Today we enter third “unit” of the class
- Unit 1 (Lectures 1-4): Foundations
- Unit 2 (Lectures 5-8): Cryptanalysis & ECC
- Unit 3 (Lectures 9-13): Zero-Knowledge

Outline:
- Proofs
- Interactive Proofs
- Zero Knowledge
- ZKP for Hamiltonian Cycles
Q: What is a proof?
   → It demonstrates truth ... or
   prover → SOMEONE convinces
   verifier → SOMEONE ELSE that
   statement → SOMETHING is true

A convention: statements
   as membership

   "x is prime"

   x ∈ L primes ← set of primes
   instance → language
   statement

Formally (from complexity theory):

A language is a set of strings L ⊆ Σ*,
A statement takes the form x ∈ L

Examples:
- "15 is biprime" → 15 ∈ \{pq : p, q ∈ L primes\}
- "\phi is satisfiable" → \phi ∈ \{formula \phi \mid \exists x \phi(x) = 1\}
- "\phi is unsatisfiable" → \phi ∈ \{formula \phi \mid \forall x \phi(x) = 0\}
- "G has a Hamiltonian cycle" → G ∈ \{all graphs with Ham. cycle\}
- "\phi_{PRTTAG} is true" → ...

Not all statements are equally hard to prove!
Non-Interactive Proofs

Classically, a proof is simply read (non-interactive)

Fix a language $L$. A prover $P$ wants to create a proof $\pi$ to convince a verifier that $x \in L$

$\text{unbounded} \rightarrow P(x) \quad \pi \quad V(x) \leftarrow \text{polytime}$

Verify $(x, \pi) \rightarrow \{0, 1\}$

Key Properties:
- Completeness: $\forall x \in L, \text{Verify}(x, \pi) = 1$
- Soundness: $\forall x \notin L, \text{Verify}(x, \pi) = 0$

If we have completeness and soundness $\Rightarrow L \in \text{NP}$

($\pi$ is an NP witness)

NP Complexity Class:
$\rightarrow$ Informally, a language $L \in \text{NP}$ if statement $x \in L$
can be proven with a non-interactive proof

$\rightarrow$ Formally, there exists an efficient algorithm $M(\cdot, \cdot)$
$s.t. \quad x \in L \iff \exists w \in \{0, 1\}^{\text{poly}(n)} s.t. \quad M(x, w) = 1$

Ex: To prove $\phi \in \text{SAT}$, $P$ sends satisfying assignment to $V$

More generally, $P$ sends $w$ and $V$ checks $M(x, w) = 1$

How can we prove more?

By changing the model!

$\Rightarrow$ in a court, people ask questions!
Interactive Proofs \hspace{1cm} [Goldwasser, Micali]

- $P(x) \leftrightarrow V(x)$
- (access to randomness)
- $0/1$

- $P, V$ are randomized, interactive Turing machines
  $\leftrightarrow$ # rounds, message length, $V$ time : poly
  $\leftrightarrow P$: unbounded

Key Properties:
- Completeness: $\forall x \in L, \Pr[\langle P, V \rangle(x) = 1] \geq \frac{2}{3}$
- Soundness: $\forall x \notin L, \Pr[\langle P, V \rangle(x) = 1] \leq \frac{1}{3}$

- $P$ is honest prover, and soundness should hold for any malicious prover, not just the honest one

- Denotes output of $V$ when $P, V$ interact

What do we get from interaction?
1. IP captures much broader class of problems than NP. In fact, $IP = PSPACE$!
2. Even for NP statements, interaction can allow proving a statement with communication $< |W|$.
3. Interaction enables a surprising new property: ZERO-KNOWLEDGE...
Zero Knowledge

Conceptually, a proof that shows $x \in L$ and reveals nothing else.

Examples:
- Given $\phi$, prove that $\phi \in \text{SAT}$ without revealing the satisfying assignment.
- Prove $x$ is the correct output of some algorithm without revealing my secret inputs to the algorithm.

How do we define "reveals nothing else"? How do we define knowledge?
- Say you have $N = pq$ and also the factor $p$.
  Do you know $q$? Yes b/c you can calculate $q$ efficiently!
- Say you have an encryption of $x$, $\text{Enc}_{pk}(x)$.
  Do you know $x$? Intuitively no b/c you can't efficiently recover $x$ from $\text{Enc}_{pk}(x)$.

$\Rightarrow$ KNOWLEDGE is what you can compute efficiently.

Intuition: if any info a dishonest verifier can derive from the protocol transcript could have been efficiently derived from $x$, the protocol is zero-knowledge!

Zero Knowledge: $(P,V)$ is ZK if $\forall$ PPT $V^*$, $\exists$ PPT Sim, $\forall x \in L$, $\{\text{View}_{V^*}[<P,V^*>(x)]\} = * \in \text{Sim}(x)$.

* Computational, statistical, or perfect.

Needs to be true for any (potentially malicious) verifier $V^*$, not just honest verifier $V$. If we write definition with $V$, it is called "honest verifier ZK" or HVZK.
- View \( V^* \) [\( \langle P, V \rangle(x) \)] is what \( V^* \) sees when interacting with \( P \)
- Sim(x) is the algorithm that writes down the transcript without interacting with \( P \)

* Remember: input to Sim(\( \cdot \)) essentially captures what the \( \langle P, V \rangle \) interaction leaks because that’s the information the verifier is allowed to use when writing down the transcript

How do we achieve ZK? What languages have ZK proofs?

\( \Rightarrow \) today we will prove that there is a ZK proof protocol for every language in NP!

Approach: give a ZK protocol for one NP-complete problem (HAMCYCLE), then a ZK protocol for any other NP language is just to reduce the instance to this language and use the same protocol

ZKP for Hamiltonian Cycle

\[
\text{Def: A Hamiltonian Cycle visits every node in a graph exactly once}
\]

Let \( \text{HAM} \) be the set (language) of graphs w/ a Hamiltonian Cycle

Trivial IP for \( \text{HAM} \):

\[
P(G) \quad \begin{array}{c} \text{finds a cycle} \\ \in G \end{array} \quad c \quad V(G) \quad \text{checks } c \rightarrow 0/1
\]
1) Complete? Yes!
2) Sound? Yes!
3) ZK? Probably not...
   \[ \Rightarrow \text{if } P=NP, \text{ then } V \text{ can compute the edges on the cycle itself, so this would be ZK!} \]

**A ZKP for HAM**

First, a sketch:

### $P(G)$
- $c \leftarrow$ Find Cycle ($G$)
- $\sigma \in$ permutation on vertices
- Commit to $\sigma$
- Commit to $\sigma(G)$

### $V(G)$
- $b \leftarrow \{0,1\}$
  - if $b=0$, open $\sigma, \sigma(G)$ // Shows $\sigma$ (not $c$)
  - if $b=1$, open subset of $\sigma(G)$ that is $\sigma(c)$ // Shows $\sigma(c)$ (not $\sigma$)

**Now, in detail:**

Let Commit be computationally hiding and binding
Let $G$ have $n$ vertices $[n] = \{1, 2, \ldots, n\}$ and an adjacency matrix $M \in \{0,1\}^{n \times n}$, $G \triangleq (n, M)$
For a permutation $\sigma$ on $[n]$, $\sigma(M)$ is $M'$ where $\forall i, j$, $M'(\sigma(i), \sigma(j)) = M_{ij}$
A cycle \( l \) is a list of \( n+1 \) vertices st.
- \( \forall i \in [n], M_{l_i, l_{i+1}} = 1 \) and \( \ell_{n+1} = \ell_1 \)
- \( \{ \ell_i : i \in [n] \} \) is a Hamiltonian.

\[
\begin{align*}
M &= \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 0 & 1 & 1 \\ 3 & 1 & 0 & 1 \\ 4 & 0 & 1 & 0 \end{bmatrix} \\
M' &= \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 0 & 0 & 1 \\ 3 & 1 & 1 & 0 \\ 4 & 0 & 1 & 1 \end{bmatrix}
\end{align*}
\]

\[
P((n, M), c) \quad V((n, M))
\]

\[
\sigma \in \text{Perms}([n])
\quad M' \leftarrow \sigma(M)
\]

\[
\forall i, j \in [n], r_{ij} \leftarrow \mathcal{R}
\quad c_{ij} \leftarrow \text{Commit}(M'_{ij}, r_{ij})
\]

\[
\forall i \in [n], s_i \leftarrow \mathcal{R}
\quad d_i \leftarrow \text{Commit}(\sigma(i), s_i)
\]

\[
\begin{align*}
&\text{if } b = 0: \\
&\text{all } c_{ij}, d_i \\
&\begin{array}{c}
\sigma, \text{all } r_{ij}, s_i \\
\end{array} \quad \begin{array}{c}
b \leftarrow \{0, 1\} \\
\sigma \text{ is a permutation?}
\end{array}
\begin{array}{c}
d_i \leftarrow \text{Commit}(\sigma(i), s_i) \\
M' \leftarrow \sigma(M) \\
c_{ij} \leftarrow \text{Commit}(M'_{ij}, r_{ij})
\end{array}
\]

\[
\begin{align*}
&\text{if } b = 1: \\
&\ell' \leftarrow \sigma(i) : i \in \ell \\
&\ell' \leftarrow [\sigma(i) : i \in \ell]\end{align*}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{all } c_{ij}, d_i \\
\ell', \forall i \in [n], r_{ij}, s_i \text{ and } M'_{ij} \leftarrow \\
\end{array} \\
\begin{array}{c}
b \leftarrow \{0, 1\} \\
\sigma \text{ is a permutation?}
\end{array}
\begin{array}{c}
d_i \leftarrow \text{Commit}(\sigma(i), s_i) \\
M' \leftarrow \sigma(M) \\
c_{ij} \leftarrow \text{Commit}(M'_{ij}, r_{ij})
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\text{if } \ell' \text{ is a cycle?}
\end{array} \\
\end{array}
\]
An example run of the protocol:

\[ n = 4 \]

\[ M = \begin{bmatrix}
    0 & 1 & 1 & 0 \\
    1 & 0 & 1 & 1 \\
    1 & 1 & 0 & 1 \\
    0 & 1 & 1 & 0 
\end{bmatrix} \]

\[ l = [1, 2, 4, 3, 17] \]

\[ 0 = \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 1\} + [s_1, s_2, s_3, s_4] \triangleleft R^n \]

\[ M' = \begin{bmatrix}
    0 & 0 & 1 & 1 \\
    0 & 0 & 1 & 1 \\
    1 & 1 & 0 & 1 \\
    1 & 1 & 0 & 0 
\end{bmatrix} \]

\[ \text{Commit} \]

\[ \text{Commit} \]

\[ l' = [2, 3, 1, 4, 2] \]

\[ \text{reveal when } b = 0 \]

\[ \text{reveal when } b = 1 \]

Complete: \( \checkmark \)

Sound: If \( M \not\equiv \text{HAM}, \forall \sigma, \sigma(M) \not\equiv \text{HAM} \).

So, if \( P \) commits to \( \sigma(M) \):

- if \( b = 1 \) (prob 50\%) \( \Rightarrow V \) rejects
- If \( P \) commits to something else
  - if \( b = 0 \) (prob 50\%) \( \Rightarrow V \) rejects (or, a binding break)

Soundness: \( \geq \frac{1}{2} - \) binding error

To prove ZK, we must define \( \text{Sim} \):

\[ \text{Sim}(n, M): \]

\[ b \triangleleft \{0, 1\} \]

\( \sigma \triangleleft \text{Perms } [\mathbb{Z}_n] \)

if \( b = 0 \):

\[ M' \leftarrow \sigma(M) \]

if \( b = 1 \):

\[ M' \leftarrow \sigma(\tilde{M}), l = [1, 2, ..., n, 1], l' \leftarrow \sigma(l) \]

let \( \tilde{M} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \)
// commit as in protocol
b' ← V*(M, commits); if b' ≠ b, restart (for dishonest V*)

// open as in protocol
if b = 0, l/l' are not needed
if b = 1, relationship M' = σ(M) isn't checked

Output transcript

Note: this is only HVZK (since b ∈ {0, 1})
- a malicious V* might bias b (so add b')
  L→ if b' and b are independent:
  Pr [restart] = 50% ⇒ 2 reps for a 2^-2 failure rate

L→ if not:
  b' is correlated w/ b is correlated w/ the msgs
  ⇒ attack on commitment hiding!

Now, we must show

\{ \text{View}_{V^*} [\langle P, V^*(G) \rangle^2] =_o \text{Sim}(G) \}
\{ \langle G, c_{ij}, d_i, b, \text{openings} \rangle \}

- all of these are distributed exactly as in the real protocol
- some of these are opened: they're also distributed exactly as in the real protocol
- the rest are un-opened: if they can be distinguished, we have an attack on hiding.
  (hybrid argument over all unopened commitments)