Logistics: HW2 due this Friday at 5pm (submit via Gradescope).
HW3 posted this Friday.
See Piazza for office hour information for upcoming week (course staff is traveling).

Previous lecture: 2-party computation (Yao’s garbled circuits)

Suppose Alice and Bob want to compute function \( f \) on joint input \((x, y)\):

- Alice (\( x \)) requests for \( x \).
- Bob (\( y \))

\( OT \) requests for \( x \) \( \rightarrow \) \( OT \) responses
\( garbled \) circuit for \( f \) \( \rightarrow \) \( garbled \) inputs for \( y \)

\( f(x, y) \) requires algebraic assumptions (more expensive)
\( only \) requires cheap symmetric primitives

HW3: Realize large number of OTs from a small (\( \approx t \)) number of base OTs

Security: Neither party learns more from protocol other than what is revealed by \( f(x, y) \)

- e.g. \( \{\text{View}_A(x, y)\} \approx \{S(1^t, x, f(x, y))\} \)
- view of Alice in protocol execution
- Real protocol transcript from \( y \)
- can be simulated just given
- party’s input and output of computation.

Question: What if we have more than two parties?

This lecture: Secret sharing and MPC

Beaver triples and MPC in the preprocessing model

Secret sharing: Suppose we have a secret and want to distribute it among \( n \) parties such that any \( t \) of them can subsequently recover the secret and any \((t-1)\) subset cannot [e.g., Board of directors at Coca-Cola want to protect Coca-Cola recipe]

Def. A \((t,n)\)-secret sharing scheme over a message space \( M \) and share space \( S \) consists of two efficient algorithms:

- Share: \( M \to \mathbb{G}^n \)
- Reconstruct: \( S^t \to M \)

with the following properties:

- Correctness: Any \( t \) shares can be used to reconstruct \( m \):

  \[
  \forall m \in M : (s_1, \ldots, s_n) \leftarrow \text{Share}(m) \Rightarrow \forall S \subseteq \{s_1, \ldots, s_n\} \text{ where } |S| = t : \text{Reconstruct}(S) = m
  \]

- Security: Need at least \( t \) shares to learn the secret

  \[
  \forall m \in M, \forall I \subseteq [n] \text{ where } |I| < t : \{s_1, \ldots, s_n \} \leftarrow \text{Share}(m) : s_i \text{ for all } i \in I \} \neq \{s_1, \ldots, s_n \} \leftarrow \text{Share}(m) : s_i \text{ for all } i \notin I \}
  \]

- Note: can relax to computational indistinguishability (in which case, Share takes additional security parameter)
Examples: Additive secret sharing \([n \text{ out of } n]\): \(M = \mathbb{Z}_p^n\), \(S = \mathbb{Z}_p^n\)

- Share \((m)\): Sample \(r_1, \ldots, r_n \in \mathbb{Z}_p\) and set \(r_n = m + \sum_{i=1}^{n-1} r_i \in \mathbb{Z}_p\)

- Reconstruct \((r_1, \ldots, r_n)\): Output \(\sum_{i=1}^{n} r_i\)

Combinational secret sharing \([t \text{ out of } n]\): Will use a symmetric encryption scheme over \(\mathbb{Z}_p^n\) (e.g., AES-CTR)

- Share \((m)\): Sample \(n\) keys \(k_i, \ldots, k_n\) for encryption scheme.

For every \(t\)-subset \([i_1, \ldots, i_t] \subseteq [n]\), encrypt \(m\) using \(k_{i_1}, \ldots, k_{i_t}\) (e.g., \(\text{Enc}(k_{i_1}, \ldots, k_{i_t}, m)\))

Let \([ct]\) be the collection of ciphertexts

Output shares \((k_{i_1}, \text{Enc}(k_{i_1}, m)), \ldots, (k_{i_t}, \text{Enc}(k_{i_t}, m))\)

- Reconstruct \((k, (ct)), \ldots, (k, (ct))\): Each share is one \(ct \in [ct]\) encrypted under \(k_{i_1}, \ldots, k_{i_t}\), so decrypt accordingly

 Shamir secret sharing \([t \text{ out of } n]\): \(M = \mathbb{F}_p^n\), \(S = \mathbb{F}_p^n\) (require \(p > n\))

\[ f(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1} \]

Output shares \((x_i, f(x_i)) \forall i \in [n]\) Each share is just 2 field elements (independ of threshold \(t\) or \(n\))

- Reconstruct \((y_i, y_j), \ldots, (y_i, y_k)\): Interpolate the unique polynomial \(f\) of degree \(t-1\) defined by the points \((x_i, y_i), \ldots, (x_k, y_k)\)

Correctness: Follows by uniqueness of the interpolating polynomial

Security: Given any subset of \((t-1)\) shares \((y_1, y_2), \ldots, (y_i, y_k)\), and any message \(m \in \mathbb{F}_p\), there is a unique polynomial \(f\) of degree \(t-1\) where

\[ f(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1} \]

Thus, any \((t-1)\) shares can be consistent with secret sharing of any message \(m \Rightarrow \text{information-theoretic security} \)

Efficiency: Secret sharing and secret reconstruction correspond to polynomial evaluation and interpolation (over \(\mathbb{F}_p\)). Both of these operations can be expressed as linear functions:

- Suppose \(f(x) = \alpha_0 + \alpha_1x + \cdots + \alpha_{t-1}x^{t-1}\)

- Evaluation of \(f\) at points \(x_1, x_2, \ldots, x_n\) can be expressed as matrix-vector product:

\[
\begin{align*}
\begin{pmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\
1 & x_2 & x_2^2 & \cdots & x_2^{t-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & x_n^2 & \cdots & x_n^{t-1}
\end{pmatrix}
\begin{pmatrix}
\alpha_0 \\
\alpha_1 \\
\vdots \\
\alpha_{t-1}
\end{pmatrix}
= 
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{pmatrix}
\end{align*}
\]

in particular, \(y_i = f(x_i) \forall i \in [n]\)

* Polynomial evaluation corresponds to computing product \(X\alpha\) and interpolation corresponds to computing the product \(X^\top y\)
* Both of these operations can be done in time \(O(n \log n)\) using the Fast Fourier Transform (FFT)
Computing on secret-shared data: Another paradigm for 2PC (and MPC) - better-suited for evaluating arithmetic circuits

Assuming $43: Amazing, insight

Computing on shares: Given shares of $X_a$ and $X_b$,

$[X_a + X_b] = [X_a] + [X_b]$ (component-wise addition)

Specifically if $[X_a] = (X_{a1}, X_{a2}, X_{a3})$ where $X_{a1} + X_{a2} + X_{a3} = X_a \in \mathbb{F}_p$

$[X_b] = (X_{b1}, X_{b2}, X_{b3})$ where $X_{b1} + X_{b2} + X_{b3} = X_b \in \mathbb{F}_p$

then $[X_a + X_b] = (X_{a1} + X_{b1}, X_{a2} + X_{b2}, X_{a3} + X_{b3})$ and $([X_a + X_b]) + ([X_a] + [X_b]) = X_a + X_b \in \mathbb{F}_p$

More generally:
1. Share addition: $[X_a + X_b] = [X_a] + [X_b]$
2. Scalar multiplication: $[kX_a] = k \cdot [X_a]$
3. Addition by constant: $[X_a + k] = (X_{a1} + k, X_{a2}, X_{a3})$

Question: How to multiply shared values?

Amazing insight due to Beaver: Suppose parties have a secret-sharing of a random product: $[a], [b], [c]$ where $c = ab \in \mathbb{F}_p$

Then, given $[a]$ and $[b]$, we proceed as follows:

1. Each party computes $[x-a]$ and publishes their share of $x-a$.
2. Each party computes $[y-b]$ and publishes their share of $y-b$.
3. All of the parties compute non-interactively:

$[z] = [c] + [(x-a)(y-b)] = (x-a)(y-b)$

Claim: $z = xy$. Follows by following calculation:

$z = c + x(y-b) + y(x-a) - (x-a)(y-b)$

$= ab + xy - bx + xy - yx + bx + 2xy - ab$

$= xy$

Observe: Parties only see $x-a$ and $y-b$ in this protocol. Since $a, b$ are uniformly random and unknown to the parties, $x-a$ and $y-b$ is a one-time pad encryption of $x$ and $y$. Resulting protocol provides information-theoretic privacy for parties' inputs.

Assuming we have access to Beaver multiplication triples, we can evaluate any arithmetic circuit as follows (among $n$-parties):

1. Every party secretly shares their input with every other party.
2. For each addition gate in the circuit, parties locally compute on their shares.
3. For each multiplication gate in the circuit, parties run Beaver's multiplication protocol (using different triple each time!).
4. Every party publishes share of the output; parties run share reduction to obtain output.
<table>
<thead>
<tr>
<th>Comparison with Yao</th>
<th>Secret Sharing</th>
<th>Yao</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of computation</td>
<td>Arithmetic circuits ($F_p$)</td>
<td>Boolean circuits</td>
</tr>
<tr>
<td>Number of parties</td>
<td>Arbitrary ($n$)</td>
<td>2</td>
</tr>
<tr>
<td>Round complexity</td>
<td>Depth of circuit</td>
<td>2</td>
</tr>
<tr>
<td>Communication</td>
<td>Information-theoretic (with Beaver triples)</td>
<td>Computational</td>
</tr>
<tr>
<td>Security</td>
<td>*Can be improved further!</td>
<td>*leverages several optimizations (half-gates + free XOR)</td>
</tr>
</tbody>
</table>

Aside: Preprocessing is also possible for OT (OT correlations): gives fast, information-theoretic OT in the online phase.

**Question:** Where do Beaver triples come from?

Typically, generated in an offline “preprocessing” phase. Many techniques:
- Trusted dealer (e.g., Intel SGX) can generate them
- Using oblivious transfer (OT) - accelerate using OT extensions (HW2, Problem 4) assuming existence of OT
- Using somewhat homomorphic encryption (discussed later in the course)

**Question:** What if parties are malicious? [So far, everything is only semi-honest secure]

The GMW compiler transforms any protocol with semi-honest security to one with malicious security.

High-level idea: Parties include a zero-knowledge proof that each step of the protocol was performed according to the protocol specification.

**Corollary:** Anything that can be computed with a trusted party can be computed without!
- For $n$-parties, if we have fewer than $n/3$ corrupted parties, this is even possible information-theoretically!
- With cryptographic assumptions (i.e., OT), we can support $n-1$ corrupted parties (with some caveats...).