CRYPTOGRAPHY

(lecture 11)

Literature:

"Lecture Notes on Introduction to Cryptography" by V. Goyal (ch 12.5,**12.7.2, 12.7.3**,12.4, 13.7, 13.8)

"Lecture Notes on Cryptographic Protocols" (ch **5.4.0,5.4.1**, all ch 5)

"Efficient Secure Two-Party Protocols" by C. Hazay & H. Lindell (ch 3.3, 3.4)

Module 3: Agenda

Introduction to MPC
Commitment Schemes (Pedersen)
(Verifiable) Secret Sharing (Shamir)
Oblivious Transfer

MPC Security
Zero-Knowledge Proofs
Σ (Sigma) Protocols
- Schnorr

Σ (Sigma) Protocols
- Knowledge of Pedersen Commitments

HA3

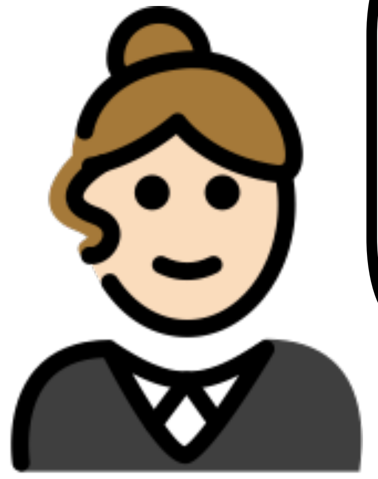
Removing Interaction
- Fiat-Shamir Heuristic

Generic 2 Party Computation
- Garbled Circuits
- Yao’s Two Party Protocol

Proving Knowledge of Pedersen Commitments

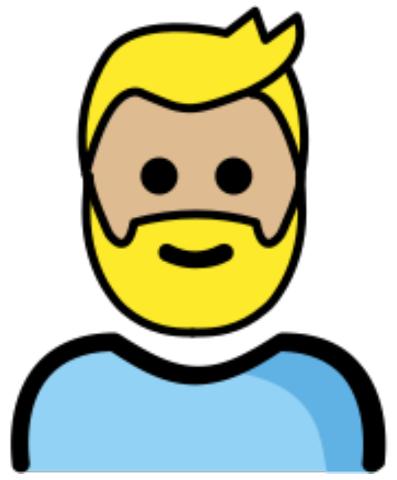
Setup(sec.par) → ($\mathbb{G}$, q, g, h)
Commit(m,r) = $g^m h^r$ mod q =: c
Open(m,r,c) = 1 if c = $g^m h^r$ mod q,
and 0 otherwise

From Lecture 9 in Module 3



Prover

c is a well-formed commitment
(I know m, r s.t. $c = g^m h^r$)



Verifier

A: $r_1, r_2 \leftarrow \$\mathbb{Z}_q$; $a = g^{r_1} h^{r_2} \in \mathbb{G}$

$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad e \leftarrow \$\{0,1\}^t \quad}$

Z: $z_1 = r_1 + e \cdot m \in \mathbb{Z}_q$; $z_2 = r_2 + e \cdot r \in \mathbb{Z}_q$

$\xrightarrow{\quad z = (z_1, z_2) \quad}$

$0/1 \leftarrow V(a, e, z)$

check that $z \in \mathbb{Z}_q \times \mathbb{Z}_q$
if $g^{z_1} h^{z_2} = c^e \cdot a$ return 1
else return 0

Module 3: Agenda

Introduction to MPC
Commitment Schemes (Pedersen)
(Verifiable) Secret Sharing (Shamir)
Oblivious Transfer

MPC Security
Zero-Knowledge Proofs
Σ (Sigma) Protocols
- Schnorr

Σ (Sigma) Protocols
- Knowledge of Pedersen Commitments

Removing Interaction
- Fiat-Shamir Heuristic **HA3**

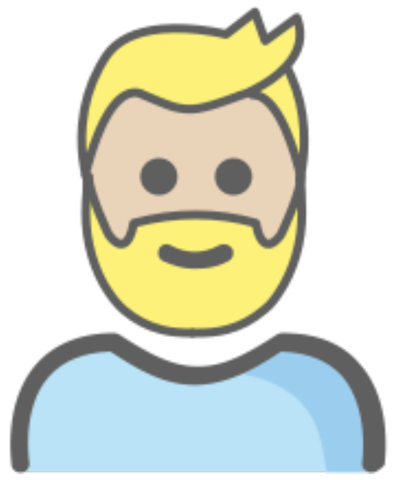
Generic 2 Party Computation
- Garbled Circuits
- Yao's Two Party Protocol

Schnorr Σ-Protocol for Knowledge of dLog

From Lecture 10 in Module 3

I know w s.t. $x = g^w$

Prover



Verifier

$r \leftarrow \$\mathbb{Z}_q$
$a = g^r \in \mathbb{G}$

$a \leftarrow A(w, r)$ →

$0/1 \leftarrow V(a, e, z)$

← $e \leftarrow \$\{0,1\}^t$

$z' = a \cdot x^e \in \mathbb{G}$
if $g^z = z'$ return 1
else return 0

$z = r + e \cdot w \in \mathbb{Z}_q$

$z \leftarrow Z(w, r, e)$ →

public inputs (available to both P and V)

- the description of a group $\mathbb{G}$ of prime order q with generator g
- the value $x \in \mathbb{G}$
- $R(x, y) : \mathbb{G} \times \mathbb{Z}_q \rightarrow \{0,1\}$, defined as $R(x, y) = 1 \; iff \; x = g^y$
- the challenge length $t = log_2(q) \in \mathbb{N}$

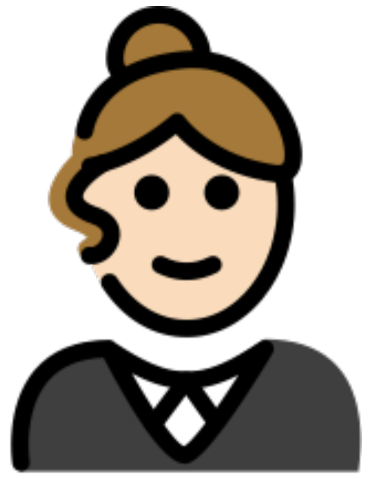
Completeness

Special Soundness

Special HV ZK

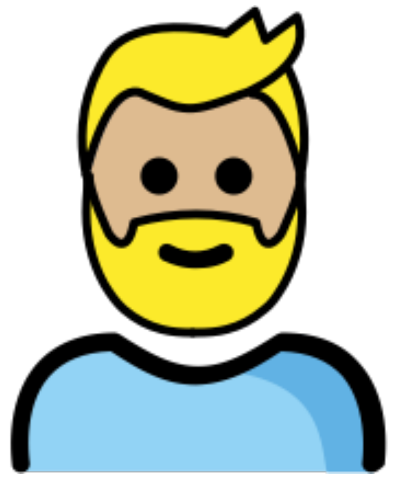
whiteboard proofs

Schnorr Σ-Protocol for Knowledge of dLog NON-Interactive?



Prover

I know w s.t. $x = g^w$



Verifier

$r \leftarrow \$\mathbb{Z}_q$
$a = g^r \in \mathbb{G}$

$a \leftarrow A(w, r)$ →

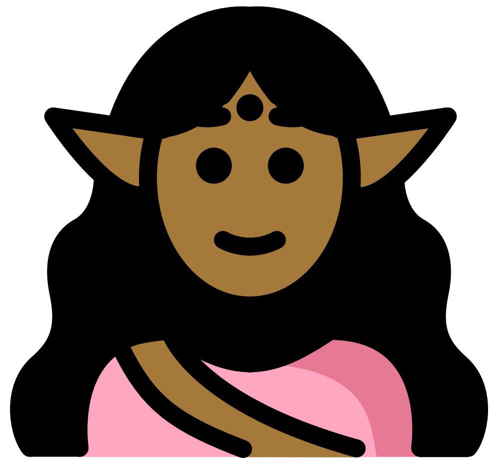
$\cancel{e \leftarrow \$\{0,1\}^t}$ ←

$z = r + e \cdot w \ \in \mathbb{Z}_q$

$z \leftarrow Z(w, r, e)$ →

$0/1 \leftarrow V(a, e, z)$

$z' = a \cdot x^e \in \mathbb{G}$
if $g^z = z'$ return 1
else return 0



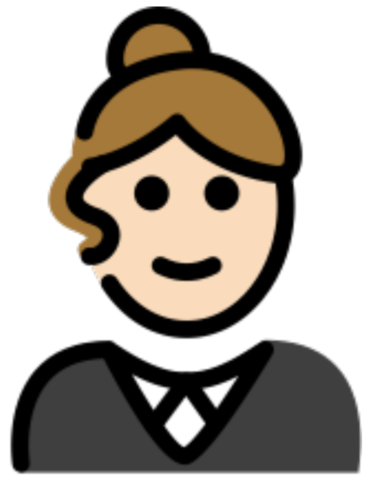
Verifier

Verifier

🧐 *how to generate a fresh/ good challenge e?*

Fiat-Shamir Heuristic model the hash function as a random oracle and compute the challenge as $e = H(g, x, a)$

Schnorr Σ-Protocol for Knowledge of dLog NON-Interactive?



Prover

I know w s.t. $x = g^w$

$$\boxed{\begin{array}{l} r \leftarrow \$\mathbb{Z}_q \\ a = g^r \in \mathbb{G} \end{array}}$$

$$e = H(g, x, a) \qquad m$$

$$\boxed{z = r + e \cdot w \ \in \mathbb{Z}_q}$$

🧐 *where is the message?*

Recipe to create a digital signature from a ZK Proof

1. pick randomness
2. generate new (unpredictable) randomness using the hash function
3. use the secret and hide it with both of the randomnesses
4. return a proof of knowledge of the secret value

This is the same procedure as ECDSA is built

Fiat-Shamir Heuristic model the hash function as a random oracle and compute the challenge as $e = H(g, x, a)$

Module 3: Agenda

Introduction to MPC
Commitment Schemes (Pedersen)
(Verifiable) Secret Sharing (Shamir)
Oblivious Transfer

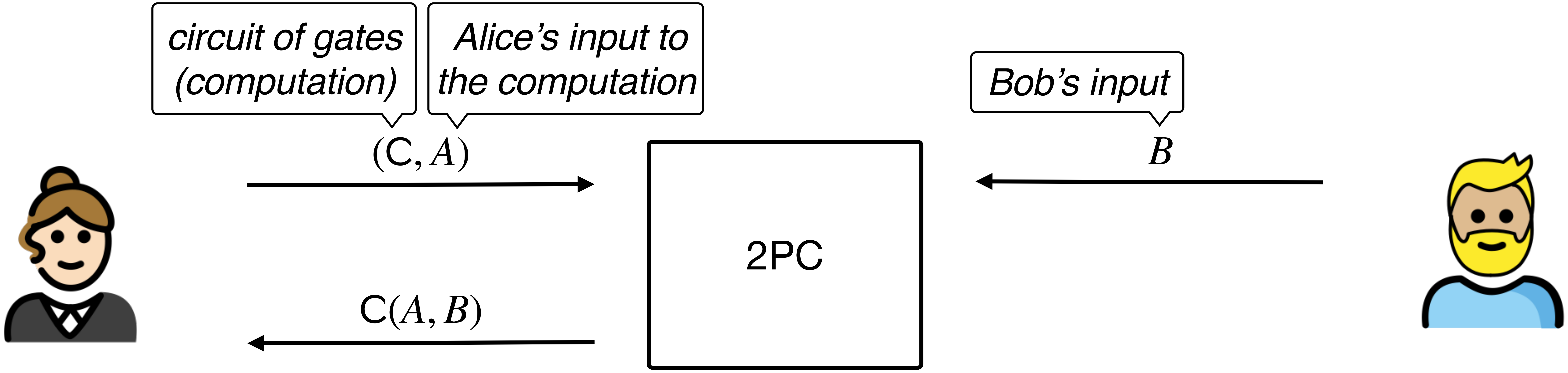
MPC Security
Zero-Knowledge Proofs
Σ (Sigma) Protocols
- Schnorr

Σ (Sigma) Protocols
- Knowledge of Pedersen Commitments

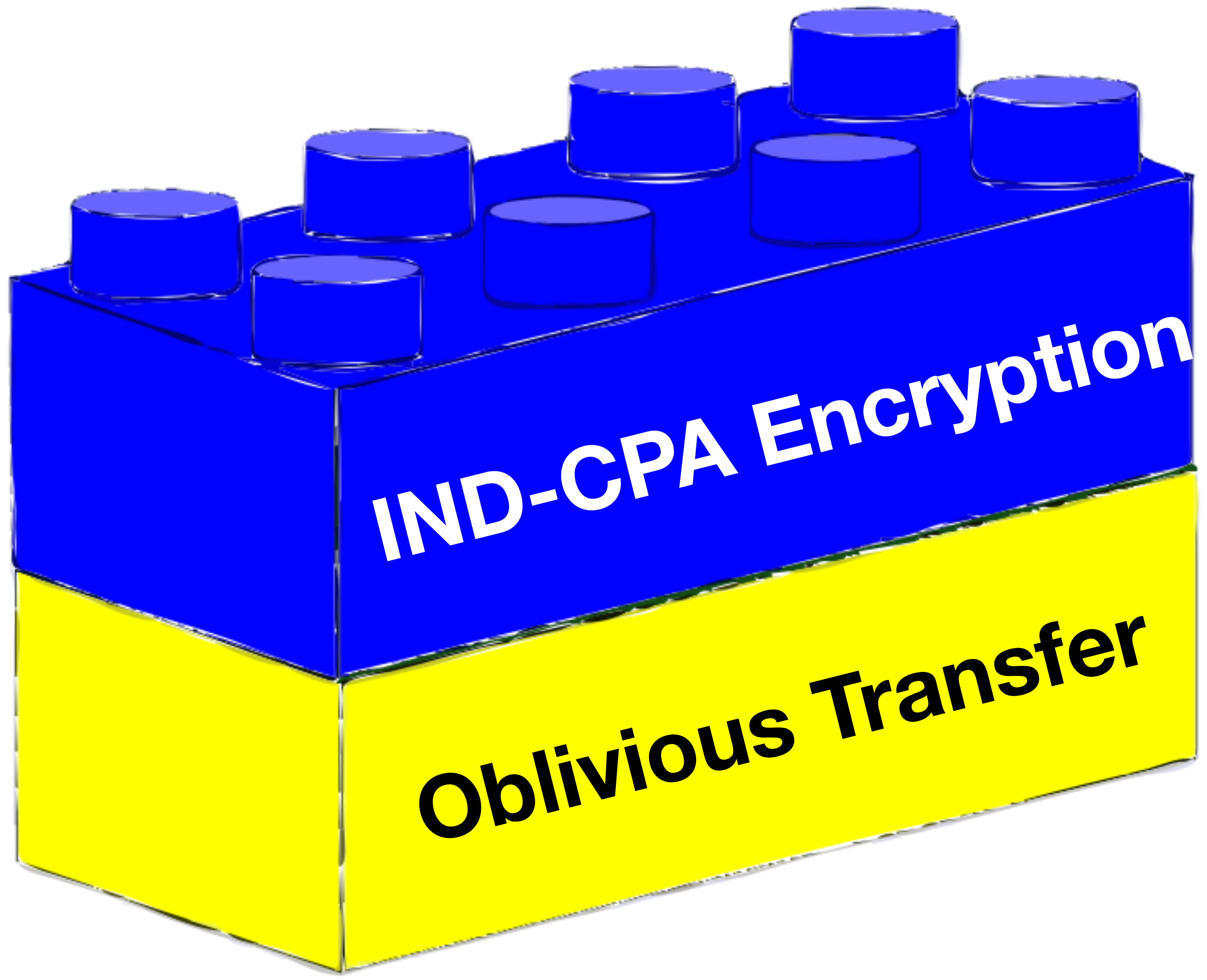
Removing Interaction
- Fiat-Shamir Heuristic

Generic 2 Party Computation
- Garbled Circuits
- Yao’s Two Party Protocol

Generic Two Party Computation (2PC)



$$\mathscr{F}(\{\mathsf{C}, A\}, B) \mapsto (\mathsf{C}(A, B), \emptyset)$$

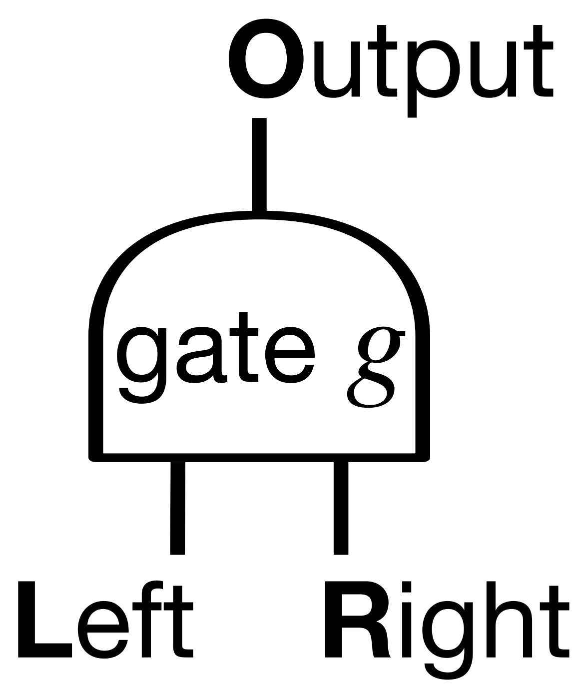


Building Garbled Circuits

confused and distorted, unclear

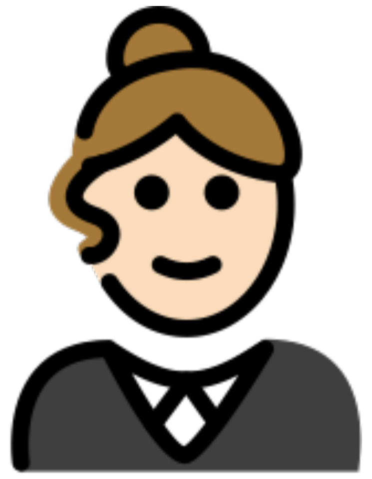
any boolean function can be represented as a boolean circuit C *composed only of AND gates and XOR gates*

$$g : \{0,1\} \times \{0,1\} \to \{0,1\}$$
$$(w_L, w_R) \mapsto w_O = g(w_L, w_R)$$



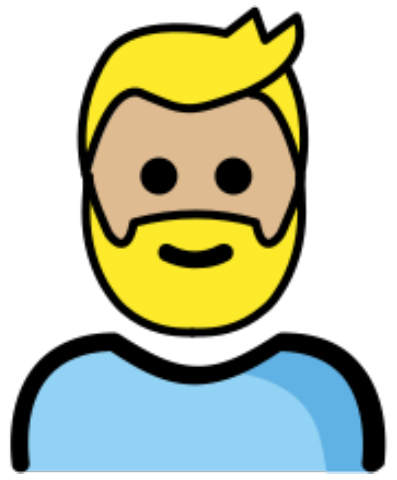
how to model a 'gate' mathematically?

$$\mathcal{F}_{\mathsf{C}}(A, B) \mapsto (\mathsf{C}(A, B), \emptyset)$$



A

B



$\mathtt{C} \leftarrow Garble(\mathsf{C}, A)$

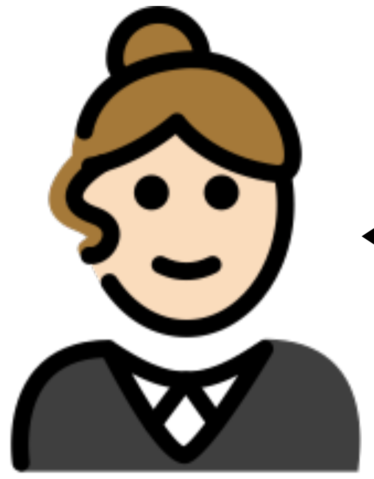
$\xrightarrow{\mathtt{C}}$

$\xleftarrow{\mathtt{c}}$

$out_A \leftarrow UnGarble(\mathtt{c})$

$\mathtt{c} \leftarrow Eval(\mathtt{C}, B)$

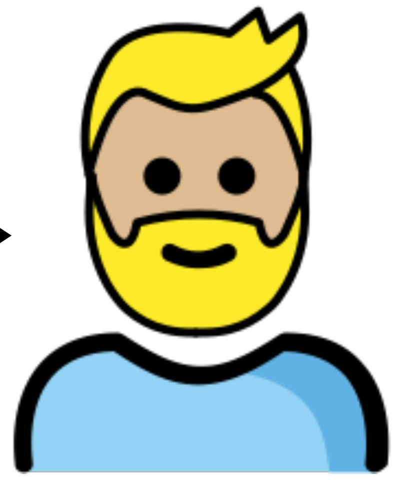
Garbling a Gate



A

compute $g(A, B)$ for Alice

B



1 pick 6 random strings:

$K_L^b, K_R^b, K_O^b \leftarrow \$\{0,1\}^\lambda, \textit{for } b \in \{0,1\}$

security parameter

2 garble the truth table of g

L	R	O
k_L^0	k_R^0	k_O^0
k_L^0	k_R^1	k_O^0
k_L^1	k_R^0	k_O^0
k_L^1	k_R^1	k_O^1

$k_O^{g(w_L, w_R)}$

truth table for AND gate

Attempt 1

3 send to Bob the garbled truth table

4 select the 2 rows with k_R^B and send them to Alice

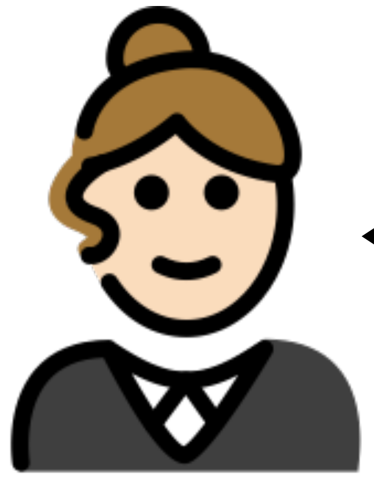
$\left(k_L^0, k_R^B, k_O^{g(0,B)}\right), \left(k_L^1, k_R^B, k_O^{g(1,B)}\right)$

5 select the row with k_L^A and decode $C = g(A, B)$ from k_O^C

🧐 what's the issue?

if Bob gets to select items provided by Alice, his choice unavoidably leaks his input value

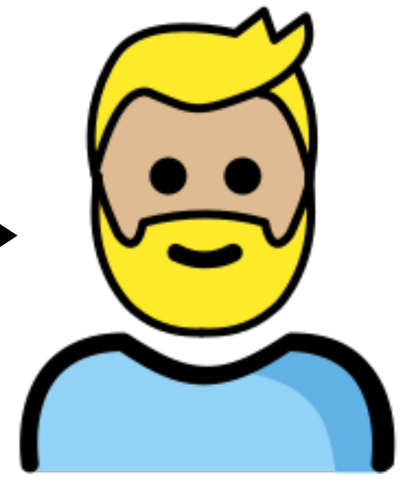
Garbling a Gate



A

compute $g(A, B)$ for Alice

B



1 pick 6 random strings:

$$K_L^b, K_R^b, K_O^b \leftarrow \$\{0,1\}^\lambda, \textit{for } b \in \{0,1\}$$

2 garble the truth table of g

L	R	O
k_L^0	k_R^0	k_O^0
k_L^0	k_R^1	k_O^0
k_L^1	k_R^0	k_O^0
k_L^1	k_R^1	k_O^1

$k_O^{g(w_L, w_R)}$

Attempt 2

3 send to Bob the *sub-table* for $g(A, \cdot\,)$

this would be $\mathrm{Garble}(g, A)$

select the row with k_R^B and send to Alice the corresponding k_O^C 4

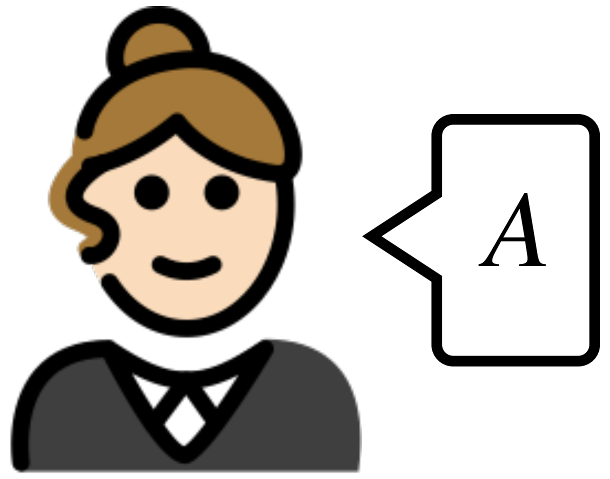
$\left(k_O^{g(0,B)}, k_O^{g(1,B)}\right)$

4 decode $C = g(A, B)$ from k_O^C

🧐 what's the issue?

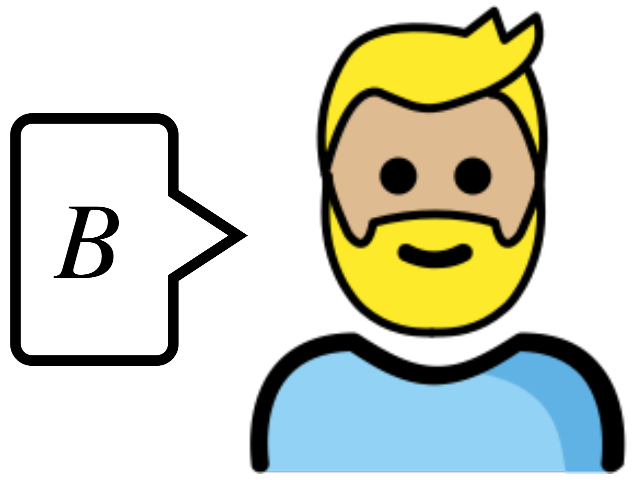
if Alice selects items for Bob, her choice may leak her input value or Bob's

Garbling a Gate



compute $g(A, B)$ for Alice

IND-CPA encryption scheme
$(KeyGen, Enc, Dec)$ symmetric



1 pick 6 random strings:
$K_L^b, K_R^b, K_O^b \leftarrow \$\{0,1\}^\lambda, \mathit{for}\ b \in \{0,1\}$

2 garble the **outputs** of g as

$$c_{0,0} = Enc_{k_L^0}\left(Enc_{k_R^0}(k_O^0)\right)$$
$$c_{0,1} = Enc_{k_L^0}\left(Enc_{k_R^1}(k_O^0)\right)$$
$$c_{1,0} = Enc_{k_L^1}\left(Enc_{k_R^0}(k_O^0)\right)$$
$$c_{1,1} = Enc_{k_L^1}\left(Enc_{k_R^1}(k_O^1)\right)$$

Attempt 3

3 send all cipher texts to Bob

4 select and return to Alice $(c_{0,B}, c_{1,B})$

5 decrypt
$C = g(A, B) = Dec_{k_L^A}\left(Dec_{k_R^B}(c_{A,B})\right)$

$\mathcal{F}_{OT}(\{x_0, x_1\}, b) \mapsto (\emptyset, x_b)$

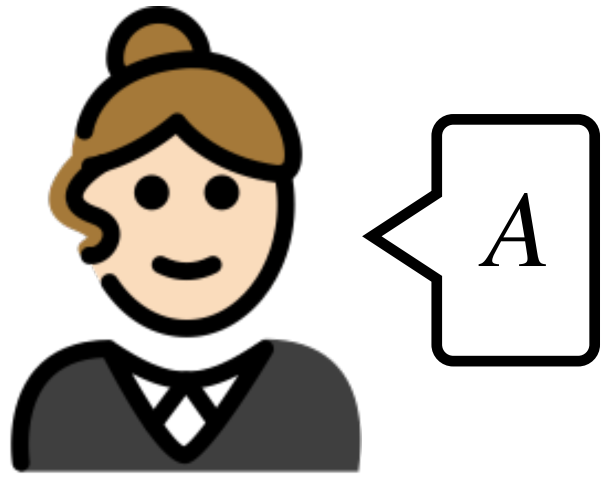
what's the issue?

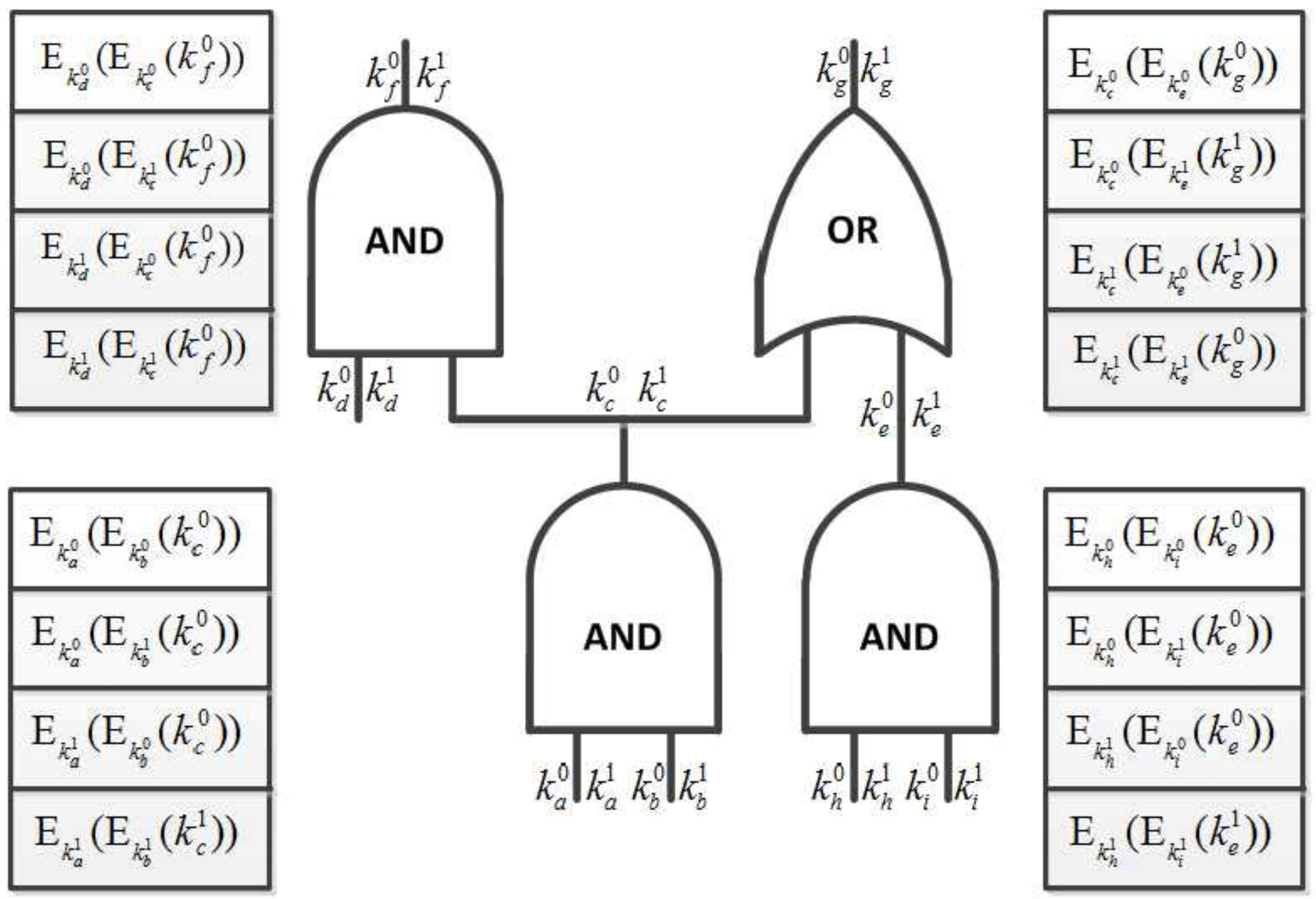
in this setting, Encryption alone does not provide privacy (Bob's selection may leak info)

Garbling a Circuit

Final Attempt

1- Garbling Phase





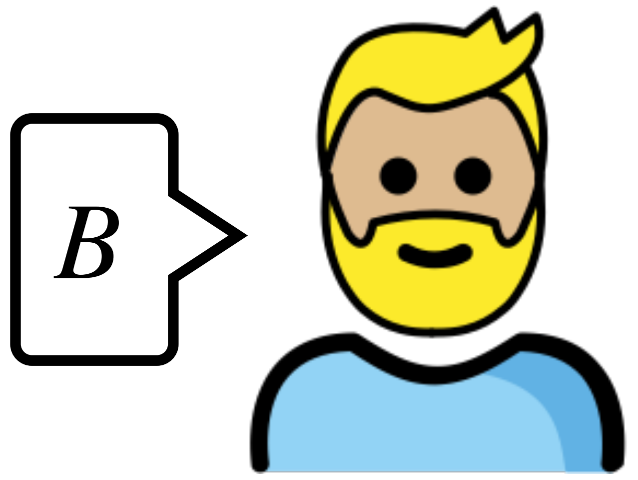
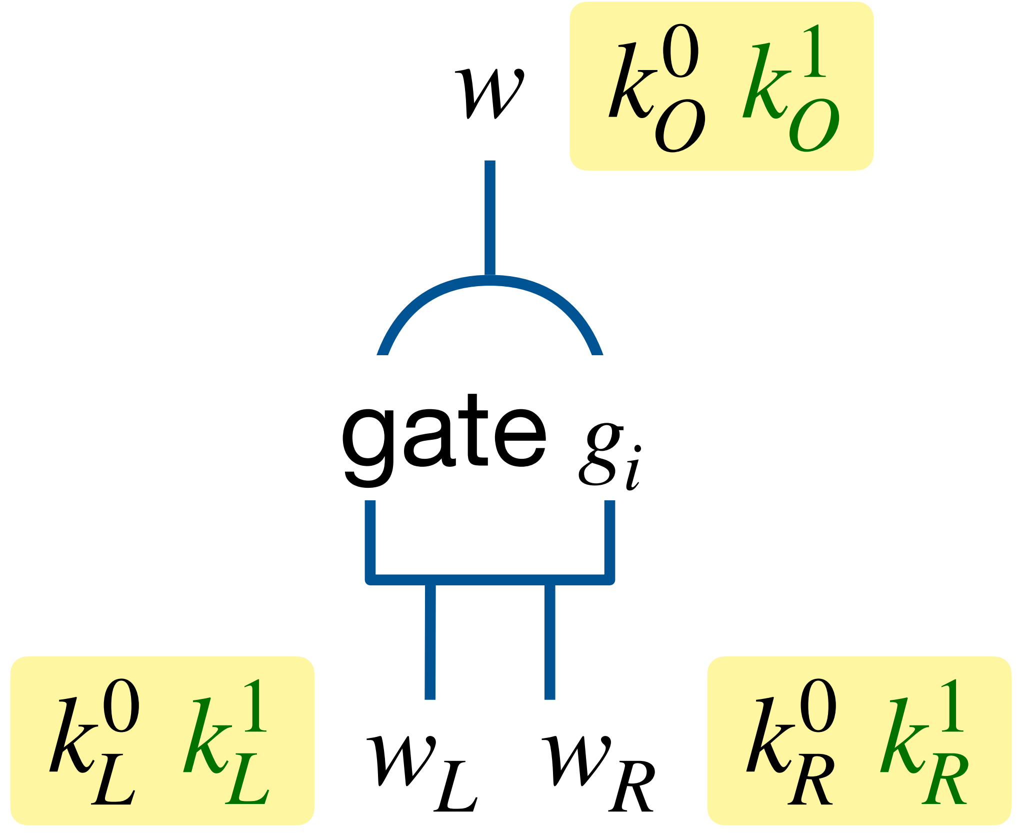


Image from:
Wang et al. "Reusable garbled gates for new fully homomorphic encryption service" *International journal of web and grid services* (2017)

1 Pick keys for each gate in **C**



2 Compute the garbled output of each gate

$$c^g_{\alpha,\beta} = Enc_{k_L^\alpha}\left(Enc_{k_R^\beta}[k_O^{g(\alpha,\beta)} \,||\, \mathbf{0}]\right)$$

$$\mathbb{C} = \left(\{c^{g_i}_{\alpha,\beta}\}^{i=1,\ldots,\#gates}_{\alpha,\beta\in\{0,1\}}, \{k^A_{Alice's\ wire}\} \right) \xrightarrow{\text{send}}$$

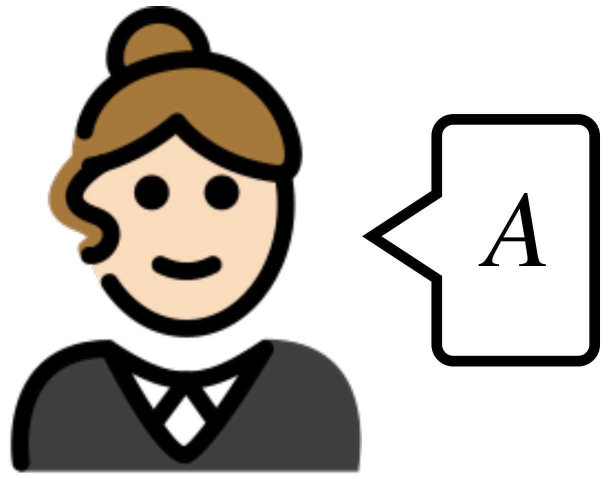
*4 cipher texts for each gate in **C***

1 key for each input wire of Alice

3 *1 pair of keys for each input wire of Bob*

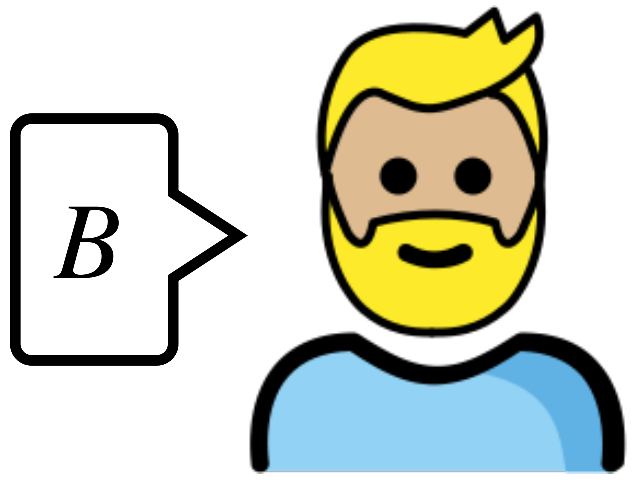
sender $\{k^0_{Bob}, k^1_{Bob}\} \longrightarrow$ run an Oblivious Transfer protocol $\longleftarrow B$ $\longrightarrow k^B_{Bob}$ **receiver** 3

Garbling a Circuit



Final Attempt

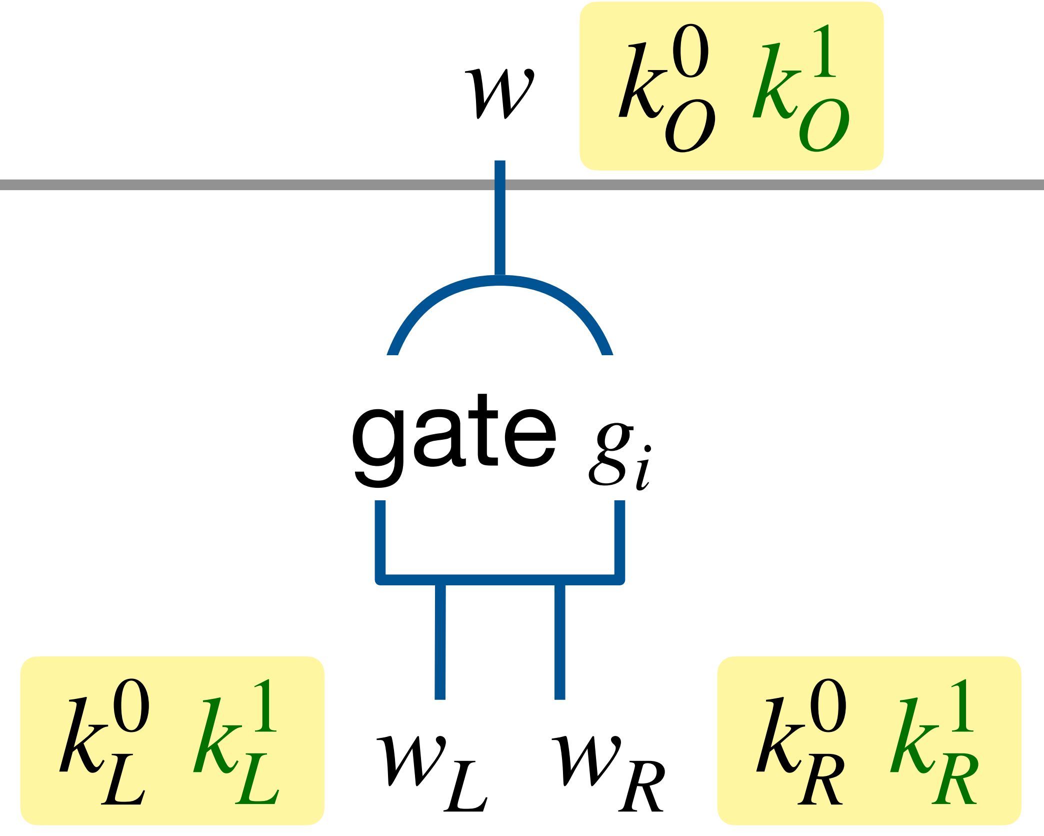
1- Garbling Phase



1 Pick keys for each gate in **C**

$$k^B_{Bob} \quad C = \left(\{c^{g_i}_{\alpha,\beta}\}^{=1,\ldots,\#gates}_{\alpha,\beta\in\{0,1\}}, \{k^A_{Alice}\} \right)$$

send



2 - Evaluation Phase

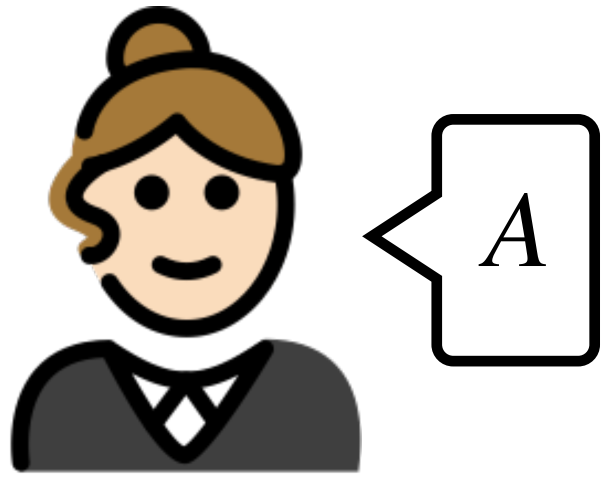
2 Compute the garbled output of each gate

$$c^g_{\alpha,\beta} = Enc_{k^\alpha_L}\left(Enc_{k^\beta_R}[k^{g(\alpha,\beta)}_O \,||\, \mathbf{0}]\right)$$

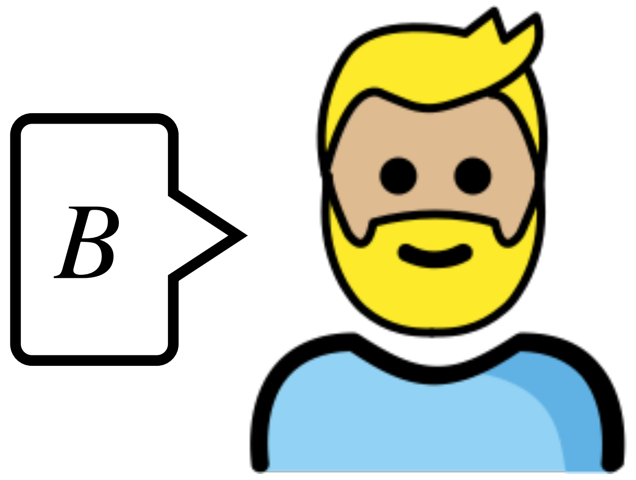
this $\mathbf{0}$ value is simply a correctness check that helps Bob understand which key $k^{\square}_O$ to use to proceed with the garbled evaluation.

Garbling a Circuit

Final Attempt



1- Garbling Phase

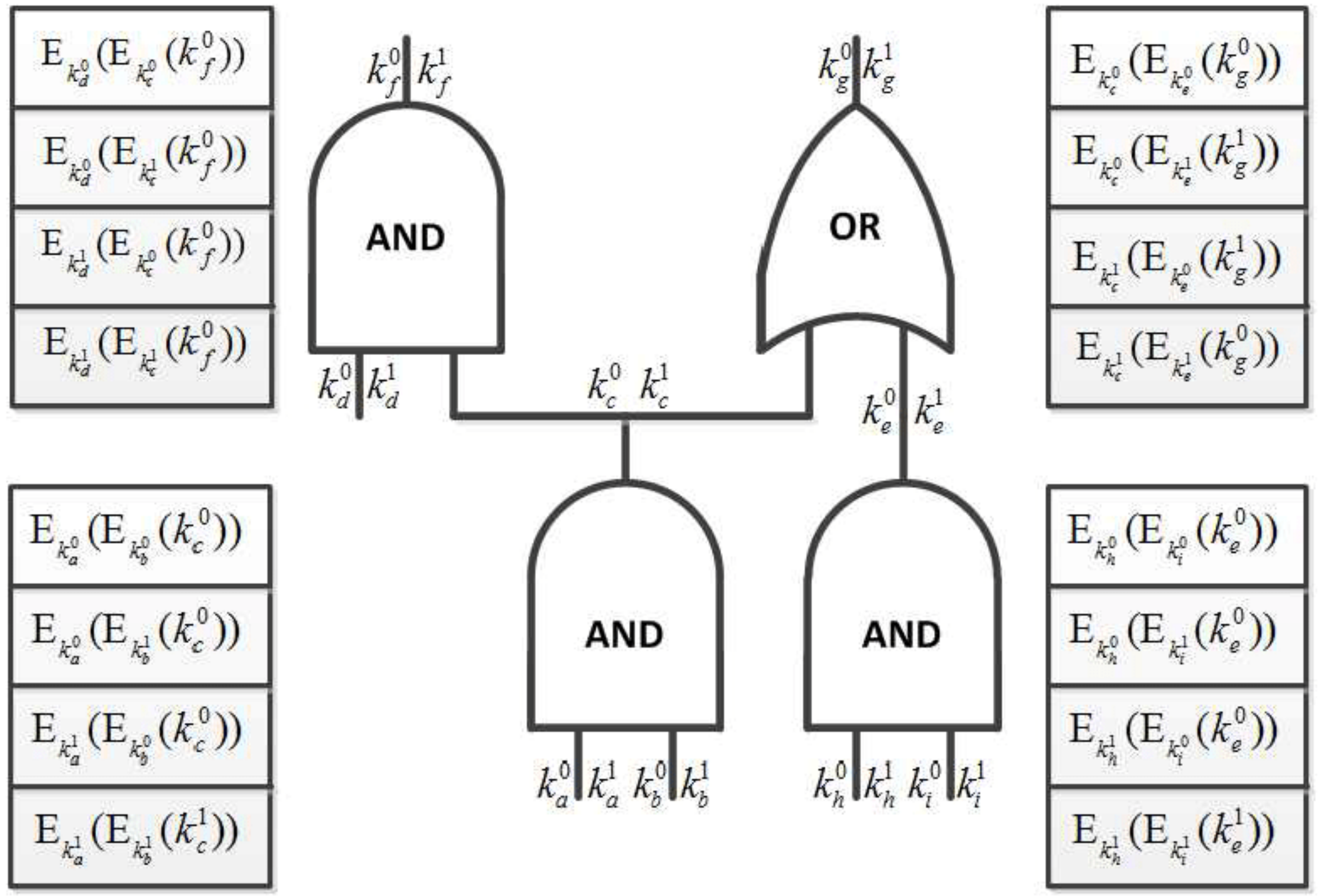


k^{B}_{Bob}

$$\mathsf{C} = \left(\{c^{g_i}_{\alpha,\beta}\}^{=1,\dots,\#gates}_{\alpha,\beta\in\{0,1\}}, \{k^{A}_{Alice}\}\right) \xrightarrow{\text{send}}$$

---

2 - Evaluation Phase



1 sequentially evaluate all gates in **C**

this means Bob progressively decrypts each 4-tuple of ciphertexts using the keys he has at hand and proceeds using the key that decrypts to $k_O^{\square}||\mathbf{0}$ (i.e., a bit strings that ends with a fixed number of 0s)

2 send the garbled value of the output wire to Alice

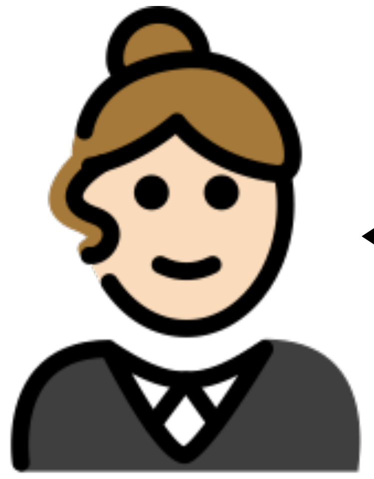
$$\xleftarrow{\text{send}} \quad \mathsf{c} = k_O^{\mathbf{C}(A,B)}$$

Garbling a Circuit

Final Attempt

1- Garbling Phase

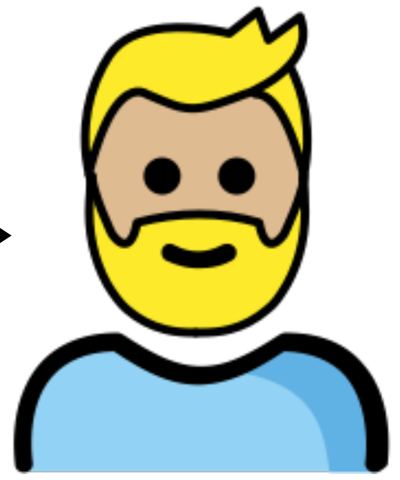
2 - Evaluation Phase



A

send $\longleftarrow$

$c = k_O^{\mathbf{C}(A,B)}$



B

send $\longrightarrow$

k_{Bob}^{B} $\quad C = \left(\{c^{g_i}_{\alpha,\beta}\}^{=1,\dots,\#gates}_{\alpha,\beta\in\{0,1\}}, \{k^{A}_{Alice}\} \right)$

---

3 - Output Phase

1 lookup the garbled truth table for $g_{\#gates}$

L	R	O
k_L^0	k_R^0	k_O^0
k_L^0	k_R^1	k_O^0
k_L^1	k_R^0	k_O^0
k_L^1	k_R^1	k_O^1

$k_O^{\mathbf{C}(A,B)}$

2 find $k_O^{\mathbf{C}(A,B)}$ and learn the bit $\mathbf{C}(A,B)$

This is Yao's protocol for generic secure two party computation !

Recipe To Compute any (Boolean) Function With 2PC

This is Yao's protocol for generic secure two party computation !

1. Alice picks a secret key $k_{\triangle}^{\square}$ for every possible input/output of the gate
2. Alice send her input keys, and the truth-table cipher texts for each gate to Bob
3. Bob evaluates the garbled gate on its input and sends the outcome back to Alice (this is a secret key $k^{\bullet}$)
4. Alice decodes the value $\bullet = \mathbf{C}(A, B)$ using $k^{\bullet}$ and the garbled truth table of the final gate of the circuit