L14 Multi-party Computation

期末
1月10日
2pm-4:30pm

- Oblivious Transfer

- GMW Protocol

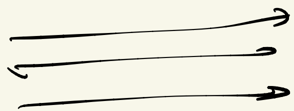
- Garbled Circuits

Oblivious Transfer

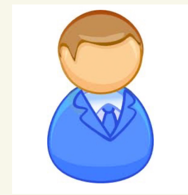
Sender

Alice

input m_0, m_1



Receiver



Bob

input $b \in \{0,1\}$

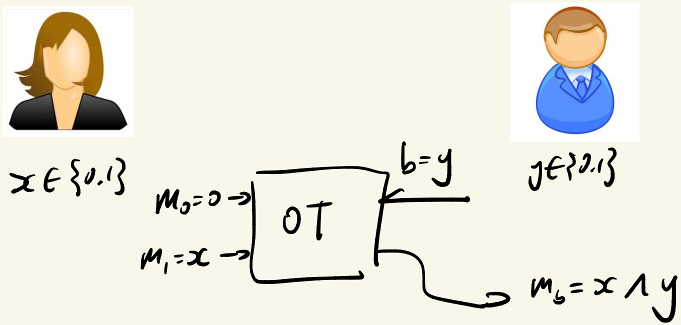
output m_b

1) Correctness
2) Security against sender : Sender learns nothing about b
3) Security against receiver : Receiver learns nothing about m_{1-b}

2PC problem

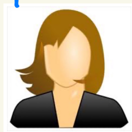
$$f((m_0, m_1), b) = (\perp, m_b)$$

e.g. AND

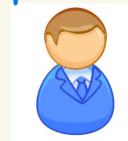


Construction of OT

input m_0 m_1



input b



trapdoor: d

→ trapdoor permutation f

N, e

hard instance Δ

← $s_0, s_1 = \begin{cases} s_b = f(r_b), \\ s_{1-b} = \Delta - s_b \end{cases}$

inputs r_b

inverts r_0 r_1

$r_0 = f^{-1}(s_0)$, $r_1 = f^{-1}(s_1)$

→ $HCB(r_0) \oplus m_0$

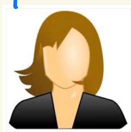
$HCB(r_1) \oplus m_1$ → compute m_b

Sender's view can be statistically simulated.

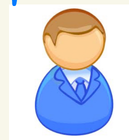
Receiver's view is computationally simulatable

Construction of OT

input m_0 m_1



input b



trapdoor: d

→ trapdoor permutation f

N, e

hard instance Δ

inputs r_b

←
$$s_0, s_1 = \begin{cases} s_b = f(r_b), \\ s_{1-b} = \Delta - s_b \end{cases}$$

inverts

r_0 r_1
" "
$f^{-1}(s_0)$ $f^{-1}(s_1)$

→

$HCB(r_0) \oplus m_0$
$HCB(r_1) \oplus m_1$

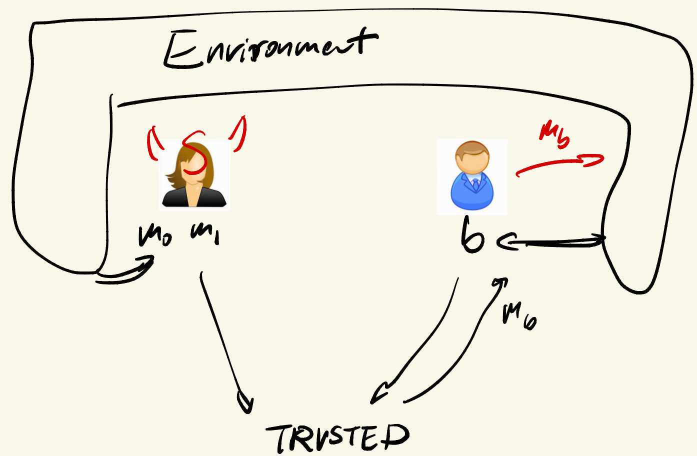
compute m_b

Malicious receiver

want simulator S

Malicious sender

IDEAL WORLD



Construction of OT

input m_0 m_1

input b

GM-encryption

pk: $\mathbb{Z}_{pq}$, $g \in QNR$

sk: (p, q)

Homomorphic evaluation

$\longleftarrow$ pk, $Enc(b)$

$\longrightarrow$

$Enc(b \cdot (m_1 - m_0) + m_0)$ $\xrightarrow{\text{decrypt}}$ m_b

$$Enc(b) = g^b r^2$$

$$Enc(b \cdot (m_1 - m_0) + m_0) = (Enc(b))^{m_1 - m_0} \cdot g^{m_0} \cdot s^2$$

Sender's View

is computationally simulatable

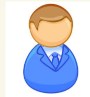
Receiver's View

is statistically simulatable

Construction of OLE

Def of OLE

input a, b



input x

output $ax+b$

Against malicious Sender

$g \leftarrow$ Hard group

$h \leftarrow$ Hard group

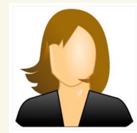
Against malicious Receiver

$h = Enc(1)$

$g \leftarrow$ Hard group

CRS

(N^2, g, h) ← key of Paillier encryption



r, w

s'

s, σ

$M = g^s h^{-\sigma}, \quad M' = g^{s'} h^{-x+\sigma}$

$\leftarrow Enc(x)$

$\longrightarrow$

$g^r \quad h^r (N+1)^a = enc(a)$

$M^r (N+1)^w = enc(w) \quad M'^r (N+1)^{b-w} = enc(b-w)$

$g^{rs} h^{-r\sigma} (N+1)^w \quad g^{rs'} h^{-rx+r\sigma} (N+1)^{b-w}$

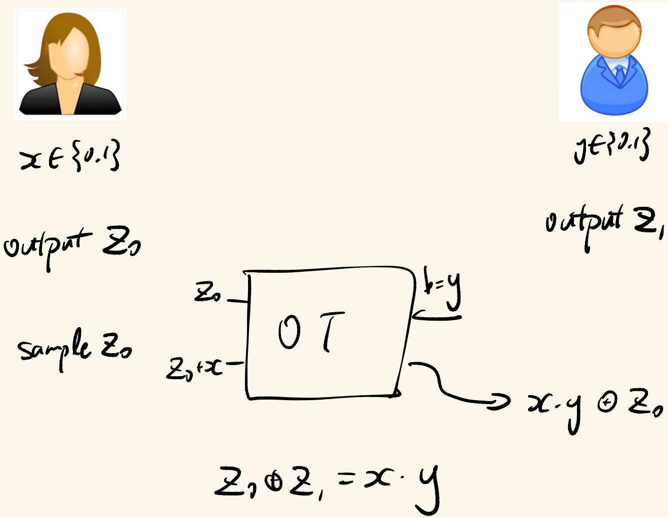
Receiver decode:

$\times \left(g^{-rs} \left(h^r (N+1)^a\right)^{\sigma} \right) \quad \times \left[g^{-rs'} \left(h^r (N+1)^a\right)^{x-\sigma} \right]$

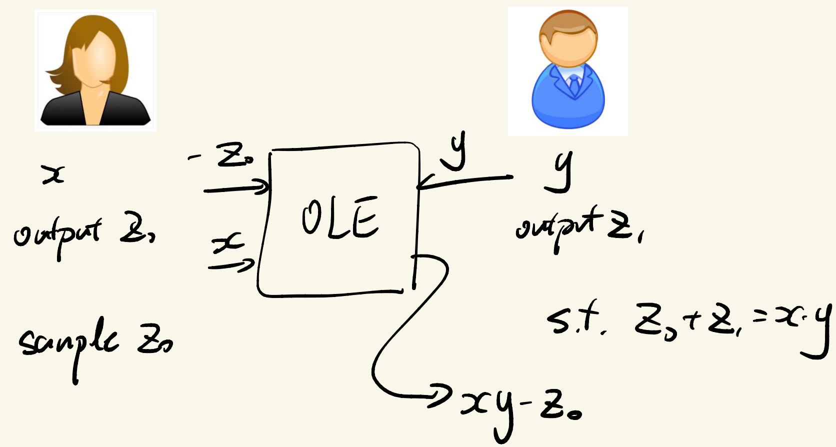
$= (N+1)^{a\sigma + w} \quad = (N+1)^{b-w+ax-a\sigma}$

2PC for "AND"

e.g. AND



e.g. Multiplication

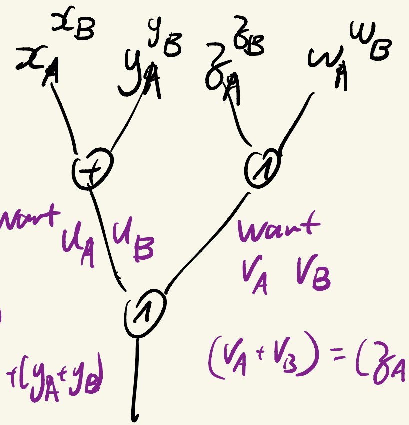


2PC for any function f:

f can be computed by XOR, AND

($\overline{\text{ADD, MULT}}$)

f:



① Additively Share the input

e.g. Alice has x

$x_A + x_B = x$

send x_B to Bob

want u_A u_B s.t.

$$(u_A + u_B) = x + y)$$

$$= (x_A + x_B) + (y_A + y_B)$$

Solution:

let: $u_A = x_A + y_A$

$u_B = x_B + y_B$

want v_A v_B

$$(v_A + v_B) = (z_A + z_B)(w_A + w_B)$$

$$= z_A w_A + z_A w_B + z_B w_B + z_B w_A$$

Solution: Use OT: $z_A w_B = \alpha_A + \alpha_B$

$z_B w_A = \beta_A + \beta_B$

let $v_A = z_A w_A + \alpha_A + \beta_A$

$v_B = z_B w_B + \alpha_B + \beta_B$

OT

⇓

2PC for any function

- Semi-honest security
- #Round = depth + O(1)
- Communication complexity $= \lambda \cdot$ (circuit size)

MPC for any function

Parties $= \{P_1\ P_2 \cdots P_n\}$

Goldreich-Micali-Wigderson GMW protocol

1) Additive share

if P_i has input x

$$x_1 \oplus x_2 \oplus \cdots \oplus x_n = x$$

send x_j to P_j

2) Compute gate-by-gate

2.1) $x = x_1 + \cdots + x_n$ $\quad y = y_1 + \cdots + y_n$

$\oplus$

$$z = z_1 + \cdots + z_n$$

let $z_i = x_i \oplus y_i$

2.2) $x = x_1 + \cdots x_n$ $\quad y = y_1 + \cdots + y_n$

$\otimes$

$$z = z_1 + \cdots + z_n = (x_1 + \cdots + x_n)(y_1 + \cdots y_n)$$

$$= \sum_i x_i y_i + \sum_{i \neq j} x_i y_j$$

Between P_i P_j use OT

$$\underbrace{a_{i,j}}_{\text{to } P_i} + \underbrace{b_{i,j}}_{\text{to } P_j} = x_i y_i$$

let $z_i = x_i y_i + \sum_j a_{i,j} + \sum_j b_{j,i}$

3) ~~3.1) rerandomization~~

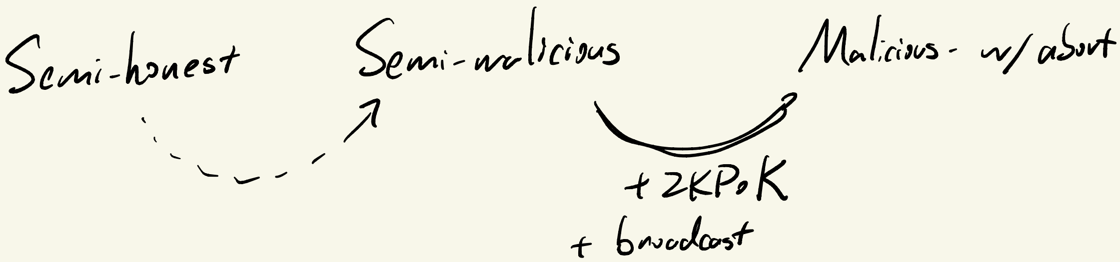
3.2) disclose output

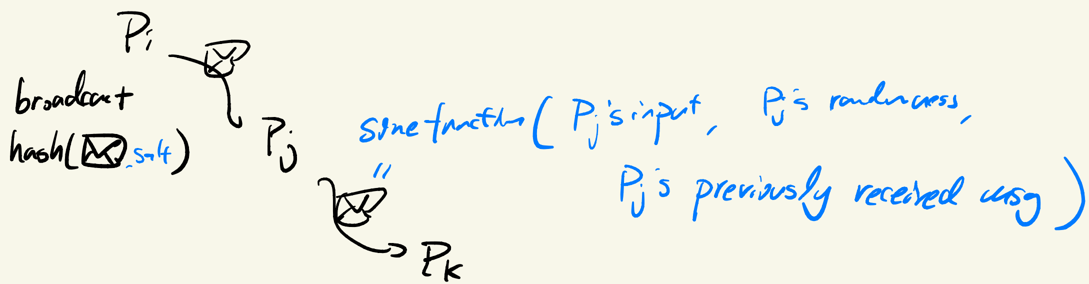
OT

$\Downarrow$

MPC for any function

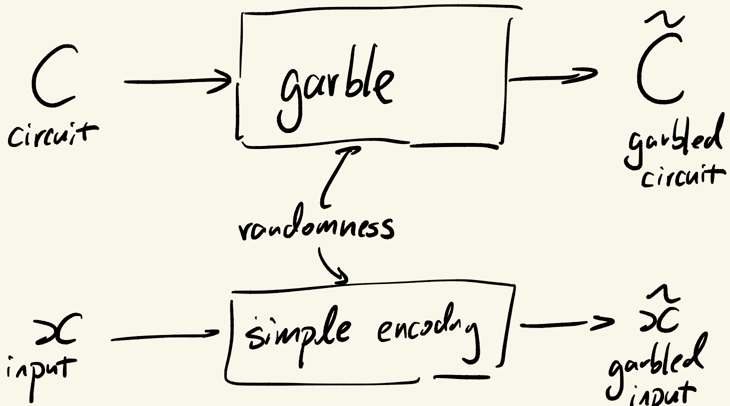
- Semi-honest security
- #Round = depth + $O(1)$
- Communication complexity $= \lambda \cdot (\text{circuit size}) \cdot (\#\text{party})^2$


Semi-honest
Semi-malicious
Malicious - w/ abort
+ ZKPoK
+ broadcast


P_i
broadcast
hash(, salt)
P_j
P_k
same function (P_j's input, P_j's randomness, P_j's previously received msg)

Yao's

Garbled Circuit (GC) ≈ computational randomized encoding



$\tilde{x} = (L_{1,x_1}, L_{2,x_2} \cdots L_{n,x_n})$

0) Simplicity: Labels $L_{1,0}, L_{1,1}, L_{2,0}, L_{2,1}, \cdots, L_{n,0}, L_{n,1}$.

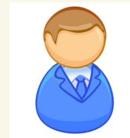
1) Correctness: $\mathrm{Eval}(\tilde{C}, \tilde{x}) \to C(x)$

2) Security: $\exists$ p.p.t Simulator S, $\quad S(C, C(x)) \approx (\tilde{C}, \tilde{x})$

any C, x

GC + OT ⇒ 2PC

$x \in \{0,1\}^n$



$y \in \{0,1\}^n$

output $f(x,y)$

$\text{garble}(f) \to \tilde{f}$

$\tilde{f}$ →

labels:

$L_{1,0}, L_{1,1}$

⋮

$L_{2n,0}, L_{2n,1}$

L_{i,x_i} for $1 \le i \le n$ →

$L_{n+i,0}$ → OT ← y_i

$L_{n+i,1}$ → OT → L_{n+i,y_i}

Correctness

GC's correctness

⇓

$\text{Eval}(\tilde{f}, L_{i,x_i}, L_{n+i,y_i}) = f(x,y)$

Semi-honest Security

GC's security

$S(f, f(x,y))$

↓

$\tilde{f}, L_1, \cdots, L_{2n}$

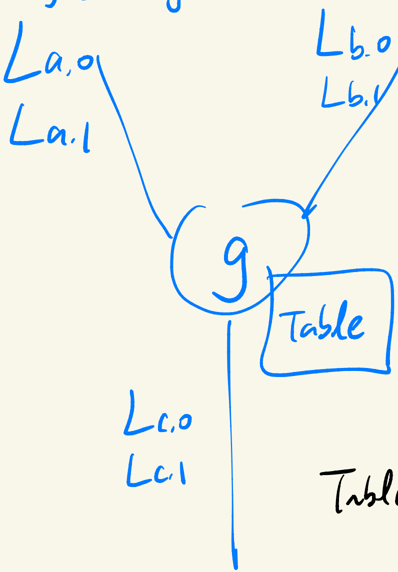
#Round = #Round of OT

communication = ______

GC construction

1) each wire: sample a pair of random labels (λ-bit long)

2) each gate



Given $L_{a,x}$, $L_{b,y}$

can compute $L_{c,g(x,y)}$

and learn nothing of $L_{c,1-g(x,y)}$

① can detect if decryption key is correct

$Enc(L_{a,0}, Enc(L_{b,0}, L_{c,g(0,0)}$ —— $))$
$Enc(L_{a,\alpha}, Enc(L_{b,\beta}, L_{c,g(\alpha,\beta)}$ —— $))$ for $\alpha,\beta \in \{0,1\}$

Table = shuffled

② shuffle

3) each output wire

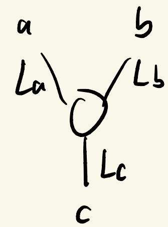
$Enc(L_{d,0}, 0)$
$Enc(L_{d,1}, 1)$

GC security:

How to simulate

1) Each wire:
   sample a random label
   (garbled input)

2) Each gate



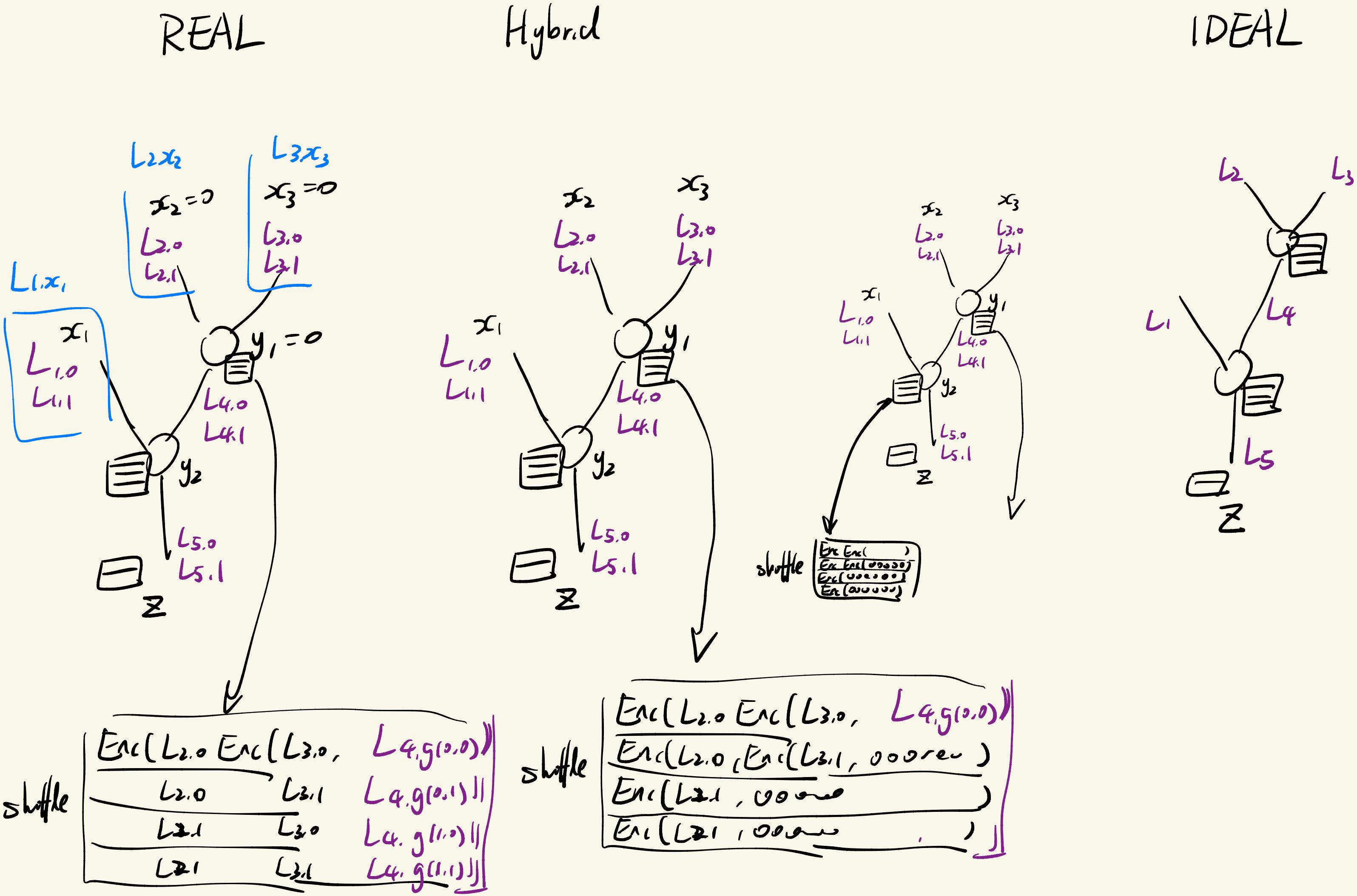
simulated gate table = shuffled

$Enc(L_a, Enc(L_b, L_c))$
$Enc(L_a, Enc(\$, any_1))$
$Enc(\$', any_2)$
$Enc(\$', any_3)$

3) Each output wire: (wire d)
   S knows the value of wire d = V

simulated output wire table =

$Enc(\$, 1-d)$
$Enc(L_d, d)$


REAL
Hybrid
IDEAL
L1,x1
L2x2
L3x3
x2=0
x3=0
y1=0
y2
z
shuffle
Enc(L2.0 Enc(L3.0, L4.g(0,0)))
L2.0 L3.1 L4.g(0,1)
L2.1 L3.0 L4.g(1,0)
L2.1 L3.1 L4.g(1,1)
L1 L2 L3 L4 L5