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 \( R_L \) if

- **Completeness:** \( \forall (x, w) \in R_L, \Pr [ P(x, w) \leftrightarrow V(x) \text{ outputs 1} ] = 1 \).

- **Soundness:** \( \forall x \in L, \forall (\text{PPT}) P^*, \Pr [ P^*(x) \leftrightarrow V(x) \text{ outputs 1} ] \leq \text{negl}(n) \).

- **Zero-Knowledge:** \( \forall \text{PPT} V^*, \exists \text{PPT} S \text{ s.t. } \forall (x, w) \in R_L, \text{Output}_{V^*} [ P(x, w) \leftrightarrow V^*(x) ] \approx S(x) \)
ZKP for Graph 3-Coloring (All NP)

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

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

$\pi : \{ \circ \circ \circ \} \rightarrow \{ \circ \circ \circ \}$
Commitment Scheme

Sender

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

\[ r \in \{0,1\}^n \]

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

Receiver

\[ C \]

Decommit:

\[ (m, r) \]

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: $\phi: V \rightarrow \{0,1,2\}$

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

**Prover**

Randomly sample $\pi: \{0,1,2\} \rightarrow \{0,1,2\}$

$\forall v \in V, \; r_v \in \{0,1\}^n, \; c_v = \text{Com}(\pi(\phi(v)); r_v)$

Repeat $\{c_v\}_{v \in V}$ $n \cdot |E|$ times

**Verifier**

Randomly pick an edge $(u,v) \in E$

$(u,v)$

Reveal decommitments of $c_u$ & $c_v$

$\alpha = \pi(\phi(u)), \; r_u$

$\beta = \pi(\phi(v)), \; r_v$

Verify:

$c_u = \text{Com}(\alpha; r_u)$
$c_v = \text{Com}(\beta; r_v)$

$\alpha, \beta \in \{0,1,2\}, \; \alpha \neq \beta$

Completeness?
Zero-Knowledge?

\[ \forall \text{PPT } V^*, \ \exists \text{PPT } S \ \text{st. } \forall (x,w) \in R_L, \]
\[ \text{Output}_{V^*}[P(x,w) \leftrightarrow V^*(x)] \preceq S(x) \]

Simulator

\[(u',v') \triangleq E\]
\[\alpha, \beta \triangleq \{0,1,2\} \text{ st. } \alpha \neq \beta\]
\[RU' \triangleq \{0,1\}^n, \ \text{Cu} : = \text{Com} (\alpha; RU')\]
\[RV' \triangleq \{0,1\}^n, \ \text{Cv} : = \text{Com} (\beta; RV')\]
\[\forall u, v \in V \setminus \{u', v'\}: \]
\[RV \triangleq \{0,1\}^n, \ \text{Cv} : = \text{Com} (v; RV)\]
\[\{CV\}_{v \in V} \]

Verifier*

\[(u, v)\]

If \((u, v) = (u', v'):\]
Reveal decommitments of \(Cu\) & \(Cv\)

Otherwise rewind \[\alpha, RU\]
\[\beta, RV\]
Non-Interactive Zero-Knowledge (NIzk) Proof

**Prover**

Input: \((x, W)\)

**Verifier**

Input: \(x\)

\[\Pi\]

Verify

- **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:** \(\forall \text{PTT } V^*, \exists \text{PTT } S \text{ s.t. } \forall (x, w) \in R_L, \text{Output}_{V^*}[P(x, w) \rightarrow V^*(x)] \approx S(x)\).

Is it possible?
Model 1: Common Random String / Common Reference String (CRS)

Prover

Input: \((x, w)\)

\(6 \leftarrow \text{Gen}(1^n)\)

Verifier

Input: \(x\)

Verify

\(S(x)\) generates both \((6, \pi)\)

Zero-Knowledge: \(\forall \text{PPS } V^*, \exists \text{PPS } S \text{ s.t. } \forall (x, w) \in R_L, \) 

Output\(_{V^*}\) \([6 \leftarrow \text{Gen}(1^n), P(x, w, 6) \rightarrow V^*(x, 6)] \approx S(x)\)

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

- Prover
  - Input: \((X, W)\)
- Verifier
  - Input: \(x\)

\(H\)

\(\pi\)

\(S\) 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>Input: ((x, w))</td>
<td>Input: (x)</td>
</tr>
<tr>
<td>(m_1)</td>
<td>(6_1 \leftarrow D_2)</td>
</tr>
<tr>
<td>(\leftarrow 6_2)</td>
<td>(\leftarrow 6_2 \leftarrow D_2)</td>
</tr>
<tr>
<td>(m_2)</td>
<td></td>
</tr>
<tr>
<td>(\leftarrow m_3)</td>
<td></td>
</tr>
</tbody>
</table>

\[6_1 := H(\langle x || m_1 \rangle)\]
\[6_2 := H(\langle x || m_1 || m_2 \rangle)\]
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

\[ \begin{align*}
& x \\
\rightarrow & \\
\leftarrow & \\
& z = f(x,y)
\end{align*} \]

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)
Secure Multi-Party Computation (MPC)

\[ z = f(x_1, \ldots, x_n) \]
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

\[ \text{Alice} \begin{array}{ccc} x & \rightarrow & y \\ \downarrow & & \downarrow \\ f(x, y) & & f(x, y) \end{array} \text{Bob} \]

**Alice's view:**

\[ \text{View}_A^T (x, y, n) := (x, \text{ internal random tape } r, \text{ messages from Bob}) \]

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

$$\left\{ \left( S_A(1^n, x, f(xy)) \right) \right\}_{n \in \mathbb{N}} \cong \left\{ \left( \text{View}^\Pi_A(x, y, n) \right) \right\}_{n \in \mathbb{N}}$$

$$\left\{ \left( S_B(1^n, y, f(xy)) \right) \right\}_{n \in \mathbb{N}} \cong \left\{ \left( \text{View}^\Pi_B(x, y, n) \right) \right\}_{n \in \mathbb{N}}$$

perfect/statistical/computational

$$\equiv \preceq \subseteq$$
Security Against Malicious Adversaries

Alice

\[\begin{align*}
\varnothing & \quad \rightarrow \quad \varnothing \\
\text{x} & \quad \rightarrow \quad \text{y} \\
\downarrow & \\
\text{f(x,y)} & \quad \downarrow
\end{align*}\]

Bob

Alice's view:

\[\text{View}_A^{\pi}(x, y, n) := (x, \text{ internal random tape } r, \text{ messages from Bob})\]

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

What output?
What's the best we can hope for? (Ideal World)

```
+---------------------+
| Trusted Party       |
+---------------------+
|                    |
|   ▼                |
|                    |
|  ▼                |
|                    |
| Alice ▶ f(x', y) ▶ f(x', y) ▶ Bob |
| x                  | y              |
```

- Alice sends $x$ to the trusted party, which in turn computes $f(x', y)$ and sends it to Bob.
- Bob receives $f(x', y)$ and computes $x'$, which is then sent back to Alice.
- The trusted party ensures the computation remains confidential and secure.
Security Against Malicious Adversaries (Real/Ideal Paradigm)

**Execution in the Real World:**

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

\[
\text{REAL}^\Pi_{A,i} := \begin{pmatrix} \text{A's output} \\ \text{Honest party's output in Real World} \end{pmatrix}
\]

**Execution in the Ideal World:**

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

\[
\text{IDEAL}^f_{S,i} := \begin{pmatrix} \text{S's output} \\ \text{Honest party's output in Ideal World} \end{pmatrix}
\]

**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. \( \forall i \in \{ \text{Alice, Bob} \}, \forall x, y, \)

\[
\{ \text{REAL}^\Pi_{A,i}(x, y, n) \}_{n \in \mathbb{N}} \simeq \{ \text{IDEAL}^f_{S,i}(x, y, n) \}_{n \in \mathbb{N}}
\]