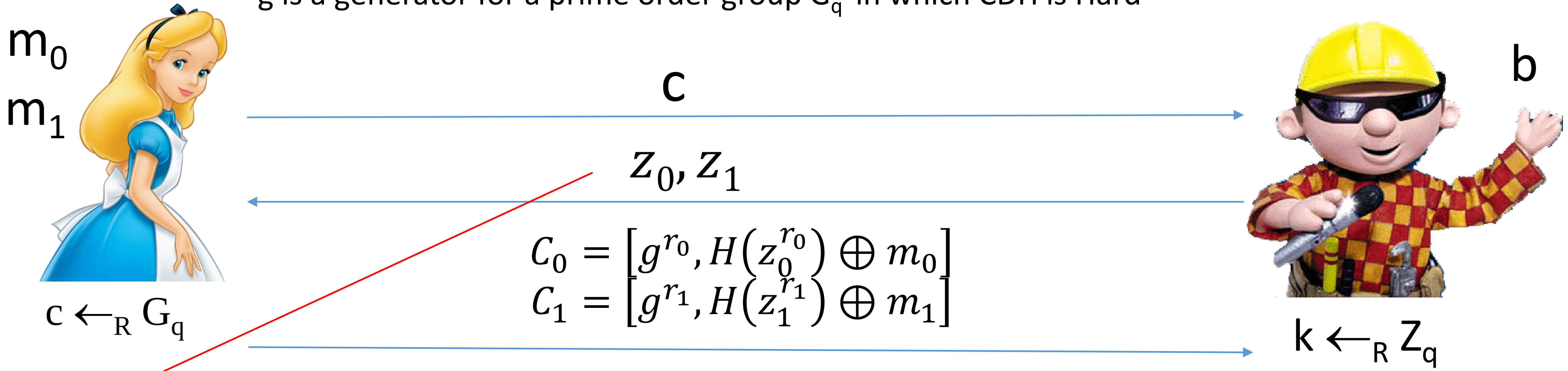
$$z_b = g^k, z_{1-b} = cg^{-k}$$

Bob can decrypt $C_\mathbf{b}$

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

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 is Hard



m_0

m_1

$c \leftarrow_R G_q$

b

$k \leftarrow_R Z_q$

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

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

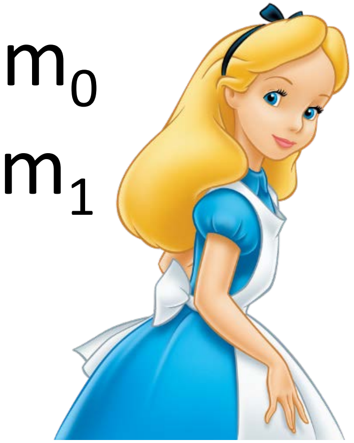
$$z_b = g^k, z_{1-b} = cg^{-k}$$
$$= c(z_b)^{-1}$$

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

- Oblivious Transfer withou
  - g is a generator for a prime

m_0

m_1



Alice does not learn b because

- $z_1 = c(z_0)^{-1}$ and
- $z_0 = c(z_1)^{-1}$ and
- z_1, z_0 are distributed uniformly at random subject to these condition.

This is an information theoretic guarantee!

$C_0 =$

$C_1 =$

$c \leftarrow_R G_q$

Alice must check that

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

Bob can decrypt C_b

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

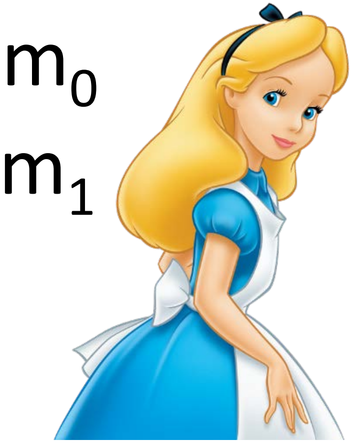
$$z_b = g^k, z_{1-b} = cg^{-k}$$
$$= c(z_b)^{-1}$$

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

- Oblivious Transfer withou
  - g is a generator for a prime

m_0

m_1



$c \leftarrow_R G_q$

$C_0 =$

$C_1 =$

Bob cannot decrypt $C_{1\text{-}b}$

Unless he queries random oracle at

- $c^{r_{1-b}} g^{-kr_{1-b}}$
- Given this value we can obtain $c^{r_{1-b}}$
- Thus, we can break CDH assumption

given random $c = g^m$ and $g^{r_{1-b}}$ it is hard to find $c^{r_{1-b}} = g^{mr_{1-b}}$

Alice must check that

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

Bob can decrypt C_b

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

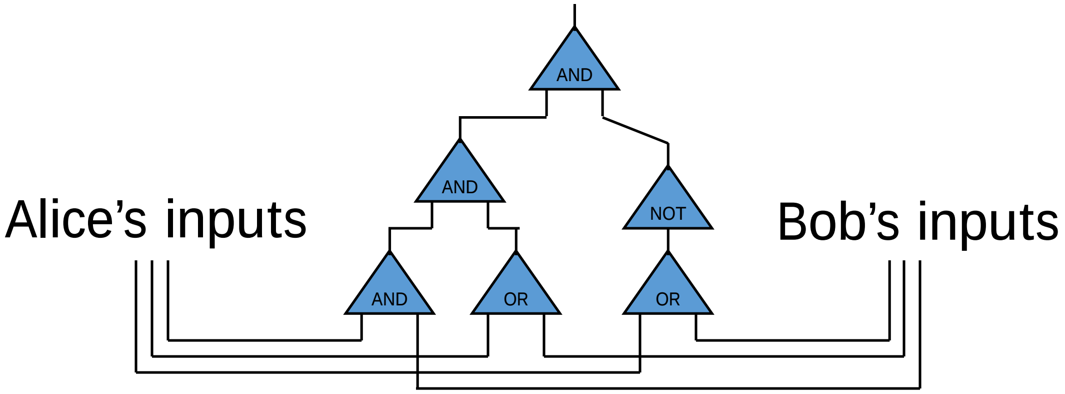
$$z_b = g^k, z_{1-b} = cg^{-k}$$
$$= c(z_b)^{-1}$$

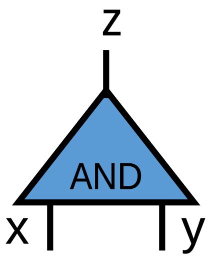
Yao’s Protocol

Vitaly Shmatikov

Yao’s Protocol

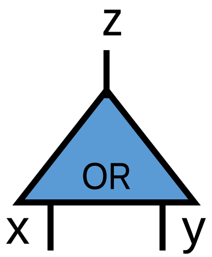
- Compute any function securely
  - … in the semi-honest model
- First, convert the function into a boolean circuit





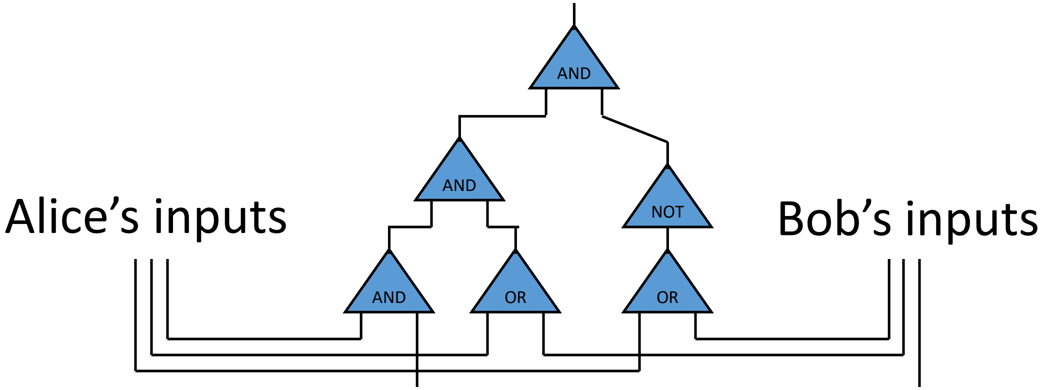
Truth table:

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



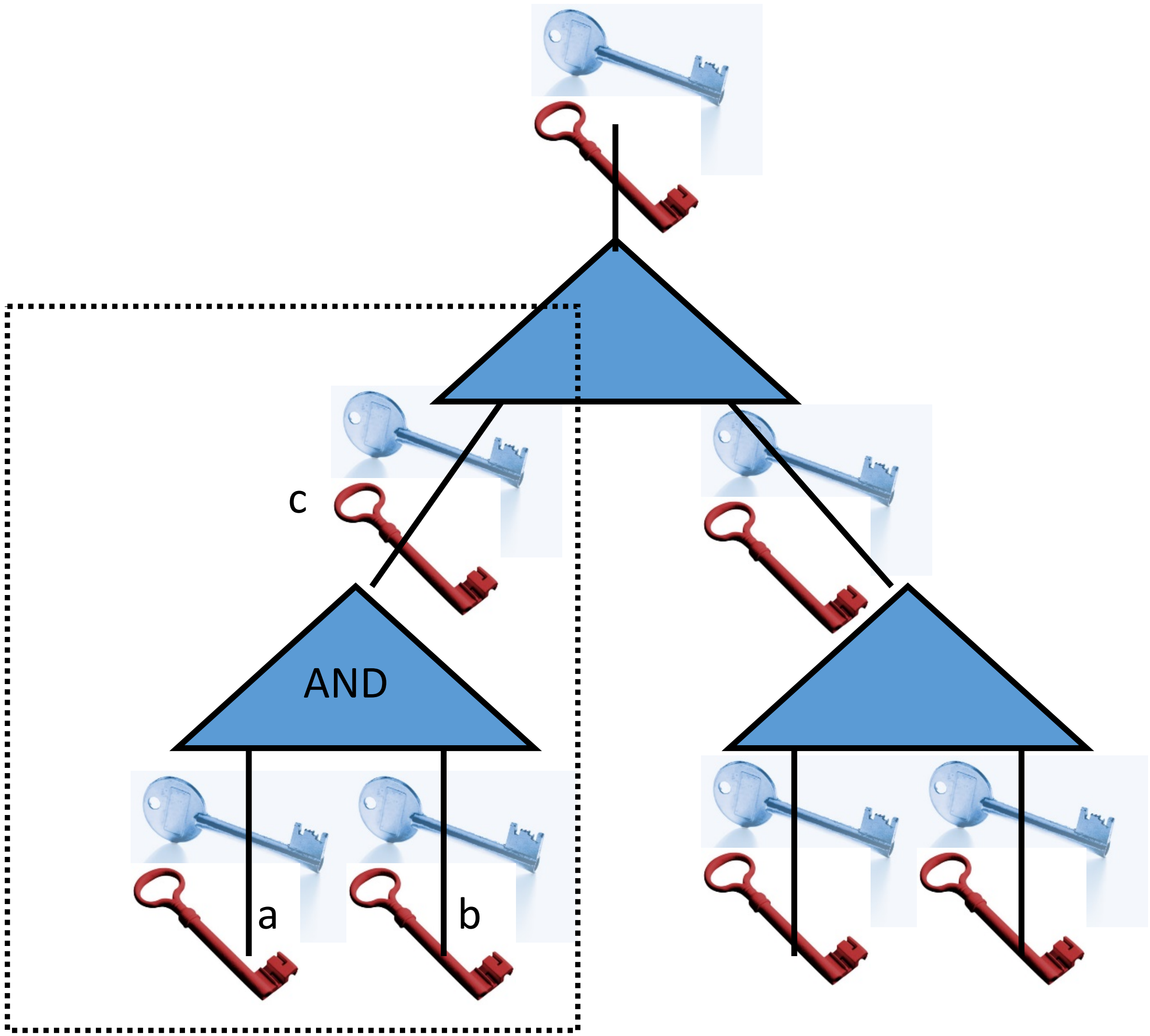
Overview:

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

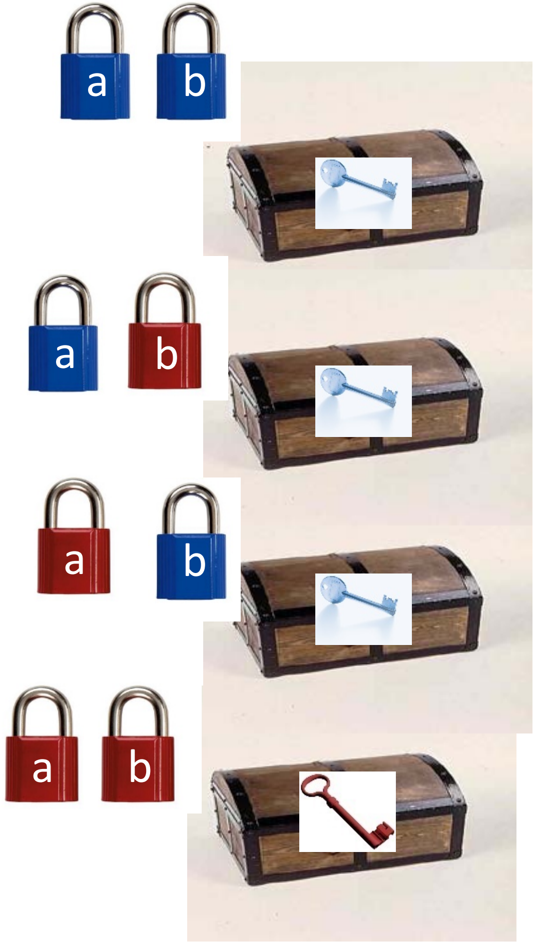
Crucial properties:

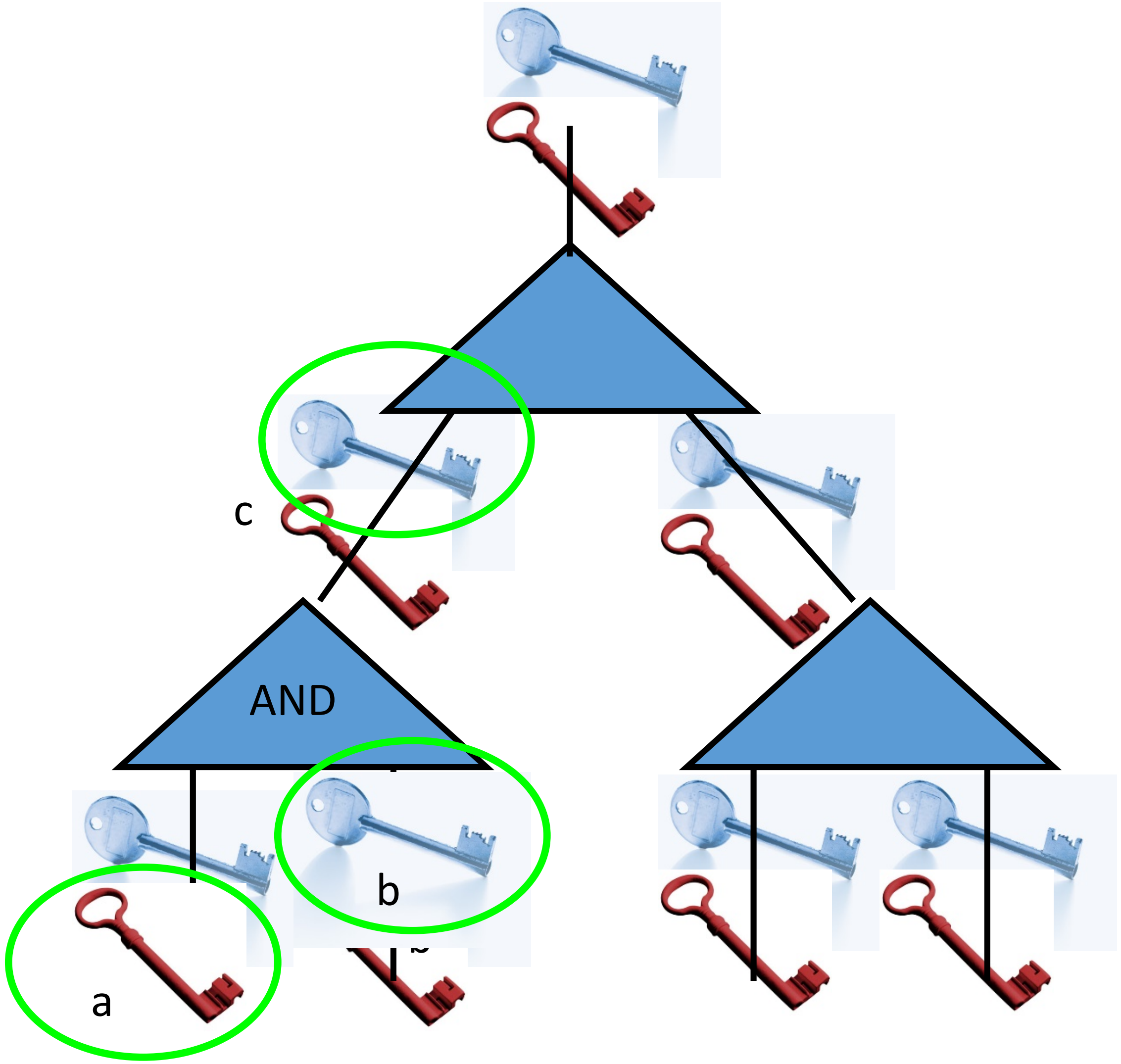
1. Bob never sees Alice's input x in unencrypted form.
2. Bob can obtain encryption of y without Alice learning y.
3. Neither party learns intermediate values.
4. Remains secure even if parties try to cheat.

Intuition



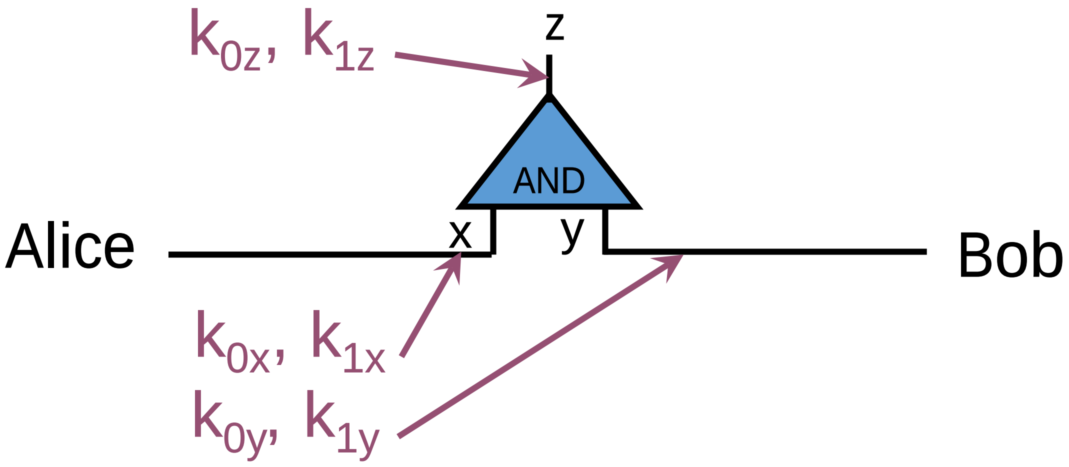
Intuition





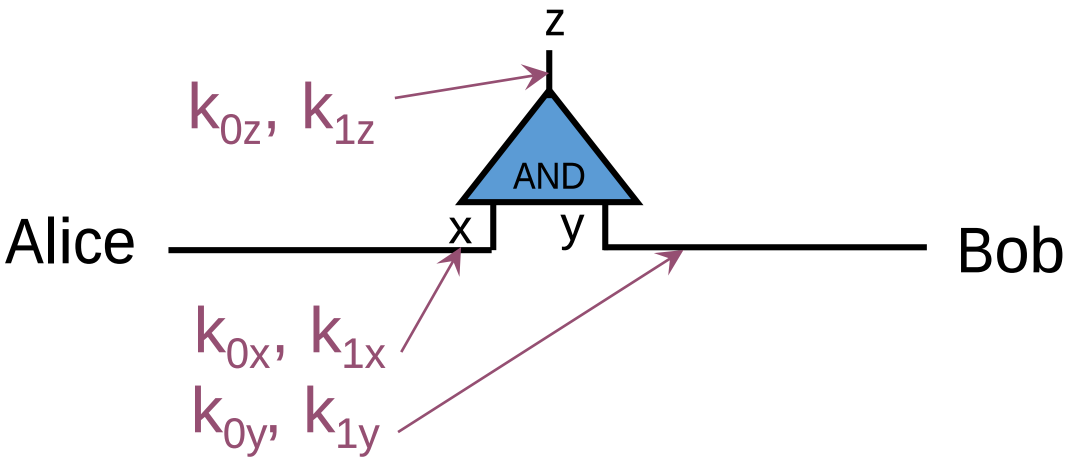
1: Pick Random Keys For Each Wire

- 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



z

k_{0z}, k_{1z}

AND

x

y

Alice

Bob

k_{0x}, k_{1x}

k_{0y}, k_{1y}

Original truth table:

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

Encrypted truth table:

$E_{k0x}(E_{k0y}(k_{0z}))$

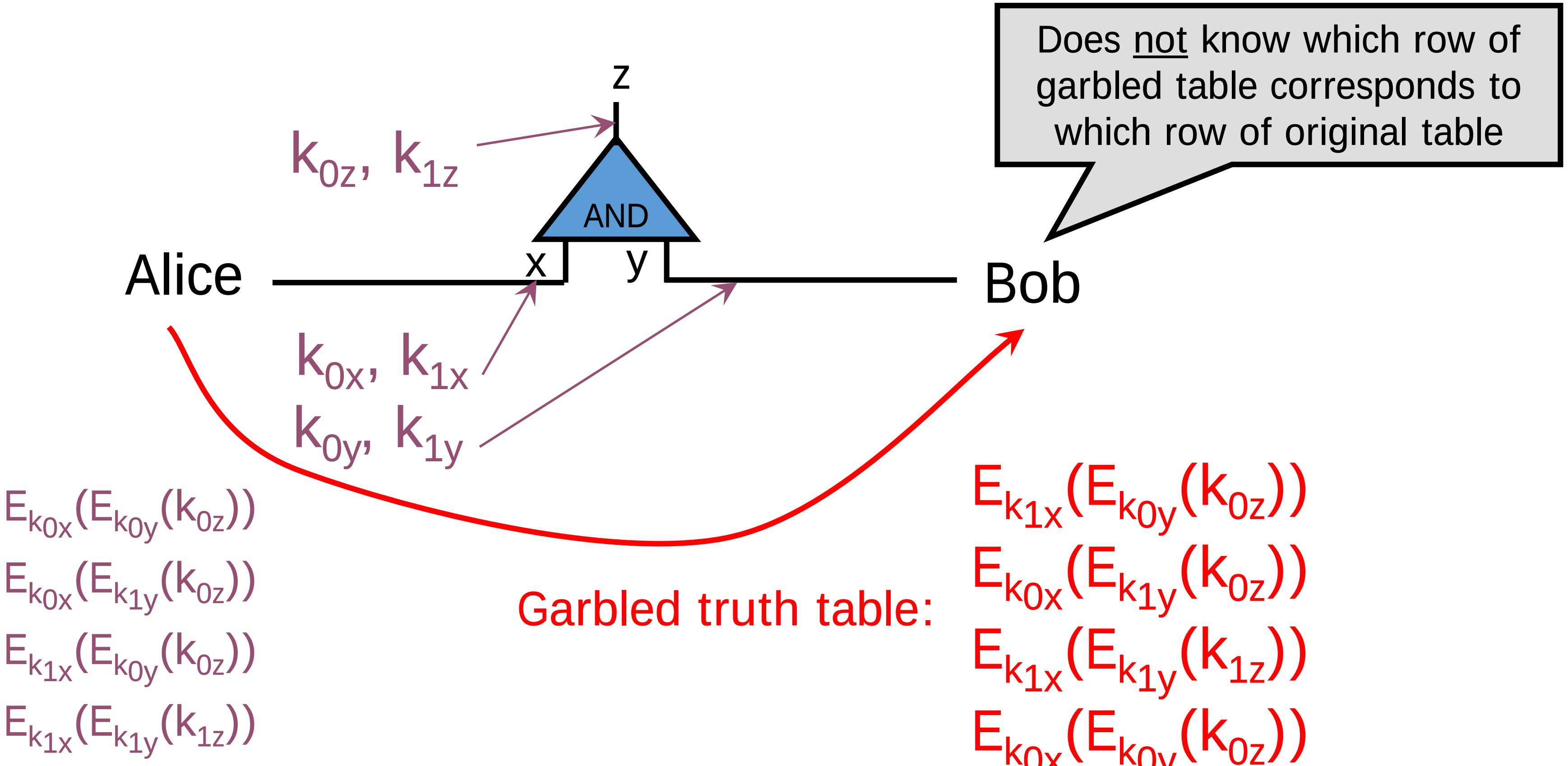
$E_{k0x}(E_{k1y}(k_{0z}))$

$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

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

$E_{k0x}(E_{k0y}(k_{0z}))$

$E_{k0x}(E_{k1y}(k_{0z}))$

$E_{k1x}(E_{k0y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

Garbled truth table:

$E_{k1x}(E_{k0y}(k_{0z}))$

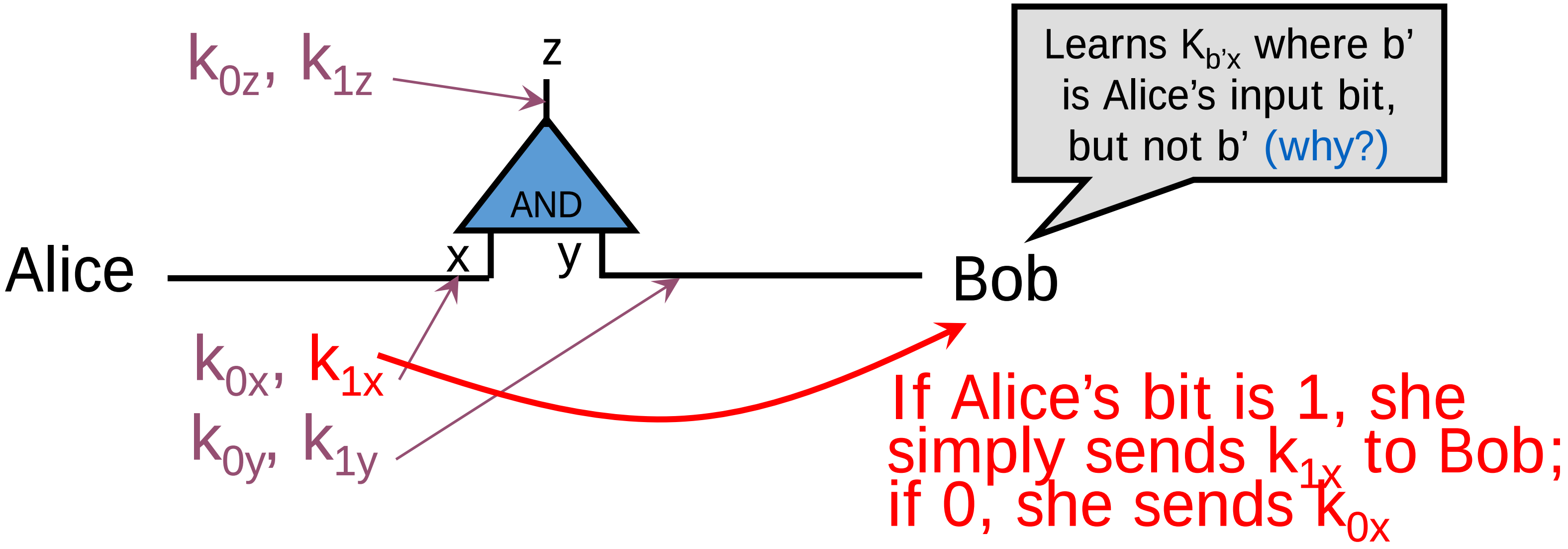
$E_{k0x}(E_{k1y}(k_{0z}))$

$E_{k1x}(E_{k1y}(k_{1z}))$

$E_{k0x}(E_{k0y}(k_{0z}))$

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



Learns $K_{b'x}$ where b' is Alice's input bit, but not b' (why?)

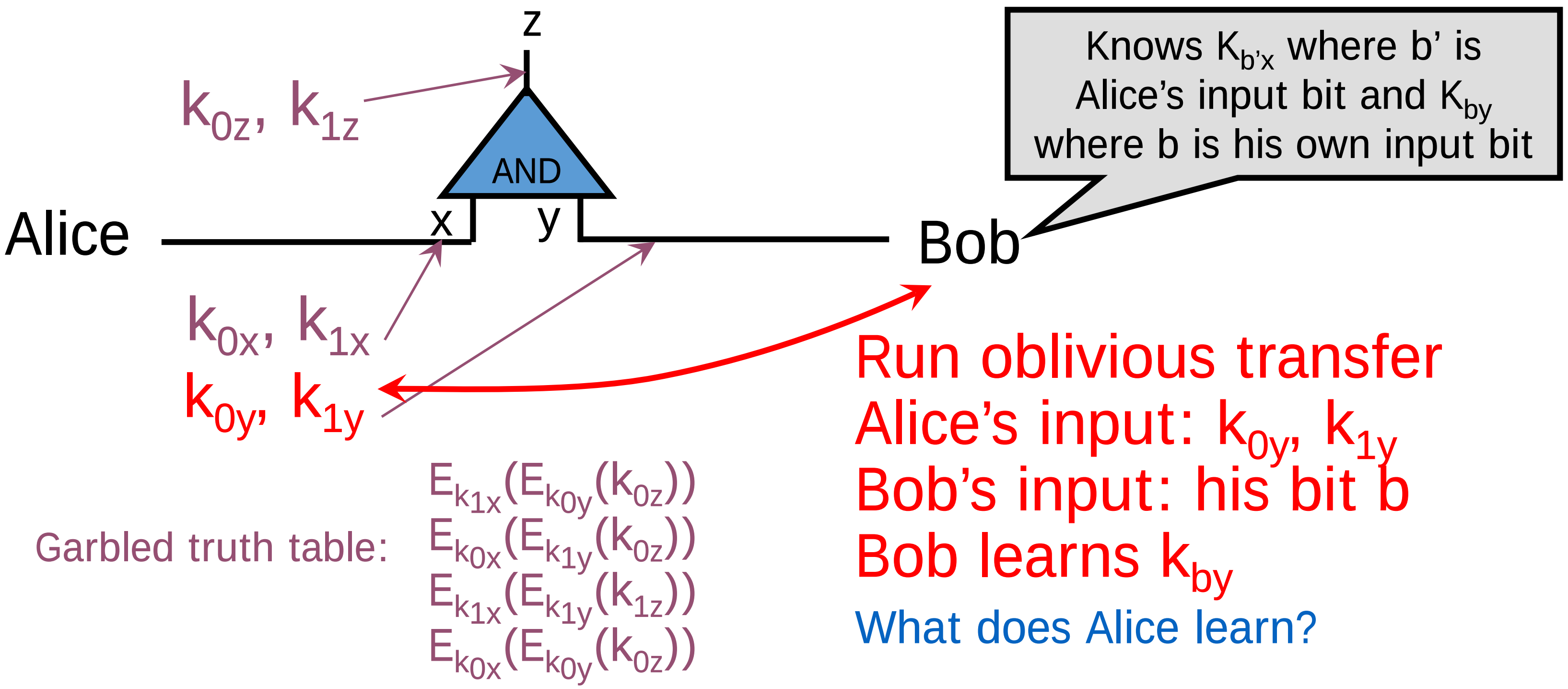
If Alice's bit is 1, she simply sends k_{1x} to Bob; if 0, she sends k_{0x}

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



z

k_{0z}, k_{1z}

AND

Alice

x

y

Bob

Knows $K_{b'x}$ where b’ is Alice’s input bit and K_{by} where b is his own input bit

k_{0x}, k_{1x}

k_{0y}, k_{1y}

Garbled truth table:

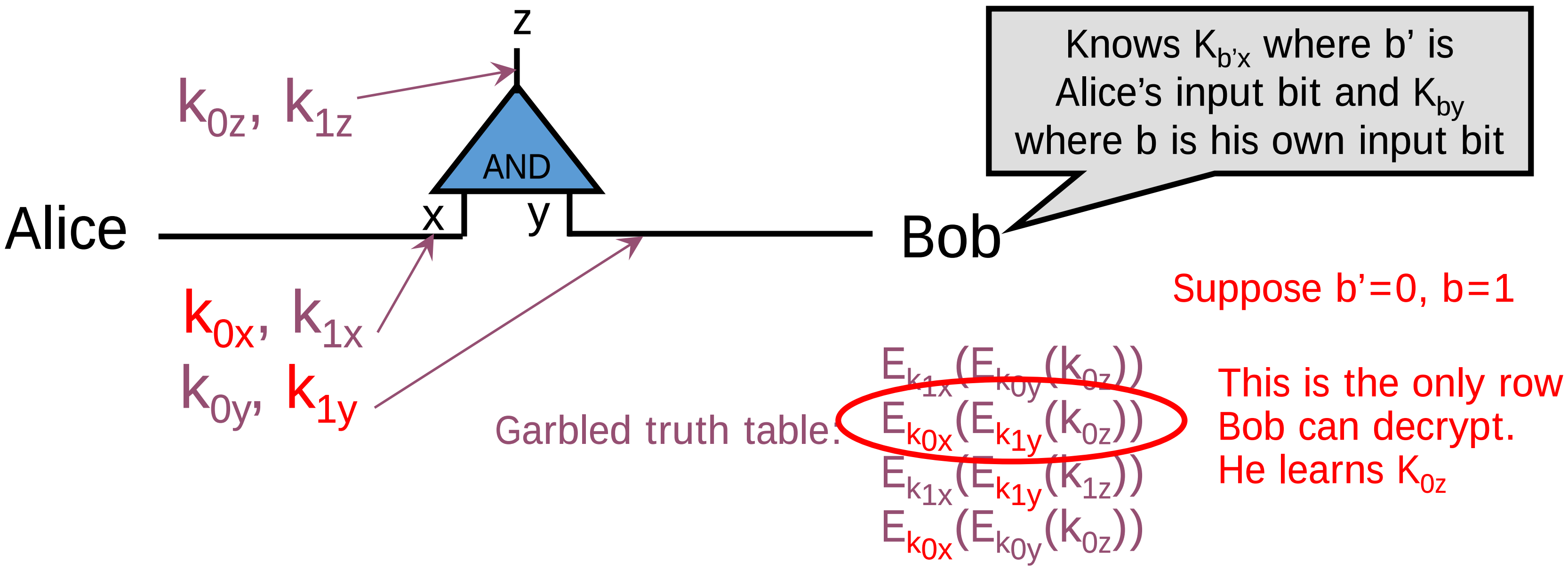
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

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?



Knows $K_{b'x}$ where b' is Alice's input bit and K_{by} where b is his own input bit

Suppose b'=0, b=1

Garbled truth table:

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

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

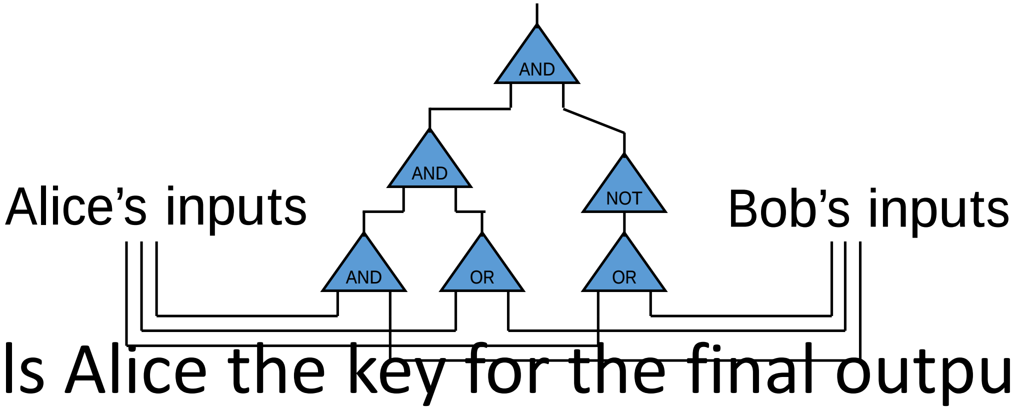
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$

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

This is the only row Bob can decrypt. He learns K_{0z}

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 not 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.
$$\{A_n\}_{n\in\mathbb{N}} \equiv_C \{S_A(n, x, f_A(x, y)\ )\}_{n\in\mathbb{N}}$$
$$\{B_n\}_{n\in\mathbb{N}} \equiv_C \left\{S_B\left(n, y, f_B(x, y)\right)\right\}_{n\in\mathbb{N}}$$

- **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.

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

Zero-Knowledge Proofs

Computational Indistinguishability

- Consider two distributions X_ℓ and Y_ℓ (e.g., over strings of length ℓ).
- Let D be a distinguisher that attempts to guess whether a string s came from distribution X_ℓ or Y_ℓ.

The advantage of a distinguisher D is

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

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

$$Adv_{D,n} \leq negl(n)$$

Computational Indistinguishability

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

- Consider two d ℓ).
- Let D be a disti came from distribution X_ℓ

The advantage of a distinguisher D is

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

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

$$Adv_{D,n} \leq negl(n)$$

P vs NP

- **P** problems that can be solved in polynomial time

- **NP** --- problems whose solutions can be **verified** in polynomial time
  - Examples: SHORT-PATH, COMPOSITE, 3SAT, CIRCUIT-SAT, 3COLOR,
  - DDH
    - **Input:** $A = g^{x_1}$, $\mathrm{B} = g^{x_2}$ and Z
    - **Goal:** Decide if $\mathrm{Z} = g^{x_1 x_2}$ or $\mathrm{Z} \neq g^{x_1 x_2}$.
  - **NP-Complete** --- hardest problems in NP (e.g., all problems can be reduced to 3SAT)
- **Witness**
  - A short (polynomial size) string which allows a verify to check for membership
  - DDH Witness: x_1,x_2.

Zero-Knowledge Proof

Two parties: Prover P (PPT) and Verifier V (PPT)

(P is given witness for claim e.g.,)

- **Completeness:** If claim is true honest prover can always convince honest verifier to accept.
- **Soundness:** If claim is false then Verifier should reject with probability at least ½. (Even if the prover tries to cheat)
- **Zero-Knowledge:** Verifier doesn’t learn anything about prover’s input from the protocol (other than that the claim is true).
- Formalizing this last statement is tricky
- **Zero-Knowledge:** should hold even if the attacker is dishonest!

Zero-Knowledge Proof

Trans(1^n,V',P,x,w,r_p,r_v) transcript produced when V' and P interact

- V' is given input X (the problem instance e.g., $X = g^x$)
- P is given input X and w (a witness for the claim e.g., w=x)
- V' and P use randomness r_p and r_v respectively
- Security parameter is n e.g., for encryption schemes, commitment schemes etc…

$\boldsymbol{X_n} =$ **Trans(1^n,V',P,x,w)** is a distribution over transcripts (over the randomness r_p,r_v)

(Blackbox Zero-Knowledge): There is a PPT simulator S such that for every V' (possibly cheating) S, with oracle access to V', can simulate X_n without a witness w. Formally,

$$\{X_n\}_{n\in\mathbb{N}} \equiv_C \left\{S^{V'(.)}(x, 1^n)\right\}_{n\in\mathbb{N}}$$

Zero-Knowledge Proof

Trans(1^n,V',P,x,w,r_p,r_v) transcript produced when V' and P interact

- V' is given input x (the problem instance e.g., $A = g^{x_1}$, $\mathrm{B} = g^{x_2}$ and z_b)
- P r the claim e.g.,
- V' espectively
- Se yption schemes,

X_n n over transcript

Simulator S is not given witness w

Oracle V'(x,trans) will output the next message V' would output given current transcript trans

(Blackbox Zero-Knowledge): There is a PPT simulator S such that for every V' (possibly cheating) S, with oracle access to V', can simulate X_n without a witness w. Formally,

$$\{X_n\}_{n\in\mathbb{N}} \equiv_C \left\{S^{V'(.)}(x,1^n)\right\}_{n\in\mathbb{N}}$$

Zero-Knowledge Proof for Discrete Log Solution

Bob (verifier);

A

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c \in \{0,1\}}$$

$$\textbf{Response } \boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ y + x & if\ c = 1 \end{cases}$$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ C = g^r\ \ and\ AB = C} \\ 0 & otherwise \end{cases}$$

Alice (prover);

x s.t.

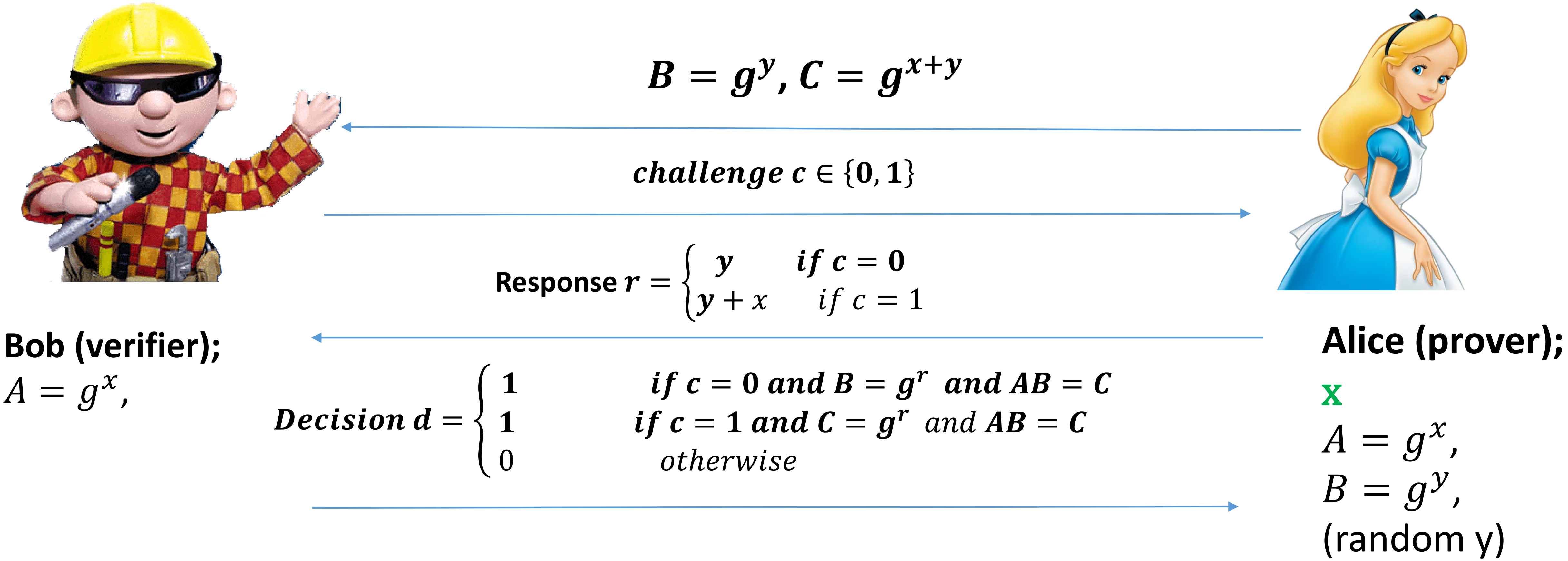
$A = g^x$,

$B = g^y$,

(random y)

Claim: There is some integer x such that $A = g^x$

Zero-Knowledge Proof for Discrete Log Solution



$\boldsymbol{B = g^y, C = g^{x+y}}$

$\boldsymbol{challenge\ c \in \{0,1\}}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y} + x & if\ c = 1 \end{cases}$

$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ C = g^r\ \ and\ AB = C} \\ 0 & otherwise \end{cases}$

Bob (verifier);

$A = g^x,$

Alice (prover);

x

$A = g^x,$

$B = g^y,$

(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

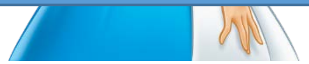
$B = g^y, C = g^{x+y}$

$challenge\ c \in \{0,1\}$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$Decision\ d = \begin{cases} 1 & if\ c = 0\ and\ B = g^r\ \ and\ AB = C \\ 1 & if\ c = 1\ and\ C = g^r\ \ and\ AB = C \\ 0 & otherwise \end{cases}$

Case 1: Challenge (c=0)



Bob (verifier);
$A = g^x,$

Alice (prover);
x
$A = g^x,$
$B = g^y,$
(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

$\boldsymbol{B = g^{y}, \boxed{C = g^{x+y}}}$

Case 2: Challenge (c=1)

$\boldsymbol{challenge\ c \in \{0,1\}}$

Response $r = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boxed{\boldsymbol{y} + x} & if\ c = 1 \end{cases}$

Bob (verifier);
$A = g^{x},$

$$\boldsymbol{Decision\ d} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ c = 0\ and\ B = g^{r}\ \ and\ AB = C} \\ \boldsymbol{1} & \boldsymbol{if\ c = 1\ and}\ \boxed{\boldsymbol{C = g^{r}}\ \ and\ \boldsymbol{AB = C}} \\ 0 & otherwise \end{cases}$$

Alice (prover);
x
$A = g^{x},$
$B = g^{y},$
(random y)

Correctness: If Alice and Bob are honest then Bob will always accept

Zero-Knowledge Proof for Discrete Log Solution

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y} + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ C = g^r}\ \ and\ \boldsymbol{AB = C} \\ 0 & otherwise \end{cases}$$

Bob (verifier);

$A = g^x,$

Alice (prover);

x

$A = g^x,$

$B = g^y,$

(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof fo

Assume that AB=C, now If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

$$B = g^y, C = g^{x+y}$$

$$challenge\ c \in \{0, 1\}$$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$$Decision\ d = \begin{cases} 1 & if\ c = 0\ and\ B = g^r\ and\ AB = C \\ 1 & if\ c = 1\ and\ C = g^r\ and\ AB = C \\ 0 & otherwise \end{cases}$$

Bob (verifier);
$A = g^x$,

Alice (prover);
x
$A = g^x$,
$B = g^y$,
(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Case 1: for all r $B \neq g^r$

$$\rightarrow Pr[reject] \geq Pr[c = 0] = \frac{1}{2}$$

fo

Assume that AB=C, now If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c \in \{0,1\}}$$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$$Decision\ d = \begin{cases} 1 & if\ \boxed{c = 0\ and\ B = g^r}\ and\ AB = C \\ 1 & if\ c = 1\ and\ C = g^r\ and\ AB = C \\ 0 & otherwise \end{cases}$$

Bob (verifier);

$A = g^x$,

Alice (prover);

x

$A = g^x$,

$B = g^y$,

(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Case 2: for all r $C \neq g^r$

$$\rightarrow Pr[reject] \geq Pr[c = 1] = \frac{1}{2}$$

fo

Assume that AB=C, now If $B = g^y$ and $C = g^{x+y}$ for some x,y then $A = g^x$

$\boldsymbol{B = g^y, C = g^{x+y}}$

$\boldsymbol{challenge\ c \in \{0, 1\}}$

Response r $= \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ y + x & if\ c = 1 \end{cases}$

Bob (verifier);

$A = g^x,$

$$\boldsymbol{Decision\ d} = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ c = 0\ and\ B = g^r\ \ and\ AB = C} \\ \boldsymbol{1} & \boldsymbol{if\ \boxed{c = 1\ and\ C = g^r}\ and\ AB = C} \\ 0 & otherwise \end{cases}$$

Alice (prover);

x

$A = g^x,$

$B = g^y,$

(random y)

Soundness: If $A \neq g^x$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof for Discrete Log Solution

Dishonest (verifier);
$A = g^x$,

$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c = V'(A, (B, C)) \in \{0, 1\}}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y} + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d = V'(A, (B, C), c, r)}$$

Alice (honest);

x

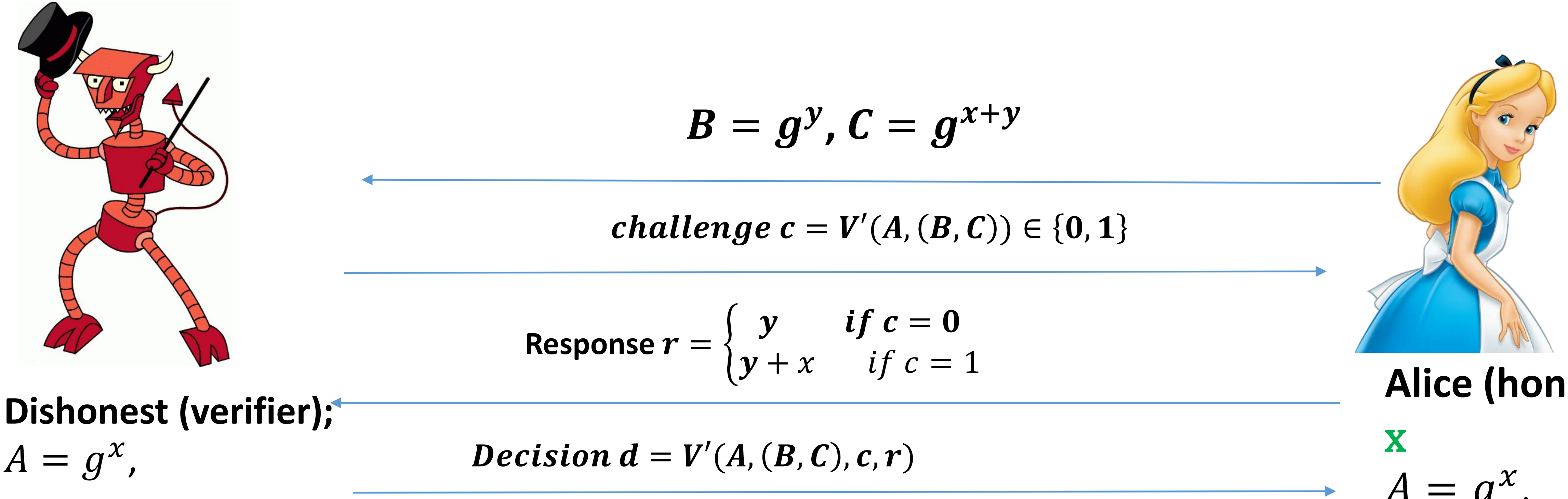
$A = g^x$,

$B = g^y$,

(random y)

Transcript: $\boldsymbol{View_{V'}} = (A, (B, C), c, r, d)$

Zero-Knowledge Proof for Discrete Log Solution



$$\boldsymbol{B = g^y, C = g^{x+y}}$$

$$\boldsymbol{challenge\ c = V'(A, (B, C)) \in \{0, 1\}}$$

Response $r = \begin{cases} y & if\ c = 0 \\ y + x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d = V'(A, (B, C), c, r)}$$

Dishonest (verifier);
$A = g^x,$

Alice (honest);
x
$A = g^x,$
$B = g^y,$
(random y)

Zero-Knowledge: For all PPT V' exists PPT Sim s.t $\boldsymbol{View_{V'}} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

Zero-Knowledge Proof for Discrete Log Solution

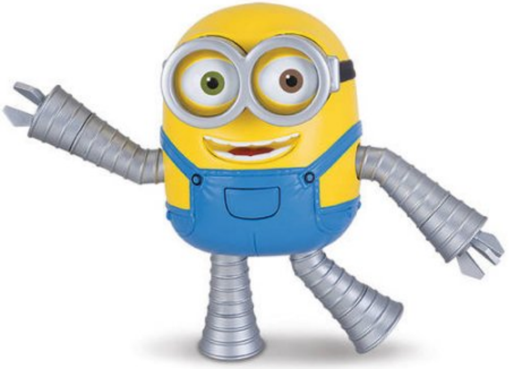
Dishonest (verifier);
$A = g^x,$

$$\begin{cases} B = g^y, C = AB & \text{if b=0} \\ B = \frac{C}{A}, C = g^y & otherwise \end{cases}$$

$$challenge\ c = V'(A,(B,C)) \in \{0,1\}$$

Response $r = \begin{cases} y & \text{if c=b} \\ \perp & otherwise \end{cases}$

$$Decision\ d = V'(A,(B,C),c,r)$$



Simulator
Cheat bit b,
$A = g^x,$
$B = g^y,$
(random y)

Zero-Knowledge: For all PPT V' exists PPT Sim s.t $\boldsymbol{View}_{V'} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

Zero-Knowledge Proof for Discrete Log Solution

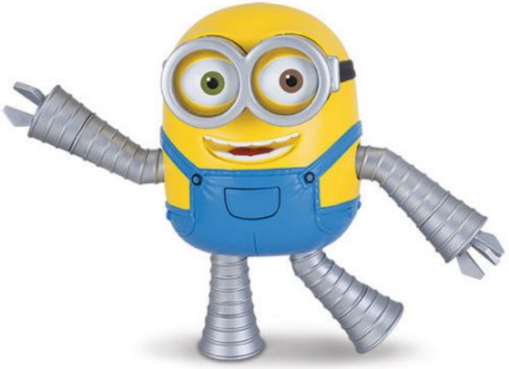
$$\begin{cases} \boldsymbol{B = g^y, C = AB} & \text{if b=0} \\ \boldsymbol{B = \frac{C}{A}, C = g^y\ otherwise} & \end{cases}$$

$\boldsymbol{challenge\ c = V'(A,(B,C)) \in \{0,1\}}$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c{=}b} \\ \perp & otherwise \end{cases}$

$\boldsymbol{Decision\ d = V'(A,(B,C),c,r)}$

Dishonest (verifier);
$A = g^x,$



Simulator
Cheat bit b,
$A = g^x,$
$B = g^y,$
(random y)

Zero-Knowledge: Simulator can produce identical transcripts (Repeat until $r \neq \perp$)

Zero-Knowledge Proof for Discrete Log Solution

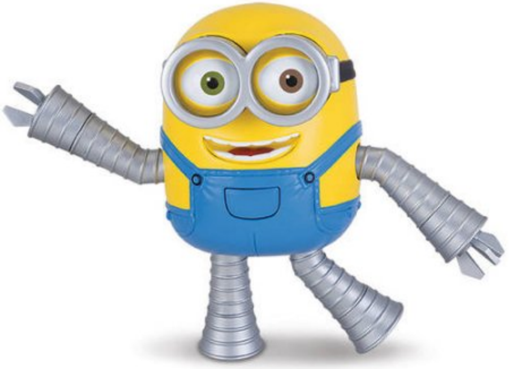
$$\begin{cases} B = g^y, C = AB & \text{if b=0} \\ B = \dfrac{C}{A}, C = g^y & \textit{otherwise} \end{cases}$$

$\boldsymbol{challenge\ c = V'(A, (B, C)) \in \{0, 1\}}$

Response $\boldsymbol{r} = \begin{cases} y & \textit{if c=b} \\ \perp & \textit{otherwise} \end{cases}$

$\boldsymbol{Decision\ d = V'(A, (B, C), c, r)}$

Dishonest (verifier);
$A = g^x$,



Simulator
Cheat bit b,
$A = g^x$,
$B = g^y$,
(random y)

Zero-Knowledge: If $A = g^x$ for some x then $\boldsymbol{View_{V'}} \equiv_C \mathrm{Sim}^{V'(.)}(A)$

Zero-Knowledge Proof for Square Root mod N

Bob (verifier);

z

$$M = zy^2 \ mod \ N$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0},\mathbf{1}\}$$

$$\textbf{Response } \boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y}x & if\ c = 1 \end{cases}$$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2}\ mod\ N \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2$ mod N

(random y)

Completeness: If Alice knows x such $z = x^2$ mod N then Bob will always accept

Zero-Knowledge Proof for Square Root mod N

Bob (verifier);

z

$$M = zy^2\ mod\ N$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ \boldsymbol{y}x & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2}\ mod\ N \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2$ mod N

(random y)

Soundness: If $z \neq x^2$ for some x then (honest) Bob will reject w.p. ½ (even if Alice cheats)

Zero-Knowledge Proof for Square Root mod N

Bob (verifier);

z

$$M = zy^2 \; mod \; N$$

$$\boldsymbol{challenge\ c} \in \{\mathbf{0}, \mathbf{1}\}$$

Response $\boldsymbol{r} = \begin{cases} \boldsymbol{y} & \boldsymbol{if\ c = 0} \\ yx & if\ c = 1 \end{cases}$

$$\boldsymbol{Decision\ d} = \begin{cases} \mathbf{1} & \boldsymbol{if\ c = 0\ and\ M = zr^2} \text{ mod N} \\ \mathbf{1} & \boldsymbol{if\ c = 1\ and\ M = r^2} \text{ mod N} \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2$ mod N

(random y)

Zero-Knowledge: How does the simulator work?

Zero-Knowledge Proof vs. Digital Signature

- Digital Signatures are transferrable
  - E.g., Alice signs a message m with her secret key and sends the signature σ to Bob. Bob can then send (m, σ) to Jane who is convinced that Alice signed the message m.

- Are Zero-Knowledge Proofs transferable?
  - Suppose Alice (prover) interacts with Bob (verifier) to prove a statement (e.g., z has a square root modulo N) in Zero-Knowledge.
  - Let $\boldsymbol{View_V}$ be Bob's view of the protocol.
  - Suppose Bob sends $\boldsymbol{View_V}$ to Jane.
  - Should Jane be convinced of the statement (e.g., z has a square root modulo N)>

Non-Interactive Zero-Knowledge Proof (NIZK)

Bob (verifier);

z

M$_1$,…M$_k$ where $M_i = y_i^2 z \; mod \; N$

$\boldsymbol{challenges \; c = (c_1, \ldots, ck) = H}$(M$_1$,…Mk)

Responses r$_1$,…,r$_k$ where $\boldsymbol{r_i} = \begin{cases} \boldsymbol{y_i} & \boldsymbol{if \; ci = 0} \\ \boldsymbol{y_i} x & if \; ci = 1 \end{cases}$

$$\boldsymbol{Decision \; d} = \prod_i \boldsymbol{d_i} \;\; \boldsymbol{where \; di} = \begin{cases} \mathbf{1} & \boldsymbol{if \; ci = 0 \; and \; M_i = r_i^2 z} \text{ mod N} \\ \mathbf{1} & \boldsymbol{if \; ci = 1 \; and \; M_i = r_i^2} \text{ mod N} \\ 0 & otherwise \end{cases}$$

Alice (prover);

X

$z = x^2$ mod N

(random $y_1, \ldots, yk$)

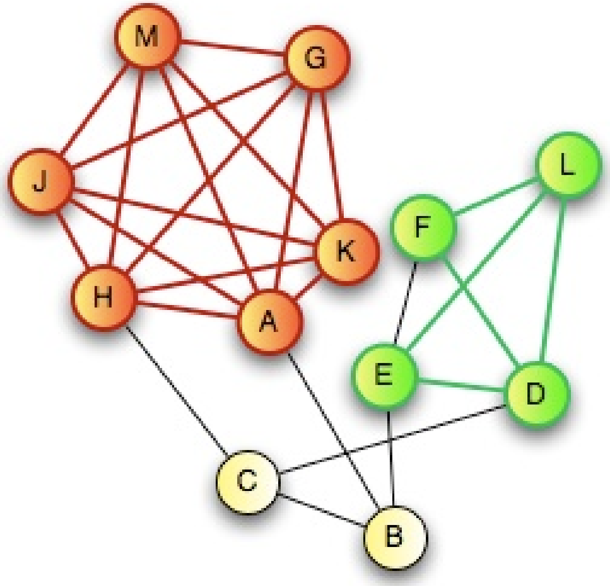
Simulator Power: Can program the random oracle

NIZK Security (Random Oracle Model)

- Simulator is given statement to proof (e.g., z has a square root modulo N)
- Simulator must output a proof π'_z and a random oracle H’

- Distinguisher D
  - World 1 (Simulated): Given z, π'_z and oracle access to H’
  - World 2 (Honest): Given z, π_z (honest proof) and oracle access to H
  - Advantage: $\mathrm{ADV_D} = |Pr[D^H(\mathrm{z}, \pi_z) = 1] - Pr[D^{H'}(\mathrm{z}, \pi'_z) = 1]|$
- **Zero-Knowledge:** Any PPT distinguisher D should have negligible advantage.
- NIZK proof π_z is transferrable (contrast with interactive ZK proof)

Zero-Knowledge Proof for all NP

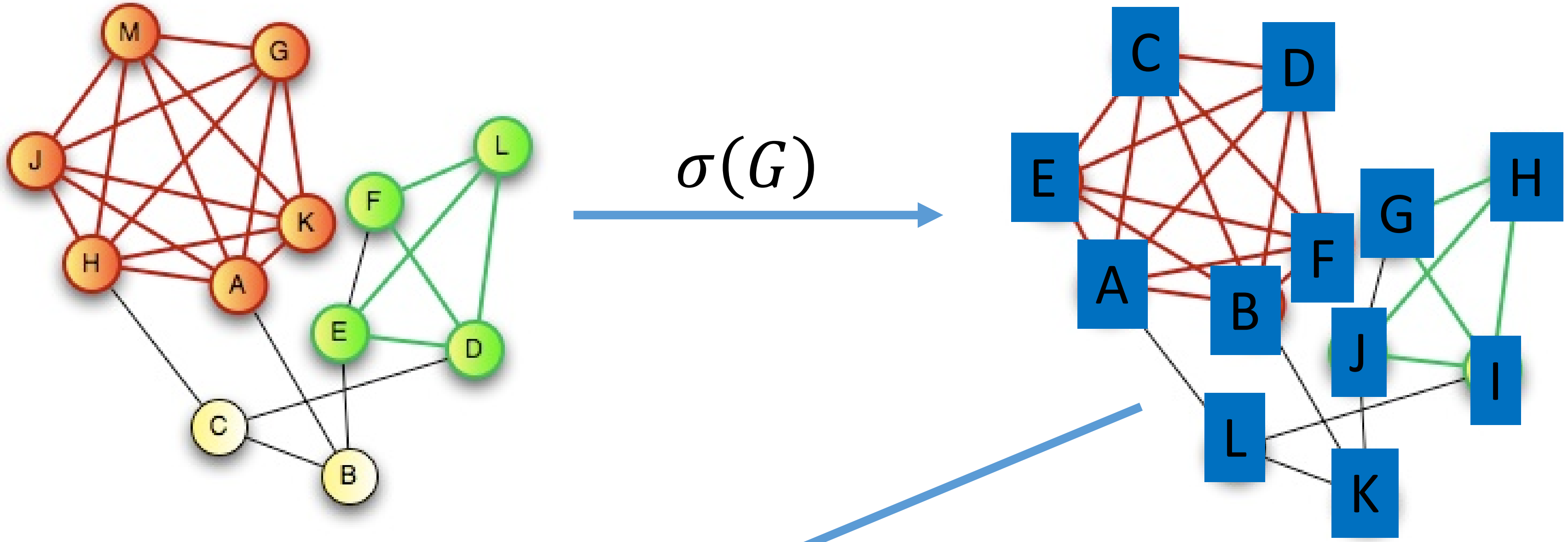


- CLIQUE
  - Input: Graph G=(V,E) and integer k>0
  - Question: Does G have a clique of size k?

- CLIQUE is NP-Complete
  - Any problem in NP reduces to CLIQUE
  - A zero-knowledge proof for CLIQUE yields proof for all of NP via reduction

- Prover:
  - Knows k vertices $v_1,\ldots,v_k$ in G=(V,E) that form a clique

Zero-Knowledge Proof for all NP



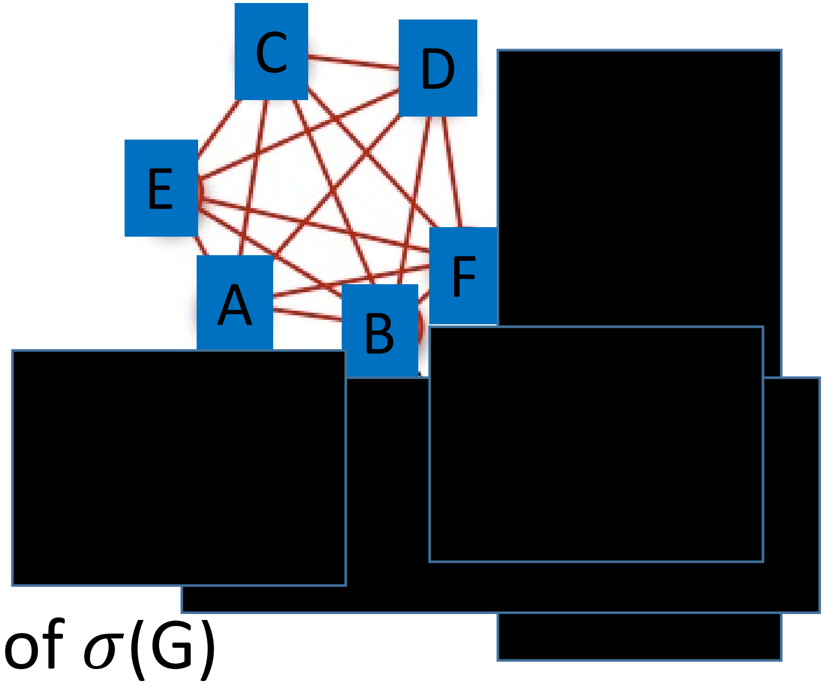
Adjacency matrix $A_{\sigma(G)}$

$$\begin{array}{c|ccc} & A & & L \\ \hline A & 0 & \cdots & 1 \\ & \vdots & \ddots & \vdots \\ L & 1 & \cdots & 0 \end{array}$$

Commitment to $A_{\sigma(G)}$

$$\begin{array}{c|ccc} & A & & L \\ \hline A & Com(0, r_{A,A}) & \cdots & Com(1, r_{A,L}) \\ & \vdots & \ddots & \vdots \\ L & Com(1, r_{L,A}) & \cdots & Com(0, r_{L,L}) \end{array}$$

Zero-Knowledge Proof for all NP



- Prover:
  - Knows k vertices v$_1$,…,v$_k$ in G=(V,E) that for a clique

1. Prover commits to a permutation σ over V
2. Prover commits to the adjacency matrix $A_{\sigma(G)}$ of σ(G)
3. Verifier sends challenge c (either 1 or 0)
4. If c=0 then prover reveals σ and adjacency matrix $A_{\sigma(G)}$
   1. Verifier confirms that adjacency matrix is correct for σ(G)
5. If c=1 then prover reveals the submatrix formed by first rows/columns of $A_{\sigma(G)}$ corresponding to $\sigma(v_1), \ldots, \sigma(v_k)$
   1. Verifier confirms that the submatrix forms a clique.

Zero-Knowledge Proof for all NP

- **Completeness**: Honest prover can always make honest verifier accept
- **Soundness**: If prover commits to adjacency matrix $A_{\sigma(G)}$ of σ(G) and can reveal a clique in submatrix of $A_{\sigma(G)}$ then G itself contains a k-clique. Proof invokes binding property of commitment scheme.
- **Zero-Knowledge:** Simulator cheats and either commits to wrong adjacency matrix or cannot reveal clique. Repeat until we produce a successful transcript. Indistinguishability of transcripts follows from hiding property of commitment scheme.

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
  - Prove: My behavior in the protocol is consistent with honest party