Advanced Crypto 2024
Multi-Party Computing

Léo Colisson Palais

leo.colisson-palais@univ-grenoble-alpes.fr

`https://leo.colisson.me/teaching.html`

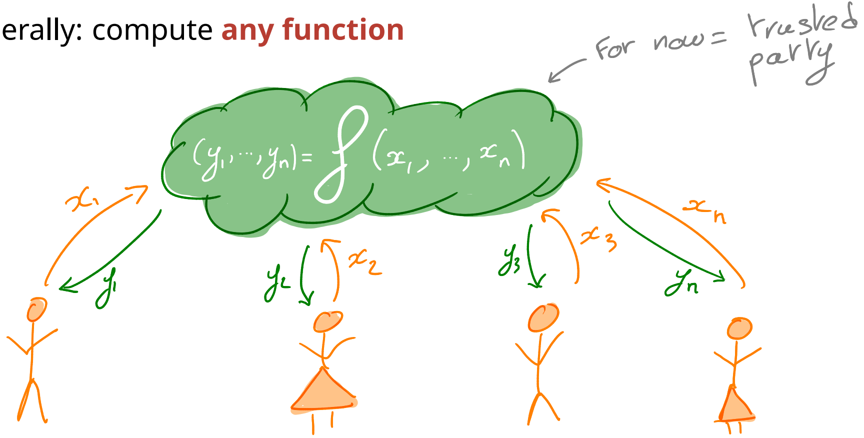
Motivations


?
<
Multi-Party Computing (MPC)

Motivation MPC

Millionaire's problem: find the richest person in a group without revealing the individual fortunes

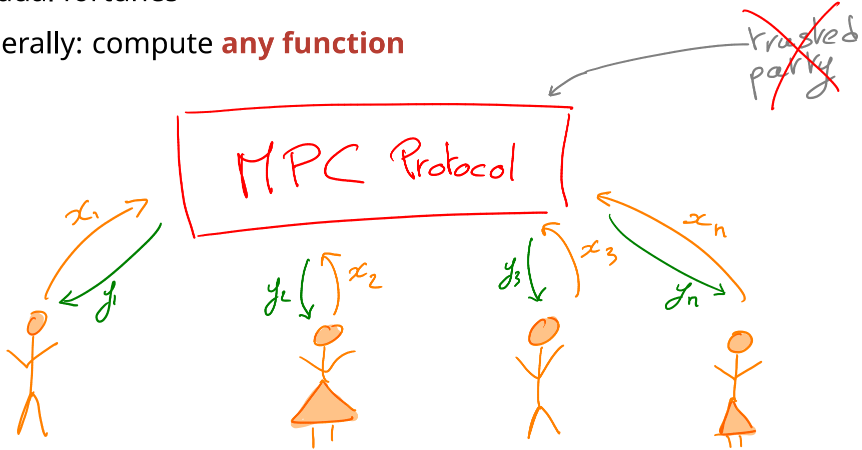
More generally: compute **any function**



Motivation MPC

Millionaire's problem: find the richest person in a group without revealing the individual fortunes

More generally: compute **any function**



Motivation MPC

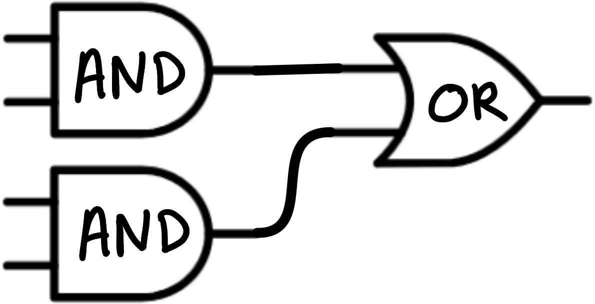
In the millionaire's problem, what is the implemented function $f(x_1, \ldots, x_n)$? (same f for all parties)

- (A) Max
- (B) Min
- (C) ArgMax
- (D) ArgMin

2-party MPC:
Garbled circuits

Yao's garbled circuit [Yao86]

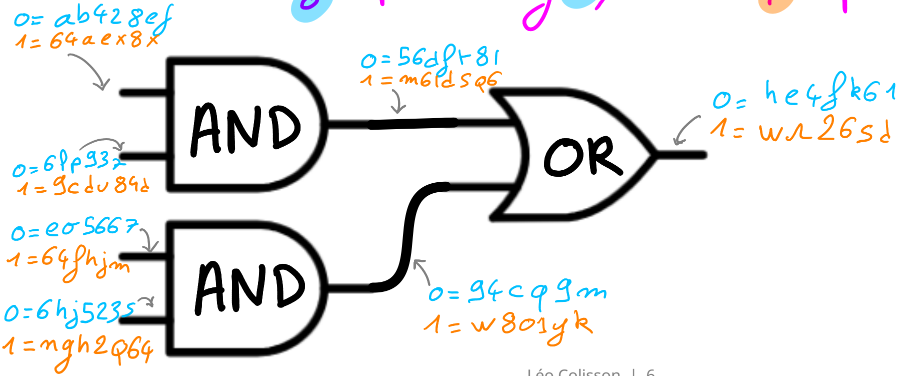
Alice ↳ Step 1 : describe f as a circuit



Alice

Step 1: describe f as a circuit

Step 2: for each wire w, assign random labels w_0 representing 0, and w_1 representing 1.



Yao's garbled circuit [Yao86]

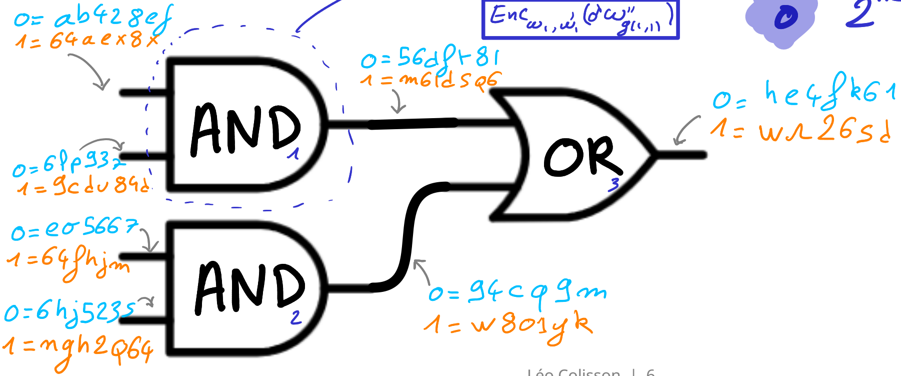
Alice ↪ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

(input wires: w, w'; output wire: w'')

$Enc(G) =$
$Enc_{w_0, w'_0}(\sigma w''_{g(0,0)})$
$Enc_{w_0, w'_1}(\sigma w''_{g(0,1)})$
$Enc_{w_1, w'_0}(\sigma w''_{g(1,0)})$
$Enc_{w_1, w'_1}(\sigma w''_{g(1,1)})$

? What is on the 4th line of the 2nd garbled gate?



Yao's garbled circuit [Yao86]

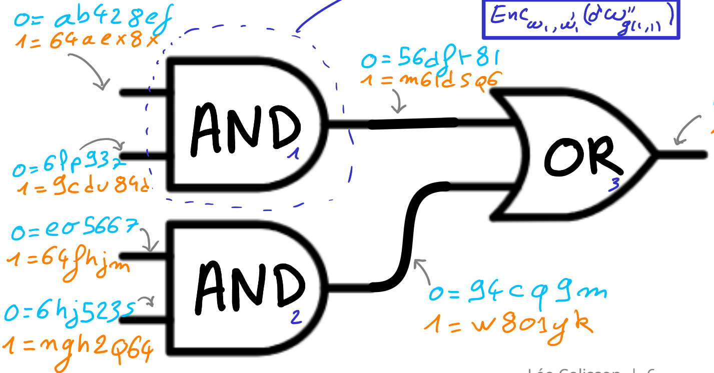
Alice ↳ Step 3: Alice "encrypts" (= garble) each gate G(w, w') -> w'' (w, w': input wires; w'': output wire) as a table:

Enc(G) =
$Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

? What is on the 4th line of the 2nd garbled gate?

⇓

$Enc_{64fhjm\,ngh2q64}(0^\lambda w801yk)$

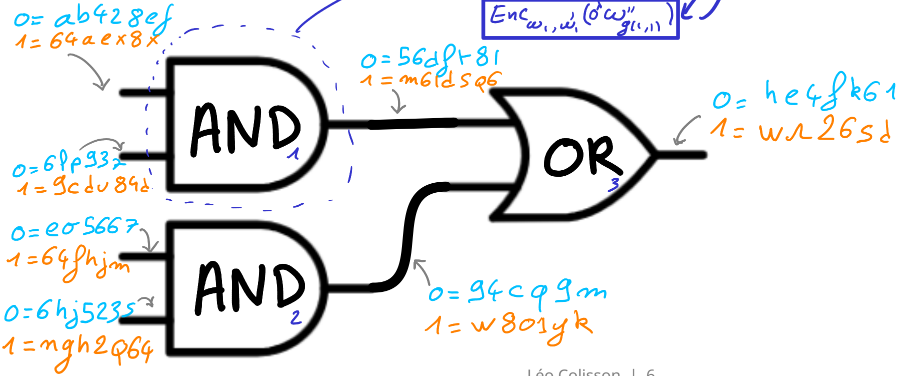


Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table: and randomly permutes it. + send to Bob

T_G =

Enc(G) =
$Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$



Yao's garbled circuit [Yao86]

Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

T_G =

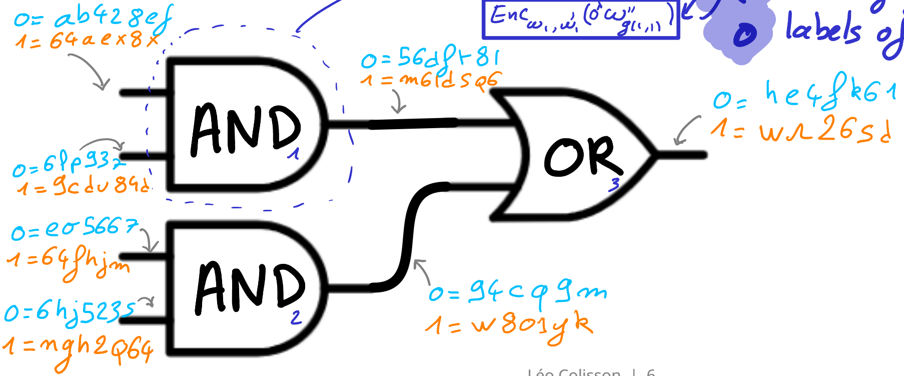
Enc(G) =
$Enc_{w_0, w'_0}(0^{\lambda} w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^{\lambda} w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^{\lambda} w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^{\lambda} w''_{g(1,1)})$

and randomly permutes it.

+ send to Bob

?

How can you evaluate C if you know the labels of the inputs?



Yao's garbled circuit [Yao86]

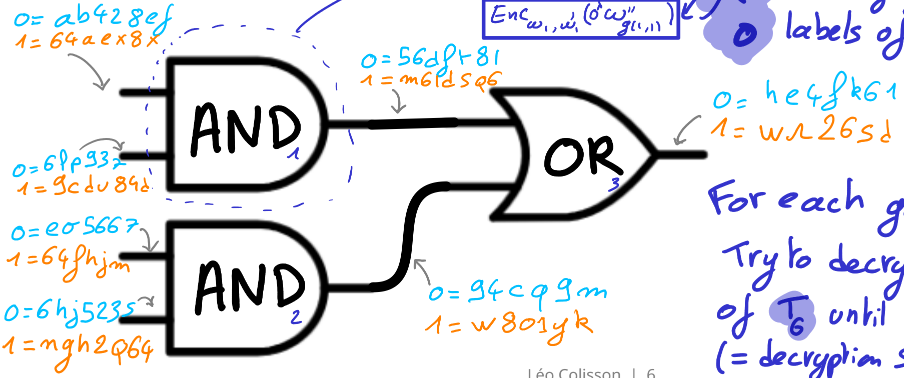
Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table: and randomly permutes it. + send to Bob

$T_G =$

Enc(G) =
$Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

? How can you evaluate C if you know the labels of the inputs?



For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ Step 3

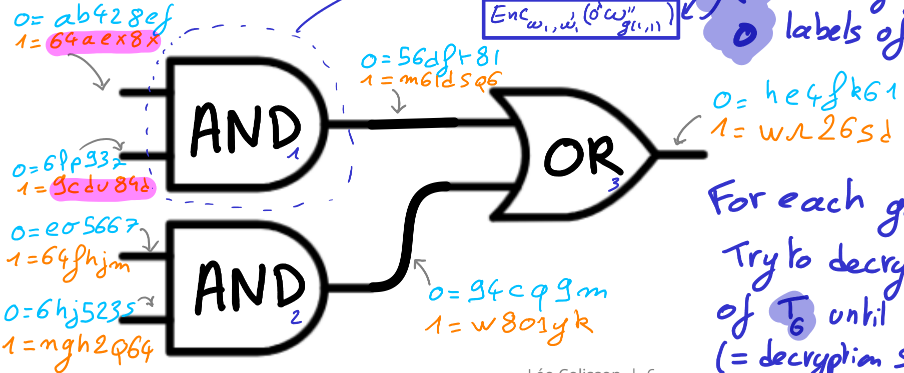
Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

T_G =

Enc(G) =
$Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

and randomly permutes it.
+ send to Bob

? How can you evaluate C if you know the labels of the inputs?



For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ Step 3

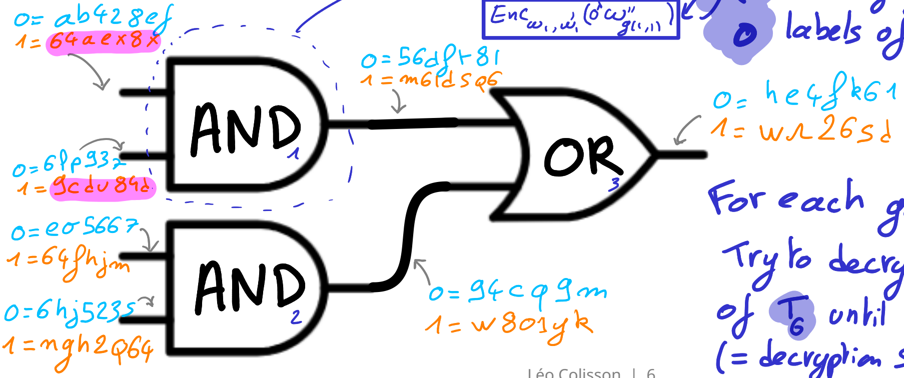
Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

$T_G =$

Enc(G) =
$Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
$Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
$Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

and randomly permutes it. + send to Bob

? How can you evaluate C if you know the labels of the inputs?



For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ **Step 3**

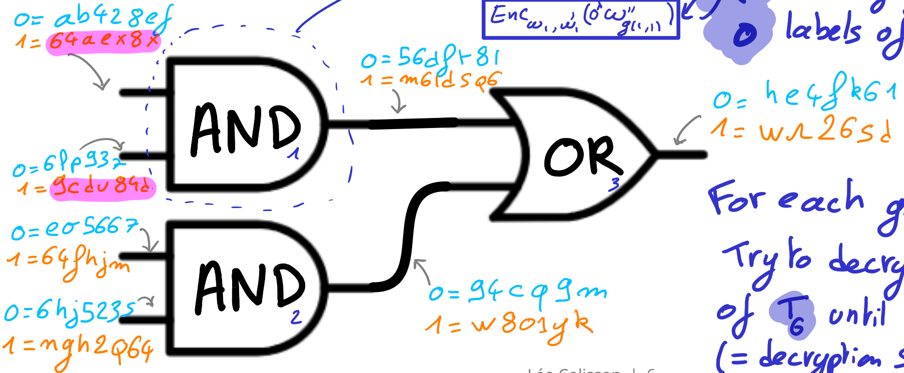
Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

$$T_G = \text{Enc}(G) = \begin{array}{|c|} \hline \text{Enc}_{w_0, w'_0}(0^\lambda w''_{g(0,0)}) \\ \hline \text{Enc}_{w_0, w'_1}(0^\lambda w''_{g(0,1)}) \\ \hline \text{Enc}_{w_1, w'_0}(0^\lambda w''_{g(1,0)}) \\ \hline \text{Enc}_{w_1, w'_1}(0^\lambda w''_{g(1,1)}) \\ \hline \end{array}$$

and randomly permutes it.

+ send to Bob

How can you evaluate C if you know the labels of the inputs?



For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ **Step 3**

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

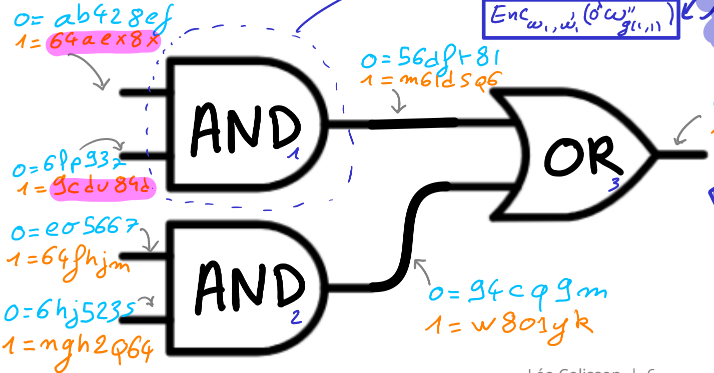
$T_G =$

$Enc(G) =$
✗ $Enc_{w_0,w'_0}(0^\lambda w''_{g(0,0)})$
✗ $Enc_{w_0,w'_1}(0^\lambda w''_{g(0,1)})$
✗ $Enc_{w_1,w'_0}(0^\lambda w''_{g(1,0)})$
$Enc_{w_1,w'_1}(0^\lambda w''_{g(1,1)})$

and randomly permutes it.

+ send to Bob

? How can you evaluate C if you know the labels of the inputs?



0= he4fk61
1= wn26sd

For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ Step 3

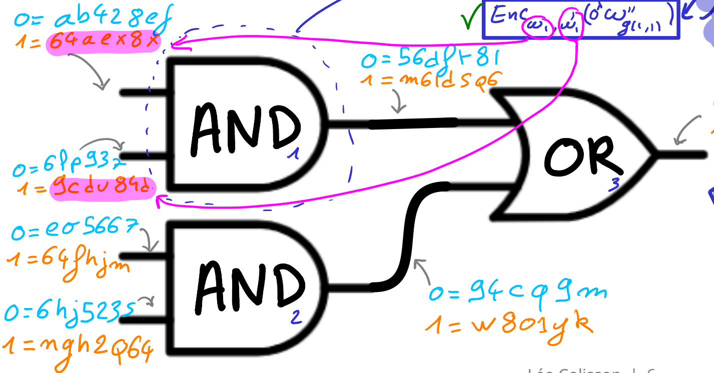
Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

T_G = Enc(G) =

✗ $Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
✗ $Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
✗ $Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
✓ $Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

and randomly permutes it. + send to Bob

? How can you evaluate C if you know the labels of the inputs?



For each gate G:

Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!

Yao's garbled circuit [Yao86]

Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table: and randomly permutes it. + send to Bob

T_G =

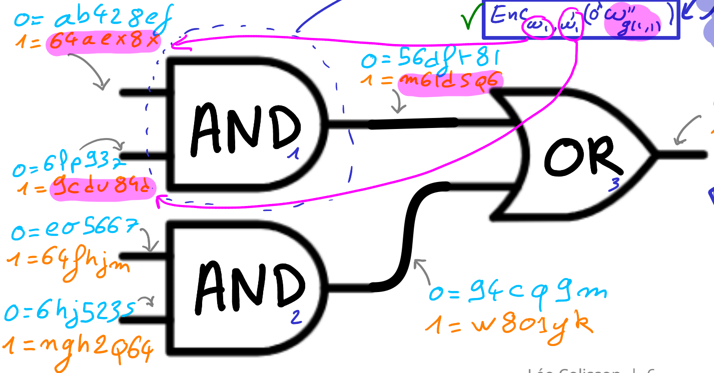
Enc(G) =
✗ $\mathrm{Enc}_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
✗ $\mathrm{Enc}_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
✗ $\mathrm{Enc}_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
✓ $\mathrm{Enc}_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

?

How can you evaluate C if you know the labels of the inputs?

Done by Bob

For each gate G: Try to decrypt each line of T_G until we succeed (= decryption starts with 0^λ)!



Yao's garbled circuit [Yao86]

Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

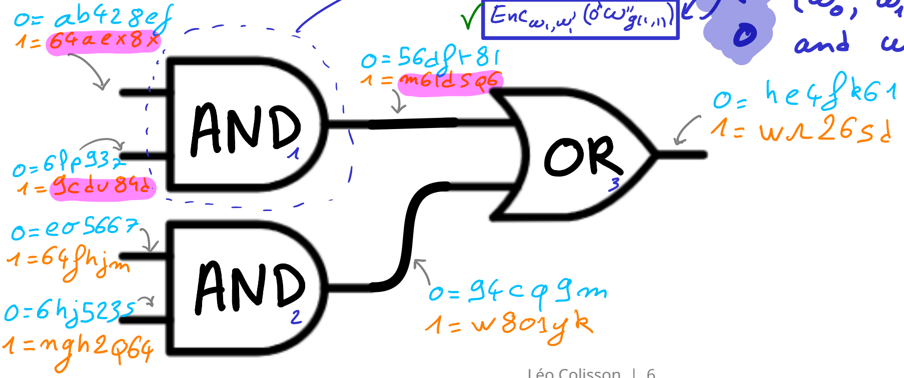
T_G =

Enc(G) =
✗ $Enc_{w_0, w'_0}(0 \, w''_{g(0,0)})$
✗ $Enc_{w_0, w'_1}(0 \, w''_{g(0,1)})$
✗ $Enc_{w_1, w'_0}(0 \, w''_{g(1,0)})$
✓ $Enc_{w_1, w'_1}(0 \, w''_{g(1,1)})$

and randomly permutes it. + send to Bob

If Bob knows G, $\{w_0, w_1\}$ (but not the order) and w_b, can he know b?

?



Yao's garbled circuit [Yao86]

Alice ↳ Step 3

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table: and randomly permutes it. + send to Bob

$T_G =$

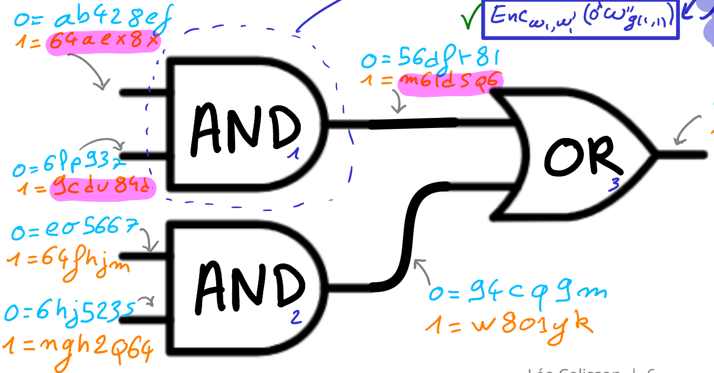
Enc(G) =
✗ $Enc_{w_0, w'_0}(0^\lambda w''_{g(0,0)})$
✗ $Enc_{w_0, w'_1}(0^\lambda w''_{g(0,1)})$
✗ $Enc_{w_1, w'_0}(0^\lambda w''_{g(1,0)})$
✓ $Enc_{w_1, w'_1}(0^\lambda w''_{g(1,1)})$

?

If Bob knows G, $\{w_0, w_1\}$ (but not the order) and w_b, can he know b?

⇓

Yes! Idea: decrypt table = allow to know the label of 0 and the label of 1 (compare with G) ⇒ compare with w_b.



Yao's garbled circuit [Yao86]

Alice ↳ **Step 3**

Alice "encrypts" (= garble) each gate $G(w, w') \to w''$ as a table:

T_G = Enc(G) =

Enc(G) =
✗ $Enc_{w_0,w'_0}(0^\lambda w''_{g(0,0)})$
✗ $Enc_{w_0,w'_1}(0^\lambda w''_{g(0,1)})$
✗ $Enc_{w_1,w'_0}(0^\lambda w''_{g(1,0)})$
✓ $Enc_{w_1,w'_1}(0^\lambda w''_{g(1,1)})$

and randomly permutes it. + send to Bob

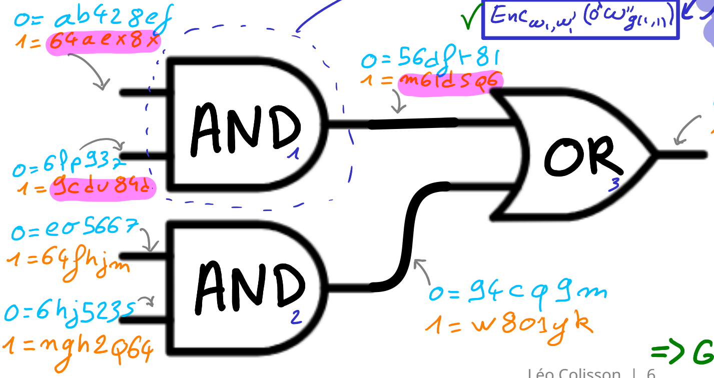
?

If Bob knows G, $\{w_0, w_1\}$ (but not the order) and w_b, can he know b?

⇓

Yes! Idea: decrypt table = allow to know the label of 0 and the label of 1 (compare with G) ⇒ compare with w_b.

⇒ Only reveal w_b, not w_{1-b}!



Yao's garbled circuit [Yao86]

Pb: How can we obtain the input labels

?

Yao's garbled circuit [Yao86]

Pb: How can we obtain the input labels ?

Alice's inputs (b)

Easy: she knows w_0 and w_b

⇒ send w_b to Bob

Yao's garbled circuit [Yao86]

Pb: How can we obtain the input labels ?

Alice's inputs (b)

Easy: she knows w_0 and w_b

⇒ send w_b to Bob

Yao's garbled circuit [Yao86]

Pb: How can we obtain the input labels

Alice's inputs (b)

Easy: she knows w_0 and w_b

=> send w_b to Bob

Bob's inputs (b)

Hard: → Bob knows his bit b

→ Alice knows w_0 and w_1

Goal: • Bob must get w_b

• Alice should NOT learn b.

Yao's garbled circuit [Yao86]

Pb: How can we obtain the input labels

Alice's inputs (b)

Easy: she knows w_0 and w_b

$\Rightarrow$ send w_b to Bob

Bob's inputs (b)

Hard: → Bob knows his bit b

→ Alice knows w_0 and w_1

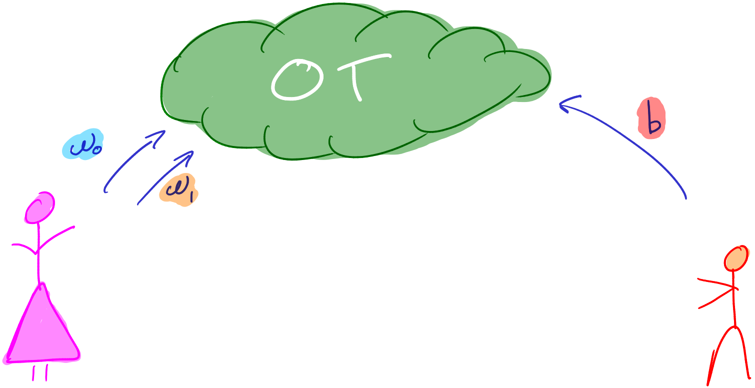
Goal: • Bob must get w_b

• Alice should NOT learn b.

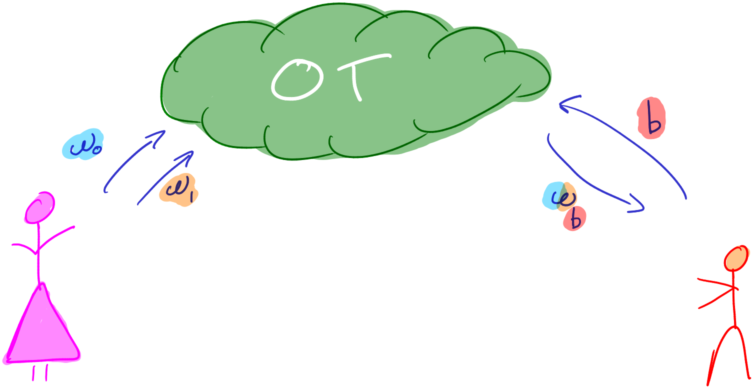
Solution: Oblivious Transfer (OT)

Oblivious Transfer

Oblivious Transfer

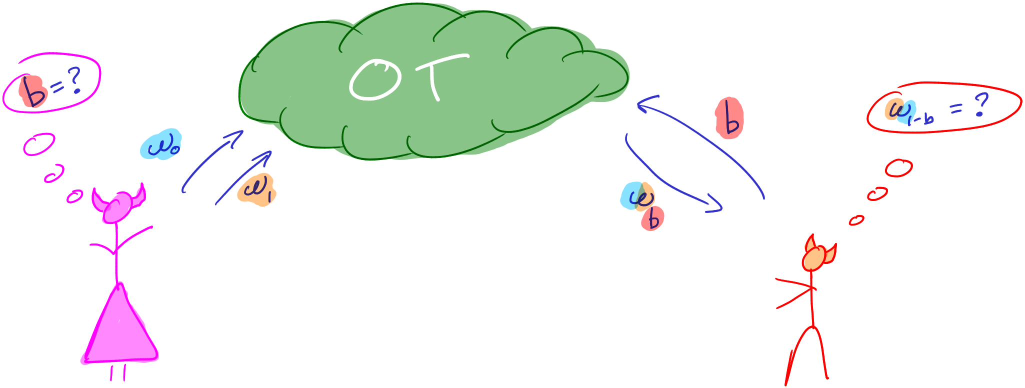


Oblivious Transfer

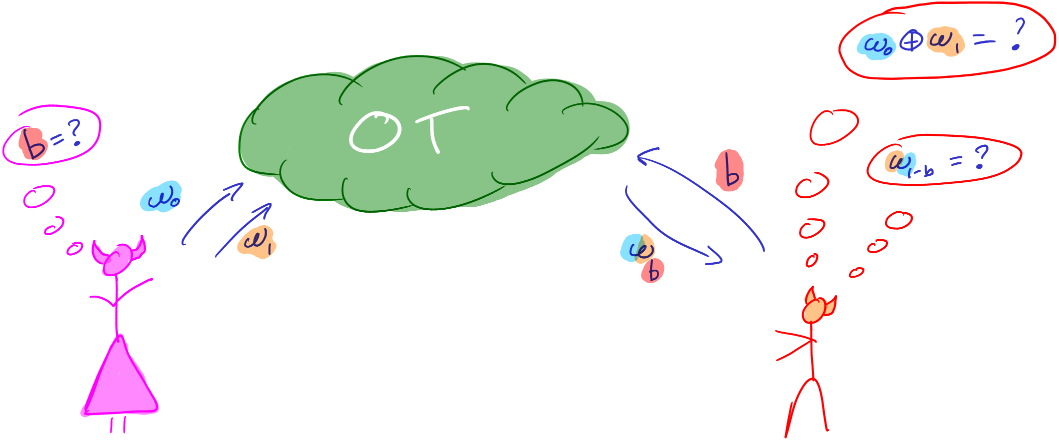


Oblivious Transfer

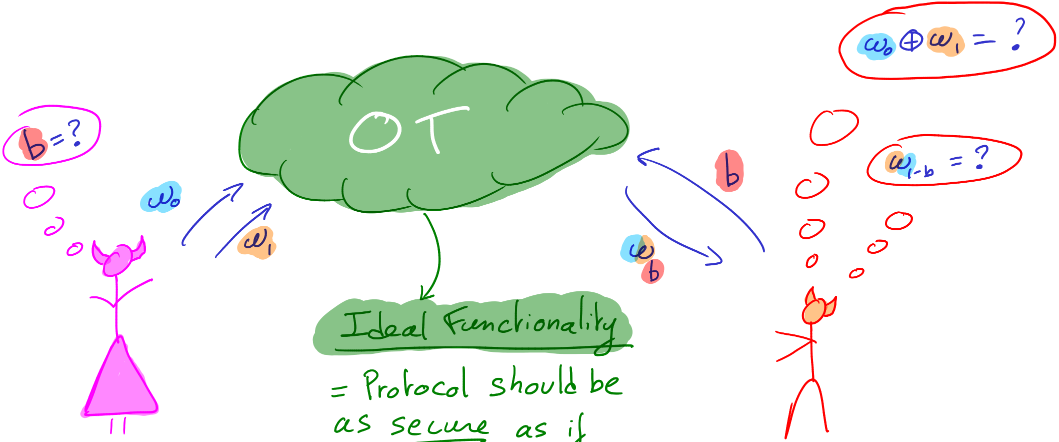
Oblivious Transfer



Oblivious Transfer



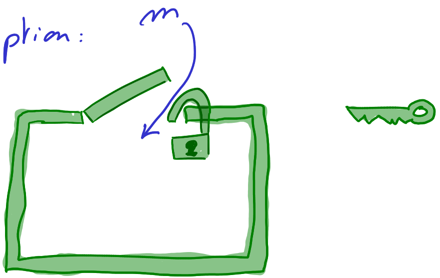
Oblivious Transfer



= Protocol should be as secure as if this were done by a trusted party

Concept 1 : Branch-based encryption

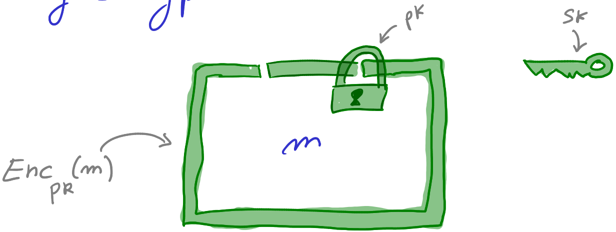
Usual public-key encryption :



Oblivious Transfer: construction [PVW08]

Concept 1 : Branch-based encryption

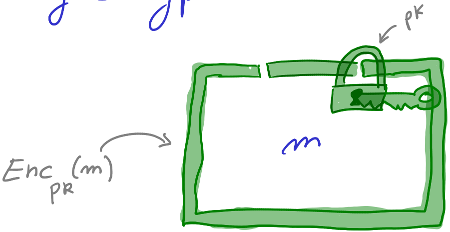
Usual public-key encryption :



Oblivious Transfer: construction [PVW08]

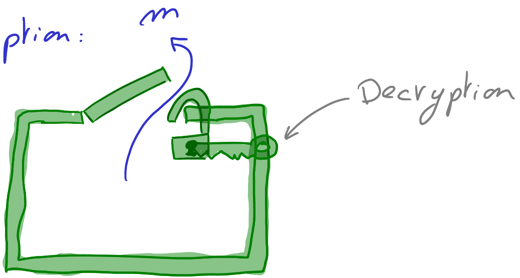
Concept 1 : Branch-based encryption

Usual public-key encryption:



Concept 1 : Branch-based encryption

Usual public-key encryption :

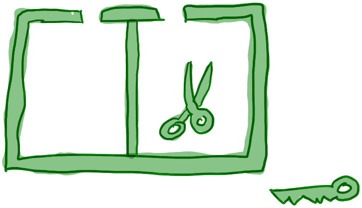


Oblivious Transfer: construction [PVW08]

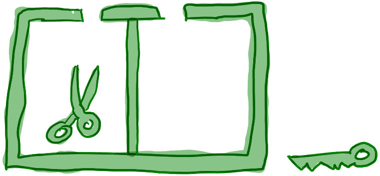
Concept 1: Branch-based encryption

2 kinds of boxes:

Branch 0:



Branch 1:

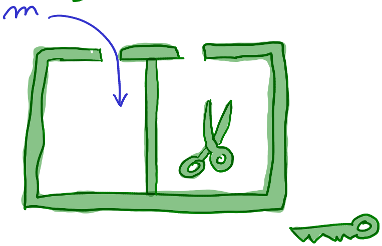


Oblivious Transfer: construction [PVW08]

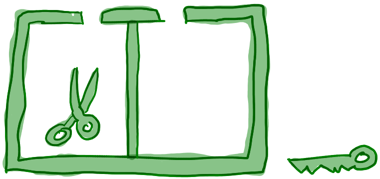
Concept 1: Branch-based encryption

2 kinds of boxes:

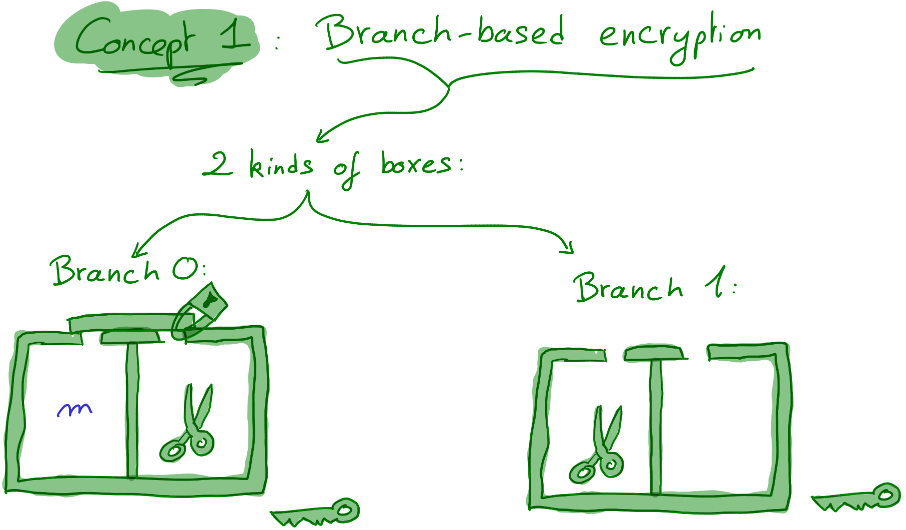
Branch 0:



Branch 1:



Oblivious Transfer: construction [PVW08]



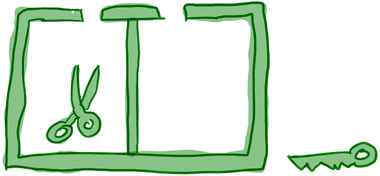
Oblivious Transfer: construction [PVW08]

Concept 1: Branch-based encryption

2 kinds of boxes:

Branch 0:

Branch 1:

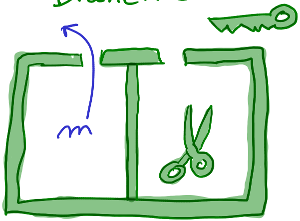


Oblivious Transfer: construction [PVW08]

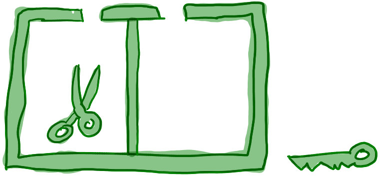
Concept 1 : Branch-based encryption

2 kinds of boxes:

Branch 0:



Branch 1:

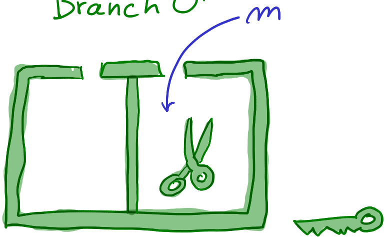


Oblivious Transfer: construction [PVW08]

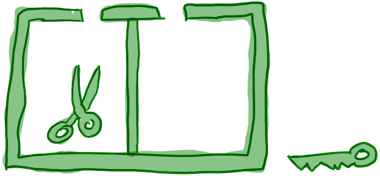
Concept 1 : Branch-based encryption

2 kinds of boxes:

Branch 0:



Branch 1:

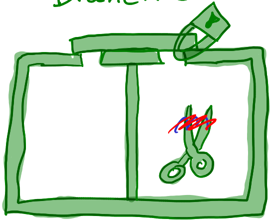


Oblivious Transfer: construction [PVW08]

Concept 1: Branch-based encryption

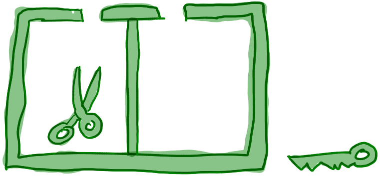
2 kinds of boxes:

Branch 0:



All info about m is lost, even with  !

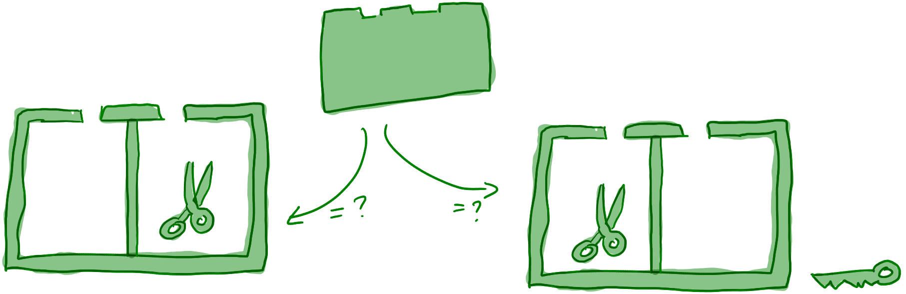
Branch 1:



Oblivious Transfer: construction [PVW08]

Concept 1: Branch-based encryption

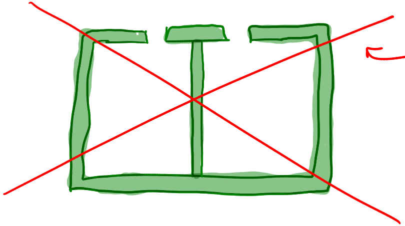
⚠ Given a box (= pk), hard to tell if branch 0 or 1



Oblivious Transfer: construction [PVW08]

Concept 1: Branch-based encryption

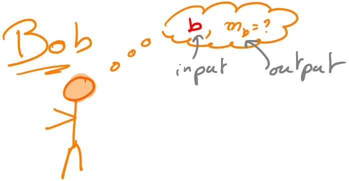
+ Impossible to generate a box "with no scissors":



← Forbidden

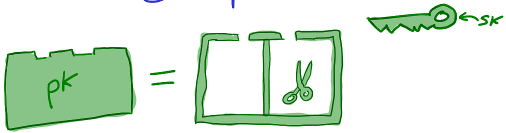
Oblivious Transfer: construction [PVW08]

Oblivious Transfer: construction [PVW08]

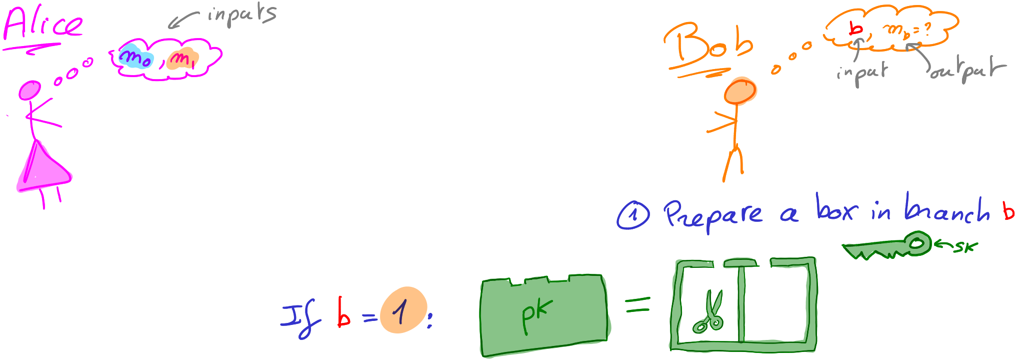


① Prepare a box in branch b

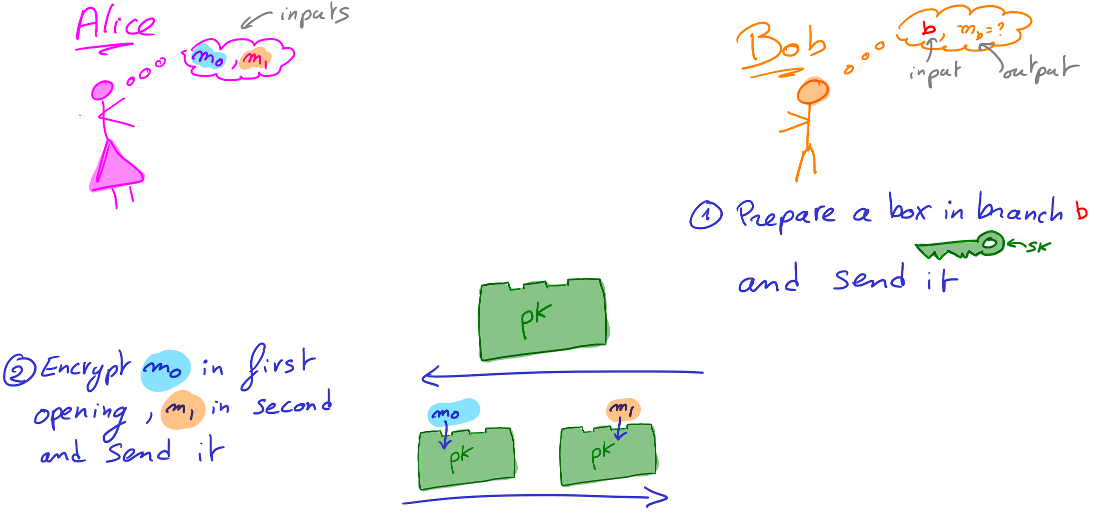
If b = 0:



Oblivious Transfer: construction [PVW08]



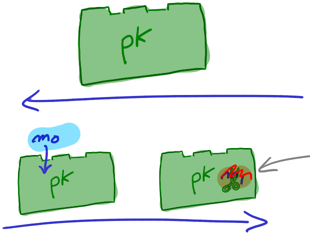
Oblivious Transfer: construction [PVW08]



Oblivious Transfer: construction [PVW08]

① Prepare a box in branch b

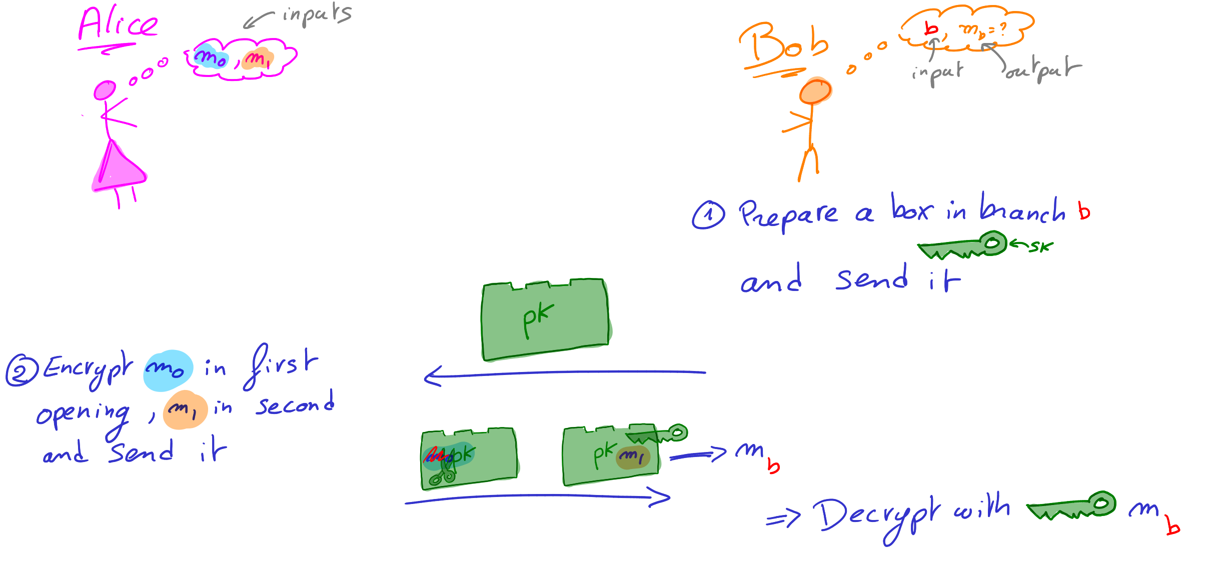
and send it



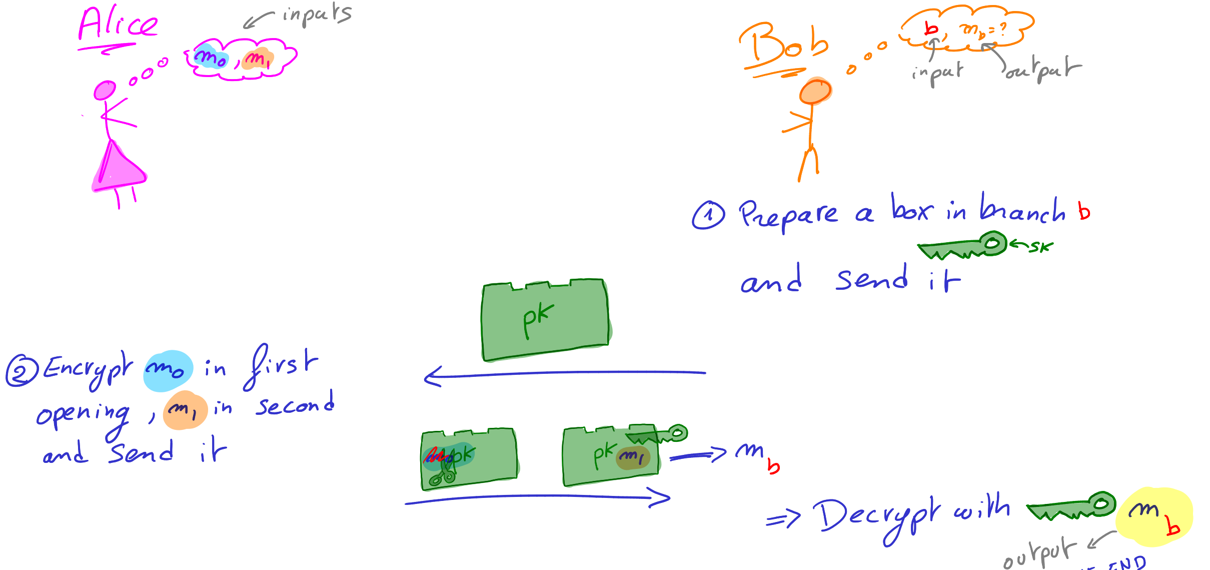
② Encrypt m_0 in first opening, m_1 in second and send it

(If b = 0, m_1 is destroyed

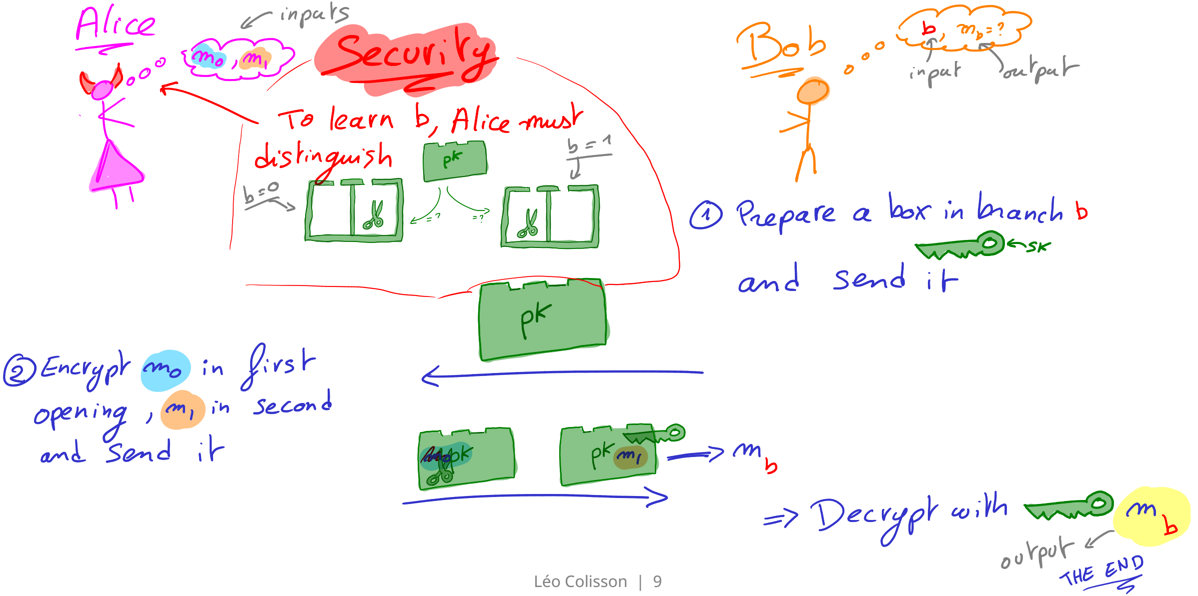
Oblivious Transfer: construction [PVW08]



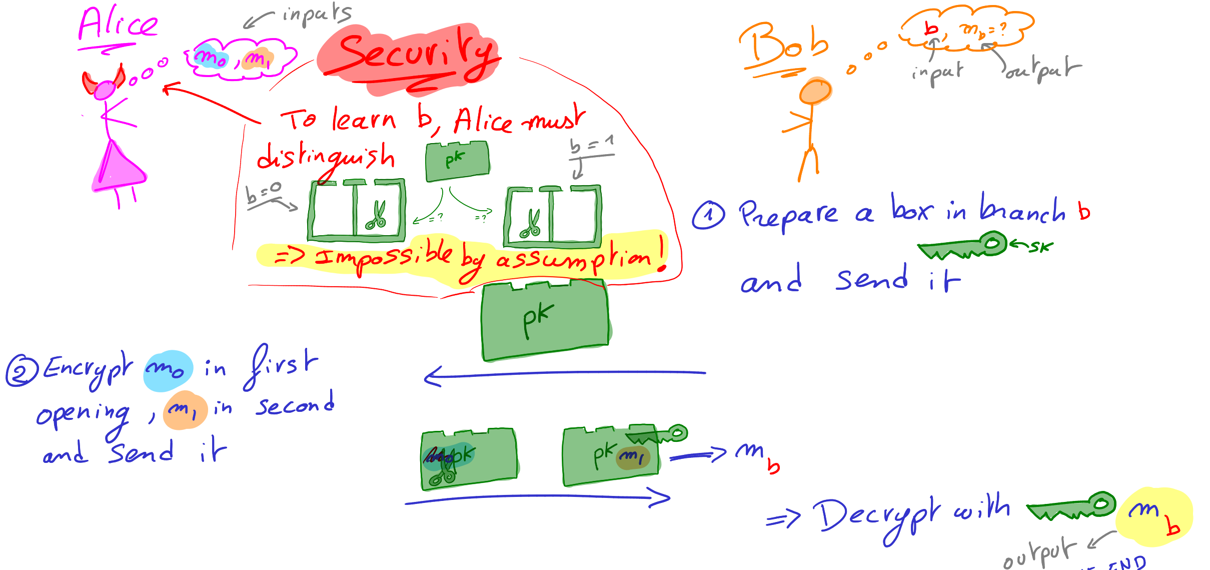
Oblivious Transfer: construction [PVW08]



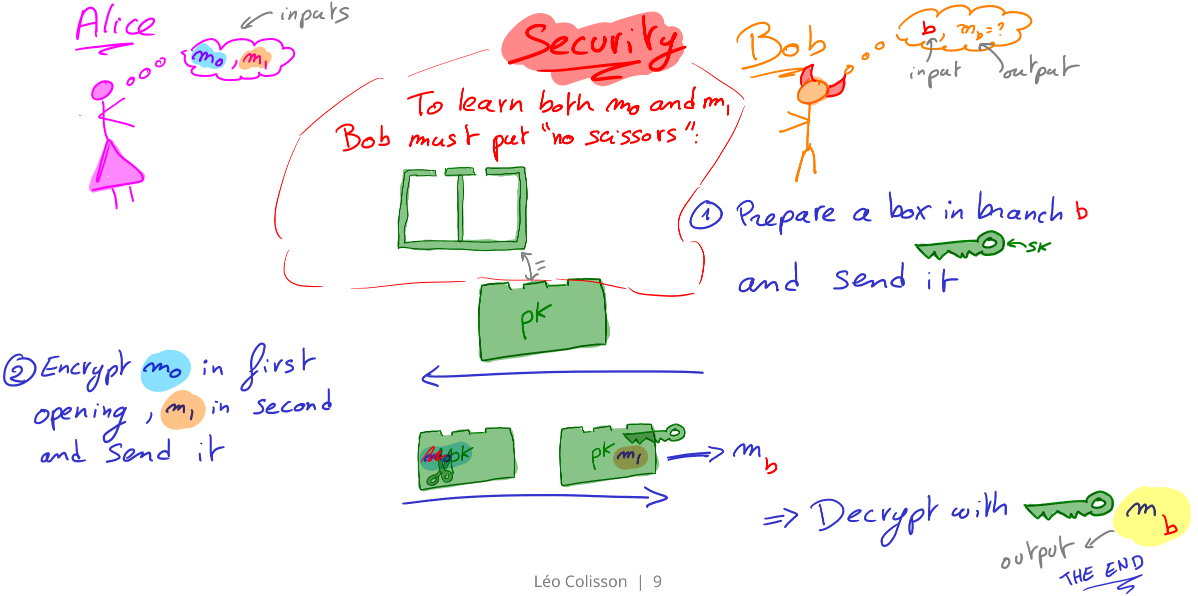
Oblivious Transfer: construction [PVW08]

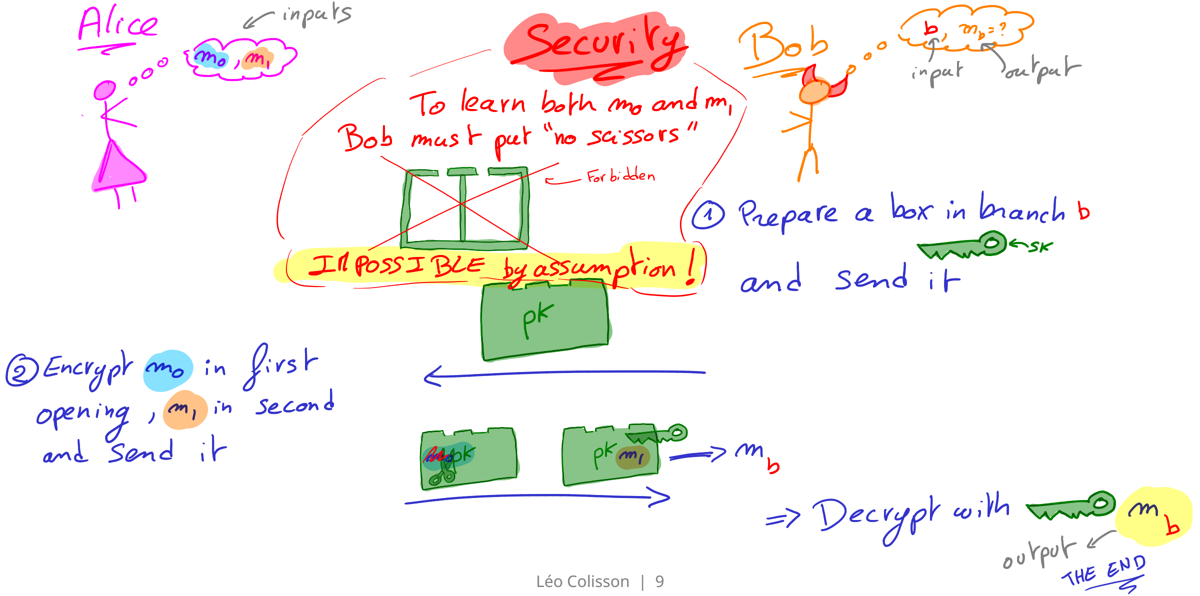


Oblivious Transfer: construction [PVW08]



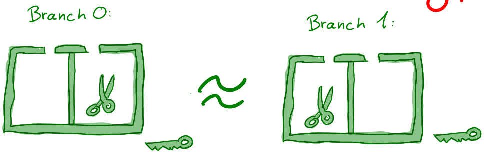
Oblivious Transfer: construction [PVW08]




Alice
inputs
m0 , m1
Security
To learn both m0 and m1
Bob must put "no scissors"
Forbidden
IMPOSSIBLE by assumption!
Bob
b, mb = ?
input
output
① Prepare a box in branch b
sk
and send it
pk
② Encrypt m0 in first
opening, m1 in second
and send it
pk
pk m1
=> mb
=> Decrypt with mb
output
THE END

Oblivious Transfer: construction [PVW08]

How to realize Branch-based encryption ?

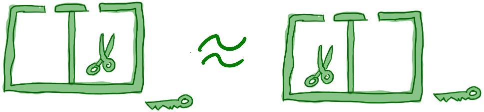


Oblivious Transfer: construction [PVW08]

How to realize

Branch-based encryption ?

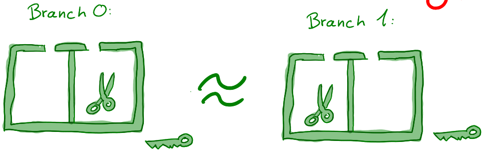
Branch 0: Branch 1:



Many ways → Hardness Discrete Log

?

How to realize Branch-based encryption?



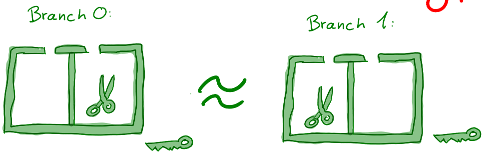
Many ways
- → Hardness Discrete Log
- → Quadratic Residuosity

Oblivious Transfer: construction [PVW08]

?

How to realize

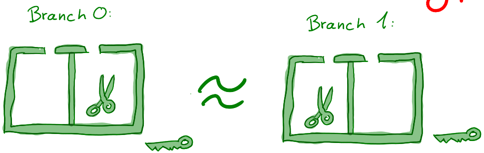
Branch-based encryption ?



Many ways

- → Hardness Discrete Log
- → Quadratic Residuosity
- → Learning With Errors

How to realize Branch-based encryption?



Many ways

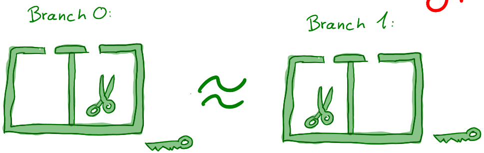
- ↳ Hardness Discrete Log
- ↳ Quadratic Residuosity

} NOT Post-quantum ✗

- ↳ Learning With Errors

Oblivious Transfer: construction [PVW08]

How to realize Branch-based encryption?

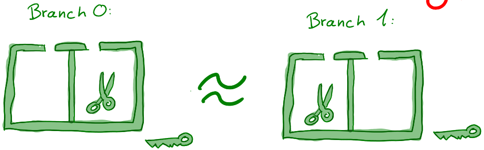


Many ways
- → Hardness Discrete Log
- → Quadratic Residuosity
- } NOT Post-quantum ✗
- → Learning With Errors → Post-quantum ✓

Oblivious Transfer: construction [PVW08]

?

How to realize Branch-based encryption?



Many ways
- ↳ Hardness Discrete Log
- ↳ Quadratic Residuosity

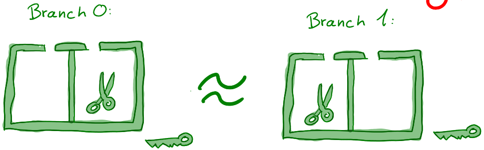
} NOT Post-quantum ✗

- ↳ Learning With Errors → Post-quantum ✓

Oblivious Transfer: construction [PVW08]

?

How to realize Branch-based encryption?



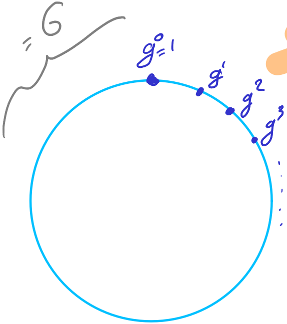
Many ways
- ↳ Hardness Discrete Log
- ↳ Quadratic Residuosity
- ↳ Learning With Errors

} NOT Post-quantum ✗ (Hardness Discrete Log, Quadratic Residuosity)

→ Post-quantum ✓ (Learning With Errors)
⇒ latter in the course

Branch-based encryption from DDH

Branch based encryption from DDH



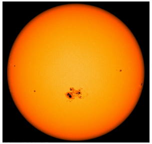
Assume: (Decision Diffie-Hellman, DDH)

- G group order p
- $(g, h, g^a, h^a) \approx (g, h, g^a, h^b)$

Indistinguishable ($\approx$); a, b: random $\in \mathbb{Z}_p$; g, h: random generators

Construction:

- **Setup:** Everyone agrees on random (g_0, h_0, g_1, h_1) — Generator of G; $\in G$



↳ Common Random String (CRS)

⇒ Option 1 : Trusted party (Dual-mode possible)

⇒ Option 2 : Common source of Randomness (e.g. sun spots)

Branch based encryption from DDH

Construction:

- **Setup:** Everyone agrees on random (g_0, h_0, g_1, h_1) — Generator of G; $h_0, h_1 \in G$. Known to simulator: think $h_0 = g_0^{x_0}$, $h_1 = g_1^{x_1}$
- **KeyGen(b)** $= (sk := r,\ pk := (g_b^r, h_b^r))$ — Branch $\in \{0,1\}$; r random $\in \mathbb{Z}_p^*$; $g := g_b^r$, $h := h_b^r$. Intuitively, this picks a random "key", (g_0,h_0) or (g_1,h_1) but hides which key with r
- **Enc**$((g,h), b', m) = (g_{b'}^s h_{b'}^r,\ m \cdot g^s h^r)$ — (g,h) = pk; s, r random; $c_0 = g_{b'}^s h_{b'}^r$, $c_1 = m \cdot g^s h^r$. Unless $g = 1$ (abort)
- **Dec**$(r, (c_0, c_1)) = \frac{c_1}{c_0^r}$ — r = sk
  - $b' = b$: $= \frac{m g^s h^r}{(g_b^s h_b^r)^r} = m \frac{g_b^{rs} h_b^{rs}}{g_b^{rs} h_b^{rs}} = m$
  - $b' \neq b$: Impossible to recover m: (c_0, c_1) is uniformly distributed => [PVW07, Lem 5.1]

Branch based encryption from DDH

Theorem [PVW07]

Assuming the hardness of DDH, the previous construction is a (dual mode) branch-based encryption scheme secure in the CRS model. As a consequence, assuming DDH, there exists an OT protocol secure in the CRS model.

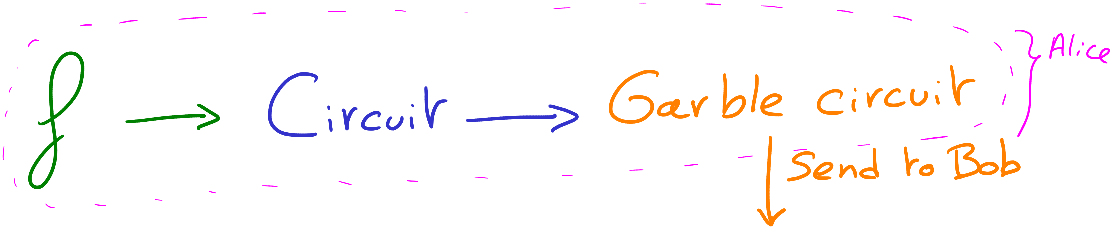
Branch based encryption from DDH

Theorem [PVW07]

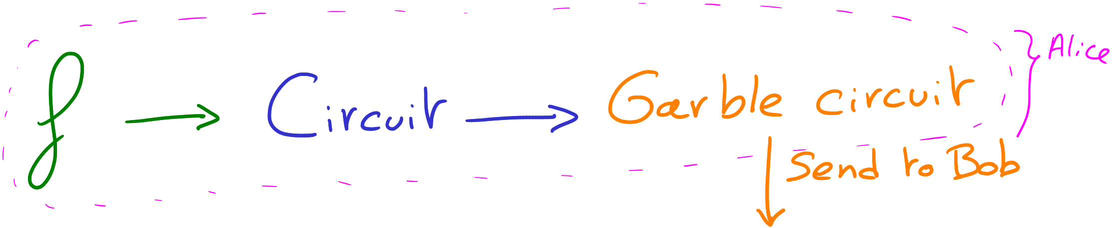
Assuming the hardness of DDH, the previous construction is a (dual mode) branch-based encryption scheme secure in the CRS model. As a consequence, assuming DDH, there exists an OT protocol secure in the CRS model.

Back to MPC

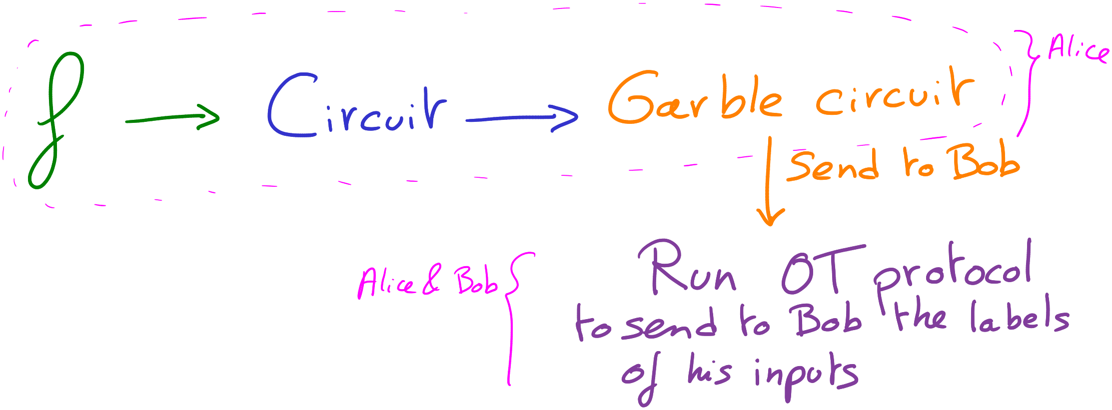
MPC: summary



MPC: summary

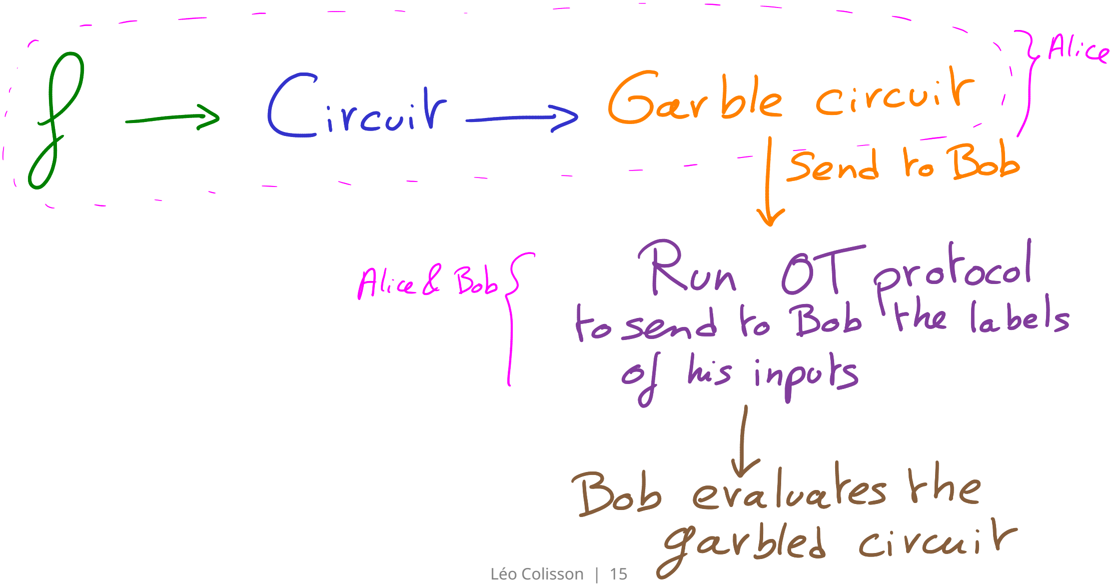


MPC: summary



f
Circuit
Garble circuit
Alice
Send to Bob
Alice & Bob
Run OT protocol to send to Bob the labels of his inputs

MPC: summary


f
Circuit
Garble circuit
Alice
Send to Bob
Alice & Bob
Run OT protocol to send to Bob the labels of his inputs
Bob evaluates the garbled circuit

MPC: summary

f → Circuit → Garble circuit

} Alice

↓ Send to Bob

Run OT protocol to send to Bob the labels of his inputs

} Alice & Bob

↓

Bob evaluates the garbled circuit

←

Bob sends to Alice the labels of her outputs, labels of Bob's output are directly 0/1

MPC: summary

This is secure in an "honest-but-curious" setting...

What can go wrong if Alice is malicious?

MPC: summary

?

This is secure in an "honest-but-curious" setting...

What can go wrong if Alice is malicious?

⟹ Alice can garble a wrong circuit (e.g. identity) to learn Bob's inputs.

MPC: summary

?!

This is secure in an "honest-but-curious" setting...

What can go wrong if Alice is malicious?

⇒ Alice can garble a wrong circuit (e.g. identity) to learn Bob's inputs.

Solution: Add a ZK proof to prove to Bob that the circuit is correctly garbled.

Secret sharing