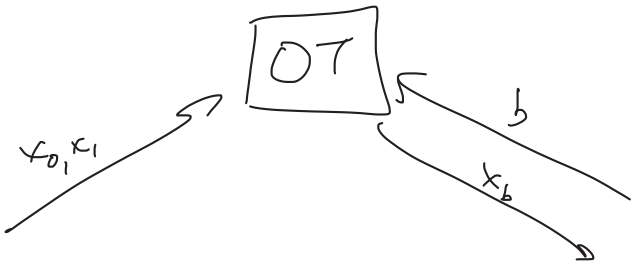
- Piazza link on course webpage
- Google spreadsheet for scribing
- Midterm Mar. 9-23

- Summarize result from last time



Definition

Let (Gen, Enc, Dec) be a public-key encryption scheme. It has *oblivious key sampling* if there are efficient algorithms $SampKey$, $SampRand$ s.t. the following are indistinguishable:

- $\{ pk \leftarrow Gen(1^k); r \leftarrow SampRand(pk) : (r, pk) \}$

and

- $\{ r \leftarrow \{0,1\}^*; pk := SampKey(1^k; r) : (r, pk) \}$.

Protocol

$P_1(x_0, x_1)$ $\qquad$ $P_2(b)$

$(pk_b, sk_b) \leftarrow Gen(1^k)$

$pk_{1-b} \leftarrow SampKey(1^k)$

$\leftarrow \quad pk_0, pk_1$

$c_0 \leftarrow Enc_{pk_0}(x_b)$

$c_1 \leftarrow Enc_{pk_1}(x_{1-b})$ $\qquad \xrightarrow{c_0, c_1}$ $\qquad x_b := Dec_{sk_b}(c_b)$

Theorem

If there is a CPA-secure public-key encryption scheme w/ oblivious key sampling, then the above securely realizes OT in the semi-honest setting.

Proof (corrupted P_2)

Real: $\{ r_b, r_{1-b} \leftarrow \{0,1\}^*;\ pk_b = Gen(1^k; r_b);\ pk_{1-b} = SampKey(1^k; r_{1-b});$
$c_b \leftarrow Enc_{pk_b}(x_b);\ c_{1-b} \leftarrow Enc_{pk_{1-b}}(x_{1-b}) : (r_0, r_1, c_0, c_1) \}$

Simulator

$r_{1-b}, r_b \leftarrow \{0,1\}^*$

$(pk_b, sk_b) \leftarrow Gen(1^k; r_b)$

$(pk_{1-b}, sk_{1-b}) \leftarrow Gen(1^k; r_{1-b})$

$r_{1-b} \leftarrow SampRand(pk_{1-b})$

$c_b \leftarrow Enc_{pk_b}(x_b)$

$c_{1-b} \leftarrow Enc_{pk_{1-b}}(0)$

output (r_0, r_1, c_0, c_1)

Ideal: $\{ r_b \leftarrow \{0,1\}^*;\ pk_b = Gen(1^k; r_b);\ pk_{1-b} \leftarrow Gen(1^k);\ r_{1-b} \leftarrow SampRand(pk_{1-b});$
$c_b \leftarrow Enc_{pk_b}(x_b);\ c_{1-b} \leftarrow Enc_{pk_{1-b}}(0) : (r_0, r_1, c_0, c_1) \}$

Hybrid: $\{ r_b \leftarrow \{0,1\}^*;\ pk_b = Gen(1^k; r_b);\ pk_{1-b} \leftarrow Gen(1^k);\ r_{1-b} \leftarrow SampRand(pk_{1-b});$
$c_b \leftarrow Enc_{pk_b}(x_b);\ c_{1-b} \leftarrow Enc_{pk_{1-b}}(x_{1-b}) : (r_0, r_1, c_0, c_1) \}$

Claim 1 Ideal $\approx$ Hybrid

By CPA-security

Claim 2 Real $\approx$ Hybrid

By oblivious key sampling

$\Rightarrow$ Ideal $\approx$ Real

Proof of Claim 2:

Assume some distinguisher D. Construct D':

$D'(r_{1-b}, pk_{1-b})$

$r_b \leftarrow \{0,1\}^t$, $pk_b \leftarrow Gen(1^k; r_b)$

$c_b \leftarrow Enc_{pk_b}(x_b)$, $c_{1-b} \leftarrow Enc_{pk_{1-b}}(x_{1-b})$

Output $D(r_0, r_1, c_0, c_1)$

$$Pr\left[r_{1-b} \leftarrow \{0,1\}^t ; pk_{1-b} \leftarrow SampKey(1^k; r_{1-b}) : D'(r_{1-b}, pk_{1-b}) = 1 \right]$$
$$= Pr\left[D(Real) = 1 \right]$$

$$Pr\left[pk_{1-b} \leftarrow Gen(1^k) ; r_{1-b} \leftarrow SampRand(pk_{1-b}) : D'(r_{1-b}, pk_{1-b}) = 1 \right]$$
$$= Pr\left[D(Hybrid) = 1 \right]$$

$\Rightarrow$ key obliviousness implies that

$$\left| Pr\left[D(Real) = 1 \right] - Pr\left[D(Hybrid) = 1 \right] \right|$$

is negligible.

Theorem

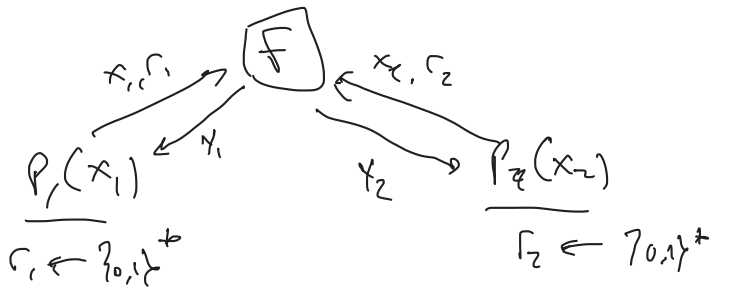
For any n-input functionality F, there is an n-party protocol that $(n-1)$-securely computes F (for semi-honest adversaries), assuming semi-honest OT.

Proof [GMW protocol] Goldreich-Micali-Wigderson '87

- Work in OT-hybrid model
  - protocol is perfectly secure in OT-hybrid model
- Consider deterministic functions
  - can reduce randomized functionalities to this case

Let g be a randomized functionality, $g(x_1, \ldots, x_n ; r)$

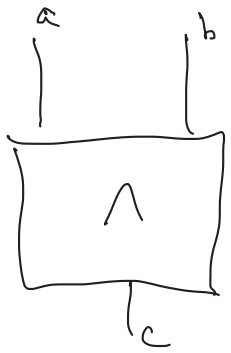
$$f\left((x_1, r_1), (x_2, r_2), \ldots, (x_n, r_n)\right) \stackrel{\text{def}}{=} g(x_1, \ldots, x_n ; r_1 \oplus r_2 \oplus \ldots \oplus r_n)$$



- Look at 2-party case first, then extend to n-party case
- Represent f as a boolean circuit

GMW Protocol (OT-hybrid world)

1) Input-sharing phase: Setup the invariant that the parties jointly hold a secret share of the value of the wires in the circuit



P_1	P_2	
a_1	a_2	$a_1 \oplus a_2 = a$
b_1	b_2	$b_1 \oplus b_2 = b$
c_1	c_2	$c_1 \oplus c_2 = c$

P_1 has input a on 1st wire

$a_1 \leftarrow \{0,1\}$

$a_2 = a \oplus a_1$

P_2 has input b on 2nd wire

$b_2 \leftarrow \{0,1\}$

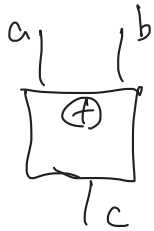
$b_1 = b \oplus b_2$

$a_2 \rightarrow$

$\leftarrow b_1$

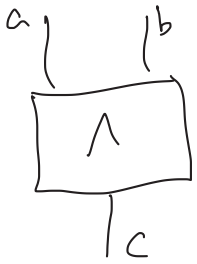
2) Computing gates of the circuit

P_1	P_2	
a_1	a_2	$a_1 \oplus a_2 = a$
$b_1 = a_1 \oplus 1$	$b_2 = a_2$	$b_1 \oplus b_2 = a \oplus 1 = \neg a$



a_1	a_2	$a_1 \oplus a_2 = a$
b_1	b_2	$b_1 \oplus b_2 = b$
$c_1 = a_1 \oplus b_1$	$c_2 = a_2 \oplus b_2$	$c_1 \oplus c_2 = a \oplus b = c$

a_1
b_1



a_2
b_2

$$c_1 \oplus c_2 = (a_1 \oplus a_2) \wedge (b_1 \oplus b_2)$$

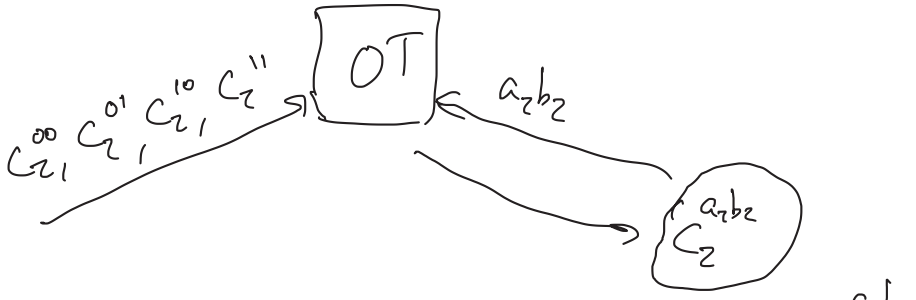
$c_1 \leftarrow \{0,1\}$

if $a_2 = 0, b_2 = 0$,
then $c_2^{00} = c_1 \oplus (a_1 \wedge b_1)$

if $a_2 = 0, b_2 = 1$,
then $c_2^{01} = c_1 \oplus (a_1 \wedge \bar{b}_1)$

⋮

c_2^{10}, c_2^{11}



c_1

$c_2 = c_2^{a_2 b_2}$

3) Say z is a value on an output wire that P_2 should learn

z_1 ⟶ z_2

$z = z_1 \oplus z_2$