- Scribes?
- lecture recording

- Coin tossing & GMW I compiler (2-parties)
- Defining malicious security in the multi-party case
- GMW I Compiler (multi-party)

$\underline{S}(m_0, m_1)$ $\qquad\qquad$ $\underline{R}(b)$

$\xleftarrow{pk_0,\ pk_1}$ $\qquad$ $(pk_b, sk_b) \leftarrow Gen$

$\qquad\qquad\qquad\qquad pk_{1-b}$

$\xrightarrow{Enc_{pk_0}(m_0),\ Enc_{pk_1}(m_1)}$

GMW I Compiler

- Compiles any protocol w/ semi-honest security into a protocol w/ malicious security
- notion of malicious security is security with abort

Main idea:

parties run the semi-honest protocol; after each step, party gives a ZK proof of correct behavior semi-honest protocol Π

- need to ensure parties use "good" randomness
- need to ensure that parties use the same input/randomness throughout

$P_1(x)$ $\qquad\qquad$ $P_2(y)$

$msg_1 = \pi_1(x, r_1)$ $\quad \xrightarrow{msg_1}$

ZK $\Rightarrow$

$\exists x, r : msg_1 = \pi_1(x, r)$

$\xleftarrow{msg_2}$ $\quad msg_2 = \pi_2(y, r_2, msg_1)$

$\Leftarrow$ ZK $\exists y, r' : msg_2 = \pi_2(y, r', msg_1)$

$\xrightarrow{msg_3}$

ZK: $\exists x, r_3 : msg_3 = \pi_3(x, r_3, msg_2)$ $\Rightarrow$

Coin-tossing protocol

- One party learns a uniform value; the other party gets a commitment to that value

P_1 $\qquad\qquad$ P_2

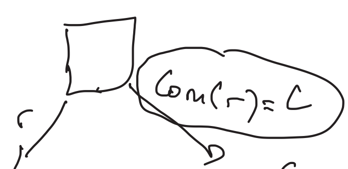
$r_1 \leftarrow \{0,1\}^n$ $\quad \xrightarrow{Com(r_1)}$

ZKPoK : r_1 $\Rightarrow$

$\xleftarrow{r_2}$ $\quad r_2 \leftarrow \{0,1\}^n$

$r = r_1 \oplus r_2$ $\quad C = Com(r) \longrightarrow$

ZKPoK : C is a commitment to $r_2 \oplus r_1$, where r_1 was the original committed value $\Rightarrow$



Sim $\qquad P_2^*$

$\xrightarrow{Com(0)}$

$\Rightarrow$

$\xleftarrow{r_2}$

$\xrightarrow{C}$

$\Rightarrow$

P_1^* Sim

r

$\xrightarrow{C_1}$

ZKPoK : r_1 extract r_1

$\xleftarrow{r_2 = r_1 \oplus r}$

$\xrightarrow{C'}$

ZKPoK

① r ② $\rho / \bot$ ③ $C = Com(r;\rho)$

GMW I Compiler

$P_1(x)$ $P_2(y)$

$\xrightarrow{C_x = Com(x)}$

ZKPoK : x

$\xleftarrow{C_y = Com(y)}$

ZKPoK : y

2x Coin tossing

r_1, C_1 $C_1 = Com(r_1)$

$C_2 = Com(r_2)$ r_2, C_2

$\xrightarrow{msg_1}$

$msg_1 = \pi_1(x, r_1)$

ZK : $\exists x, r_1 : msg_1 = \pi_1(x, r_1)$
and x, r_1 are consistent
w/ original commitments

⋮

P_1^* Sim $\xrightarrow{x}$ F $\xleftarrow{y}$ $\xleftarrow{z}$ continue $\xrightarrow{z}$

$\xrightarrow{C_x}$

ZKPoK   extract

$\xleftarrow{Com(0)}$ run $Sim_\pi(x, z)$

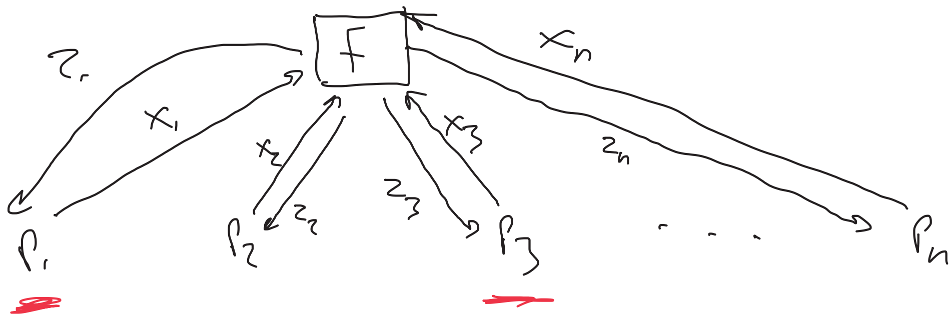
$\xleftarrow{zk}$ to gen

r r_1 transcript

$\xrightarrow{msg_1}$

$\xleftarrow{msg_2}$

⋮



$\boxed{\text{Security-with-unanimous-abort}}$ — achievable for $t<n$ given broadcast

- Adversarial parties learn their outputs, then abort or continue
- if continue, then honest parties get output
- if abort, honest parties get $\perp$

- unanimity of abort?
- Fairness – either no one gets output or everyone does
- guaranteed output delivery

$\boxed{\text{Security-without-abort (i.e., Full security)}}$
- achievable for $t < n/2$ given broadcast
- not achievable for $t \geq n/2$ (in general), even given broadcast

- all parties send input to ideal functionality
- all parties get output

GMW I compiler in the multi-party case
Compiles semi-honest protocol Π into a protocol than is secure-with-unanimous-abort

- every party commits to its input & gives a ZK PoK of its input (over broadcast channel)
- run a multi-party version of coin tossing
- run the semi-honest protocol + ZK proof of consistency at every step