CSCI 1510

- ZKP for All NP (continued)
- Non-Interactive Zero-Knowledge Proofs
- Definitions of Secure Multi-Party Computation
Zero-Knowledge Proof (ZKP)

Let \((P, V)\) be a pair of PPT interactive machines. \((P, V)\) is a zero-knowledge proof system for a language \(L\) with associated relation \(RL\) if

1. **Completeness:** \(\forall (x, w) \in RL, \quad \Pr[\text{P}(x, w) \leftrightarrow \text{V}(x) \text{ outputs 1}] = 1\).
2. **Soundness:** \(\forall x \in L, \forall \text{ (PPT) } P^*, \quad \Pr[\text{P}^*(x) \leftrightarrow \text{V}(x) \text{ outputs 1}] \leq \text{negl}(n)\).
3. **Zero-Knowledge:** \(\forall \text{PPT } V^*, \exists \text{PPT } S \text{ s.t. } \forall (x, w) \in RL, \quad \text{Output}_{V^*}[\text{P}(x, w) \leftrightarrow V^*(x)] \approx S(x)\)
ZKP for Graph 3-Coloring (All NP)

NP language \( L = \{ G : G \text{ has 3-coloring} \} \)

NP relation \( R_L = \{ (G, 3\text{COL}) \} \)

\[ \pi : \{ \bullet \bullet \bullet \} \rightarrow \{ \bullet \bullet \bullet \} \]
Commitment Scheme

Sender
\[ m \in \{0, 1\} \]

Commit:
\[ r \in \{0, 1\}^n \]
\[ c := \text{Com}(m; r) \]

Decommit:
\[ (m, r) \]

Receiver

Verify:
\[ c = \text{Com}(m; r) \]

- **Perfectly Binding**: \( \forall r, s \in \{0, 1\}^n \), \( \text{Com}(0; r) \neq \text{Com}(1; s) \)

- **Computationally Hiding**: \( \text{Com}(0; \text{Un}) \approx \text{Com}(1; \text{Un}) \)
ZKP for Graph 3-Coloring

Input: \( G = (V, E) \)
Witness: \( \phi: V \to \{0,1,2\} \)

Given a perfectly binding commitment scheme \( \text{Com} \).

Soundness?

\( G \& L, \) by perfect binding of \( \text{Com} \),
\( \Pr[p^* \text{ not caught}] \leq (1 - \frac{1}{|E|})^{|E|} \approx e^{-n} \)

---

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Randomly sample } \pi: {0,1,2} \rightarrow {0,1,2} \quad \text{( n \cdot</td>
<td>E</td>
</tr>
<tr>
<td>\forall v \in V, \quad rv \in {0,1,2}^n, \quad C_v := \text{Com}(\pi(\phi(v)); rv)</td>
<td></td>
</tr>
<tr>
<td>{C_v}_{v \in V}</td>
<td></td>
</tr>
<tr>
<td>\begin{align*} \alpha &amp;= \pi(\phi(u)), \quad ru \ \beta &amp;= \pi(\phi(v)), \quad rv \end{align*} \quad \text{Verify: } C_u &amp;= \text{Com}(\alpha; ru) \ C_v &amp;= \text{Com}(\beta; rv) \ \alpha, \beta \in {0,1,2}^3, \quad \alpha \neq \beta</td>
<td></td>
</tr>
</tbody>
</table>

Completeness?
Zero-Knowledge?

∀PPT V^*, ∃PPT S s.t. ∀(x, w) ∈ R_l,

Output V^*[P(x, w) \leftrightarrow V^*(x)] \leq S(x)

**Simulator**

\[ (u, v') \in \mathcal{E} \]
\[ \alpha, \beta \in \{0, 1\}^n \text{ s.t. } \alpha \neq \beta \]
\[ r_u \in \{0, 1\}^n \text{, } C_u := \text{Com}(\alpha, r_u) \]
\[ r_v \in \{0, 1\}^n \text{, } C_v := \text{Com}(\beta, r_v) \]
\[ \forall v \in V \setminus \{u, v\} : \]
\[ r_v \in \{0, 1\}^n \text{, } C_v := \text{Com}(0, r_v) \]

\[ \{C_v\}_{v \in V} \]

**Verifier**

If \((u, v) = (u', v')\):
Reveal decommitments of \(C_u \& C_v\)

Otherwise rewind

\[ \alpha, r_u \]
\[ \beta, r_v \]
\( (u', v') \in E \)
\( \alpha, \beta \in \{0, 1, 2\} \) s.t. \( \alpha \neq \beta \)

Construct \( \pi : \{0, 1, 2\} \rightarrow \{0, 1, 2\} \) s.t.
\( \pi(\phi(u')) = \alpha \land \pi(\phi(v')) = \beta \)
\( \forall v \in V, \, \forall v' \in \{0, 1\}^n, \, C_v := \text{Com}(\pi(\phi(v)), r_v) \)

\( \{C_v\}_{v \in V} \)

Reveal discommitments of \( Cu \) & \( Cv \)
\( \alpha = \pi(\phi(u)), \, r_u \)
\( \beta = \pi(\phi(v)), \, r_v \)
\[ Pr[\text{failure}] \leq (1 - \frac{1}{|E|})^{n \cdot |E|} \approx e^{-n}. \]
$H_0$: Prover $\leftrightarrow$ Verifier

III (identical distribution of $T_1$)

$H_1$  
$sS_1$ (negligible failure probability)

$H_2$  
$cS_1$ (computational hiding of Com)

$H_3$: Simulator $\leftrightarrow$ Verifier
Non-Interactive Zero-Knowledge (NI-ZK) Proof

\[
\begin{array}{c}
\text{Prover} \\
\text{Input: } (x, w) \\
\pi \\
\text{Verifier} \\
\text{Input: } x \\
\text{Verify}
\end{array}
\]

- Completeness: \( \forall (x, w) \in R_L, \Pr[P(x, w) \rightarrow V(x) \text{ outputs 1}] = 1 \).
- Soundness: \( \forall x \in L, \forall P^*, \Pr[P^*(x) \rightarrow V(x) \text{ outputs 1}] \leq \text{negl}(n) \).
- Zero-Knowledge: \( \forall PPT V^*, \exists PPT S \text{ s.t. } \forall (x, w) \in R_L, \\text{Output}_{V^*}[P(x, w) \rightarrow V^*(x)] \approx S(x) \).

Is it possible?

Not in the "plain" model (assuming P\( \neq \)NP)

\( x \in L \) ? \( S(x) \rightarrow \text{Verify} \)
Model 1: Common Random String / Common Reference String (CRS)

- **Prover**
  - Input: \((x, w)\)
  - Computes: \(6 \leftarrow \text{Gen}(1^n)\)

- **Verifier**
  - Input: \(x\)
  - Computes: \(\Pi\)

- **Diagram**
  - Prover: \(S(x)\) generates both \((6, \Pi)\)

- **Zero-Knowledge**: \(\forall \text{PPT } V^*, \exists \text{PPT } S \text{ s.t. } \forall (x, w) \in R_L, \) 
  
  \[ \text{Output}_{V^*}[6 \leftarrow \text{Gen}(1^n), P(x, w, 6) \rightarrow V^*(x, 6)] \simeq S(x) \]

- **Alternatively**: \(6 \leftarrow \text{Gen}(1^n), P(x, w, 6) \simeq S(x)\)
Model 2: Random Oracle Model

- **Prover**
  - Input: \((X, W)\)

- **Verifier**
  - Input: \(X\)

\(S\) controls input/output behavior of RO
**Fiat-Shamir Heuristic**

Public-Coin Honest-Verifier ZK (HVZK) \( \Rightarrow \) NIZK in the RO model

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> ((x, w))</td>
<td><strong>Input:</strong> (x)</td>
</tr>
</tbody>
</table>

\[
m_1 
\]

\[
\leftarrow 6_1 
\]

\[
m_2 
\]

\[
\leftarrow 6_2 
\]

\[
m_3 
\]

\[
6_1 := H(x \| m_1) 
\]

\[
6_2 := H(x \| m_1 \| m_2) 
\]
Secure Multi-Party Computation

Alice

Second date?
\[ f(x, y) = x \land y \]

Bob

Who is richer?
\[ f(x, y) = \begin{cases} 0 & \text{if } x > y \\ 1 & \text{otherwise} \end{cases} \]

Common friends?
\[ f(x, y) = x \land y \]
Secure Two-Party Computation (2PC)

Alice

Bob

Applications:
- Password Breach Alert (Chrome/Firefox/Azure/iOS Keychain)
- Privacy-Preserving Contact Tracing for COVID-19 (Apple & Google)
- Ads Conversion Measurements / Personalized Advertising (Google/Meta)

\[ z = f(x, y) \]
Secure Multi-Party Computation (MPC)

\[
\begin{align*}
\forall & \ x_1, \ x_2, \ x_3, \ x_4, \ \ldots, \ x_n \\
\exists & \ z = f(x_1, \ldots, x_n)
\end{align*}
\]
Secure Multi-Party Computation (MPC)

Applications:

- Privacy-Preserving Inventory Matching (J.P. Morgan)
- Setup Ceremony to securely generate CRS (Zcash)
- Distributed Key Management (Unbound / Coinbase)
- Federated Learning (Google Keyboard Search Suggestion)
- Auctions (Danish sugar beet auction)
- Boston gender wage gap (Boston Women's Workforce Council)
- Study / Analysis on Medical Data
- Fraud Detection (banks)
Setting

- $n$ parties $P_1, P_2, \ldots, P_n$
  - with private inputs $x_1, x_2, \ldots, x_n$

- Jointly compute $f(x_1, x_2, \ldots, x_n)$

Communication:
- Authenticated secure point-to-point channels between each pair $(P_i, P_j)$
  - sometimes also assume broadcast channel

- The adversary can "corrupt" a subset of the parties
  - (e.g. at most $t$ parties)

What properties do we want?
General Security Properties

• Correctness: The function is computed correctly.

• Privacy: Only the output is revealed.

• Independence of Inputs: Parties cannot choose inputs depending on others' inputs.

• Security with Abort: Adversary may “abort” the protocol.
  (preventing honest parties from receiving the output)

• Fairness: If one party receives output, then all receive output.

• Guaranteed Output Delivery (GOD): Honest parties always receive output.
Adversary’s Power

Allowed adversarial behavior:

- Semi-honest / passive / honest-but-curious:
  Follow the protocol description honestly, but try to extract more information by inspecting transcript.

- Malicious / active:
  Can deviate arbitrarily from the protocol description.

Adversary’s Computing Power:

- Unbounded computing power ⇒ Information-Theoretic (IT) Security
- PPT bounded ⇒ Computational Security
Security Against Semi-Honest Adversaries

Alice

Bob

\[
\begin{align*}
&\text{Alice's view:} \\
\text{View}_A^T (x, y, n) := (x, \text{ internal random tape } r, \text{ messages from Bob}) \\
\end{align*}
\]

Given \(x, f(x, y)\), Alice's view can be "simulated."
Security Against Semi-Honest Adversaries

**Def (Semi-honest security for 2PC)**

Let $f$ be a functionality. We say a protocol $\Pi$ securely computes $f$ against semi-honest adversaries if $\exists$ PPT algorithms $S_A, S_B$ s.t. $\forall x,y,$

\[
\begin{align*}
\left\{ \left( S_A \left( 1^n, x, f(xy) \right) \right) \right\}_{n \in \mathbb{N}} & \approx \left\{ \left( \text{View}^\Pi_A \left( x, y, n \right) \right) \right\}_{n \in \mathbb{N}} \\
\left\{ \left( S_B \left( 1^n, y, f(xy) \right) \right) \right\}_{n \in \mathbb{N}} & \approx \left\{ \left( \text{View}^\Pi_B \left( x, y, n \right) \right) \right\}_{n \in \mathbb{N}}
\end{align*}
\]

perfect/statistical/computational

\[\equiv \preceq \triangleq\]
Security Against Malicious Adversaries

\[ \text{Alice} \rightarrow \text{Bob} \]

\[ x \rightarrow y \]

\[ f(x,y) \downarrow \]

\[ f(x,y) \downarrow \]

\text{Alice's view:} \quad \text{View}_{A}^{T}(x, y, n) = (x, \text{ internal random tape } r, \text{ messages from Bob})

Given $x$, $f(x,y)$, Alice's view can be "simulated".

What output?
What's the best we can hope for? (Ideal World)
Security Against Malicious Adversaries (Real/Ideal Paradigm)

**Execution in the Real World:**

(PPT) adversary $A$ corruting party $i \in \{\text{Alice, Bob}\}$

$$\text{REAL}_{A,i} := (\text{A's output})$$

**Execution in the Ideal World:**

PPT adversary $S$ corrupting party $i \in \{\text{Alice, Bob}\}$

$$\text{IDEAL}_{S,i} := (\text{S's output})$$

**Def (malicious security for 2PC):**

Let $f$ be a functionality. We say a protocol $\Pi$ securely computes $f$ against malicious adversaries if $\forall (\text{PPT}) A$ in the real world, $\exists \text{PPT } S$ in the ideal world s.t. $\forall i \in \{\text{Alice, Bob}\}$, $\forall x, y$,

$$\left\{\text{REAL}_{A,i}(x, y, n)\right\}_{n \in \mathbb{N}} \simeq \left\{\text{IDEAL}_{S,i}(x, y, n)\right\}_{n \in \mathbb{N}}$$