Lecture 23

* Wrap up
* ZK proofs
* Secure 2-party computation

Recall (from last lecture) BLUM's protocol for HC

\[ \text{Graph } G = (V, E) \]

HC \( H \in G \)

1. \( M = \) permuted adj. matrix
   
   for \( G \)

   \( C_{ij} = \text{Commit}(\{M_{ij}\}) \)

   \( b \in \{0, 1\} \)

   \( b \)

2. if \( b = 0 \),

response = show how \( M, c \) were computed

else, response = show entries for \( H \)

Properties:
* Completeness: if everyone follows protocol, verifier accepts ✓
* Soundness: if prover cheats, then verifier catches prover cheating with probability \( \frac{1}{2} \)

Variant 1

\( ZK: \exists s.t. \forall \text{ p.p.t. } V^*, \forall x \in L \)

\[ \{ (P(x) \leftrightarrow V^*(|k, x|)) \rightarrow \text{view: view} \} \]

\& \( \{ \text{S (x, desc. of } V^*) : \text{view} \} \)

Variant 2

Black-box ZK, same as above but S doesn't get desc. of \( V^* \), only gets to use \( V^* \) as a black-box

Variant 3

\( \forall \text{ p.p.t. } V^* \exists s.t. \forall x \)
ZK simulator:

1. Guess $b'$
2. If $b' = 0$, compute $C_{a:j}$ "honestly" 
   Else, compute $C_{a:j}$ as if graph were $H$ (some cycle graph)
3. Run $V^*(G)$, send $C_{a:j}$ to $V^*$, get $b$.
4. If $bb' = b$, go to 1
   Else, open $C_{a:j}$;
      if $b = 0$, open honestly
      if $b = 1$, open $C_{a:j}$ that corresponds to $H$
5. Continue running $V^*$, giving it these openings
   Output whatever $V^*$ outputs.

... hybrid argument

§ Boosting: Soundness

New protocol: Run Blum protocol $k$ times.
Verifier accepts if accepts every time. Else, reject.

* completeness ✓ with overwhelming prob.
* soundness $1 - 2^{-k}$ w.e.p. verifier will catch a cheating prover
* ZK simulate incrementally, when rewinding, repond to last successful iteration

Aside: Composition Thm: $\exists S \forall V^* \forall x \forall$ advice string $s$
TM to non-uniform $\uparrow$
state of last iteration
What if you run k times in parallel?

completeness /

soundness /

ZK :: S needs to guess \{ b \} for b \in \{0, 1\}

THE FIX: Add extra step so we can rewind & know what the verifier is going to choose for b's

... but in trouble with soundness b/c prover is supposed to be UNBOUNDED
\therefore, commit needs to be UNCOND. HIDING.

\textbf{REMARK:} This gets us ZK for all of NP. yay!

(but some research problems remain.)

\section{Oblivious Transfer}

\begin{verbatim}
\begin{center}
\begin{tikzpicture}
\node [draw, rounded corners] (Sender) {Sender \((x_0, x_1)\)}; \node [draw, rounded corners] (Receiver) at (4,0) {Receiver \(b \in \{0, 1\}\)}; \node [draw, rounded corners] (Security) at (8,0) {Security: \(-\text{sender learns nothing about } b\)}; \node [draw, rounded corners] (Security2) at (8,0) {\(-\text{receiver learns nothing about } m_b\)}; \node [draw, rounded corners] (Protocol) at (2,0) {Protocol}; \node [draw, rounded corners] (X_b) at (4,-1) {X_b};
\draw [->] (Sender) -- (Protocol); \draw [->] (Protocol) -- (Receiver); \draw [->] (Receiver) -- (Security); \draw [->] (Sender) -- (X_b);
\end{tikzpicture}
\end{center}
\end{verbatim}

\begin{enumerate}
\item Why is this important? \textbf{A:} Useful in secure 2-party comp.
\end{enumerate}
§ TRY 1 (with honest-but-curious adversary)

\[ \text{follows protocol but then tries to learn something it wasn't supposed to} \]

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ (pk_a, sk_b) \leftarrow \text{Key Gen for Pk Enc} \]

\[ pk_b \leftarrow \text{sampling algo that does not let } sk_b \text{ be known e.g. El Gamal} \]

\[ C_2 = \text{Enc}(pk_a, x_1) \]

\[ C_1 = \text{Enc}(pk_b, x_1) \]

\[ C_0, C_1 \rightarrow \text{Decrypt } C_b \]

**SECURITY:** For adversarial sender, doesn't learn b.

For honest-but-curious receiver, doesn't learn x_1

\[ \sqrt{\text{but what about malicious receiver?}} \]

then problem b/a \[ (pk_b, sk_b) \leftarrow \text{Key Gen} \]

§ TRY 2 (with malicious receiver)

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{secure coin flipping} \]

\[ pko, pk_i, \text{zk proof that they were generated according to algo, } \]

\[ & \text{& using random bits agreed on.} \]

\[ \text{Encrypt } Enc(pko, x_0), Enc(pk_i, x_i) \]

\[ \text{decrypt} \]
How is secure coin flipping accomplished?

Sender

Receiver

commit \( \text{random PRF seed} \)

\[ \text{random } r \]

use seed = seed' \oplus r

... but malicious sender can encrypt maliciously

so add another zk proof