- Scribes?
- Lecture recording
- Coin tossing & GMW I compiler (2-parties)
- Defining malicious security in the multi-party case
- GMW I compiler (multi-party)

\[
\begin{align*}
S(m_0, m_1) & \quad R(b) \\
& \quad p_{k_0, p_k} \quad C(p_{k_1, s_{k_1}}) \leftarrow \text{Gen} \\
& \quad \text{Enc}_{p_{k_1, b}}(m_0), \text{Enc}_{p_{k_1, m}}(m_1)
\end{align*}
\]

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, each party gives a 2K proof of correct behavior.
Semihonest protocol $\mathcal{P}$

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

Commit-tossing protocol

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

$C = \text{Commit}(c)$

$C$ is a commitment to $c_2 \oplus c_1$, where $c_1$ was the original committed value
\[ C_i = \text{Com}(r_i) \]

\[ C_x = \text{Com}(x) \]

\[ 2k \mathsf{PR}: x \]

\[ C_i = \text{Com}(y) \]

\[ 2k \mathsf{PR}: y \]

\[ 2x \mathsf{Coin Tossing} \]

\[ C_x \]

\[ 2k \mathsf{PR} \]

\[ \text{msg}_1 = \Sigma_r(x, r_i) \]

\[ 2k: \exists x, r_i: \text{msg}_1 = \Sigma_r(x, r_i) \]

and \( x, r_i \) are consistent with original commitments.
Security - with-unanimous-abort — achievable for $T < n$ given broadcast
- Adversarial parties learn their output; then abort or continue
- If continue, then honest parties get output
- If abort, honest parties get $\bot$

- Unanimity of abort?
- Fairness — either no one gets output or everyone does
- Guaranteed output delivery

Security - with-abort (i.e., Full security) — achievable for $T < \frac{n}{2}$ given broadcast
- Not achievable for $T \geq \frac{n}{2}$ (in general), even given broadcast

GMW 1 Compiler in the multi-party case
Compiles semi-honest protocol $T_1$ into a protocol
- that is secure - with-unanimous - abort
- Every party commits to its input and gives a ZK proof of its input (our broadcast channel)
- Run a multi-party version of coin tossing
- Run the semi-honest protocol + ZK proof of consistency at every step