Cryptographic Protocols
Spring 2020
MPC Part 1

Sum Protocol
Goal: Compute sum of inputs

Protocol:
1. \( y_1 = r + x_1 \)
2. \( y_2 = y_1 + x_2 \)
3. \( y_3 = y_2 + x_3 \)
4. \( y_4 = y_3 + x_4 \)
5. \( y_5 = y_4 + x_5 \)
6. \( y_6 = y_5 + x_6 \)
7. \( y_7 = y_6 + x_7 \)

\( y = y_7 \)

Analysis: 1 passive cheater? 2 passive? 1 active? 2 active?

Specification
0. \( \forall P_i: \text{input } x_i \)
1. \( \forall P_i: \text{send } x_i \text{ to TTP} \)
2. TTP: \( y = \sum x_i \)
3. TTP: send \( y \) to \( \forall P_j \)

Model

Parties and Channels
- \( n \) parties \( P_1, \ldots, P_n \)
- Secure channels among parties
- Broadcast channels

Adversary
- Central adversary (collaborating parties)
- Corrupts \( t \) parties
- Passive vs active

Security
- Information-theoretic vs. Cryptographic

More Examples

Examples
- Statistics (first sex, tax evading, etc.)
- Elections / Votes / Auctions
- Millionaires problem
- Loans (several banks, same guarantee)
- ZK-proofs (Peggy sends witness to TTP, who checks & sends 0/1 to Vic)

Secure Function Evaluation (evaluate function \( f \) on all inputs)
1. \( \forall P_i: \text{send input } x_i \text{ to TTP} \)
2. TTP: compute \( (y_1, \ldots, y_n) = f(x_1, \ldots, x_n) \)
3. TTP: send output \( y_i \) to \( \forall P_j \)

Limitations
- Poker, etc (not realizable with TTP)
### Known Results

<table>
<thead>
<tr>
<th>Setting</th>
<th>Condition</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cryptographic, passive</td>
<td>( t &lt; n )</td>
<td>[GMW87]</td>
</tr>
<tr>
<td>Cryptographic, active</td>
<td>( t &lt; n/2 )</td>
<td>[GMW87]</td>
</tr>
<tr>
<td>Information-theoretic, passive</td>
<td>( t &lt; n/2 )</td>
<td>[BGW88, CCD88]</td>
</tr>
<tr>
<td>Information-theoretic, active</td>
<td>( t &lt; n/3 )</td>
<td>[BGW88, CCD88]</td>
</tr>
<tr>
<td>Information-theoretic, active assuming broadcast</td>
<td>( t &lt; n/2 )</td>
<td>[RB89, Bea91]</td>
</tr>
</tbody>
</table>

### Oblivious Transfer

**Rabin-OT**

- **Sender**
- **Receiver**

\[
\begin{align*}
    r = 0 : & \quad s \\
    r = 1 : & \quad \perp
\end{align*}
\]

**1-2-OT**

- **Sender**
  - 0
  - 1

- **Receiver**
  - 0
  - 1

**1-k-OT**

- **Sender**
  - \( s_1, \ldots, s_k \)

- **Receiver**
  - \( s_i \)

### 1-2-OST based on RSA and AES

**Sender**

- Messages \( s_0, s_1 \)

**Receiver**

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

Generate RSA-Keys

\[
\begin{align*}
    n_0, e_0, d_0 \text{ and } n_1, e_1, d_1 \\
    u_k \text{ at random, } \quad u = k^b \mod n_b
\end{align*}
\]

\[
\begin{align*}
    k_0 &= u^{d_0} \mod n_0 \\
    k_1 &= u^{d_1} \mod n_1 \\
    y_0 &= AES_{k_0}(s_0) \\
    y_1 &= AES_{k_1}(s_1) \\
    s_b &= AES_{k_b}^{-1}(y_b)
\end{align*}
\]

### MPC from OT

**Truth table:**

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

**Starting Point**

- 2 parties Alice and Bob
- Inputs \( a \in A \) and \( b \in B \)
- Fixed function \( f : A \times B \to C \)

**Protocol**

1. Alice sends \([ f(a, b_1) \mid f(a, b_2) \mid \ldots \mid f(a, b_2) ]\) via OT
2. Bob selects \( b \)-th value

**Analysis:**

- Security
- Efficiency

**Extension:**

3 parties . . .

### Multi-Party Computation: Goal II

**Specification**

**Protocol**

**Trusted party**

- Receive input
- \( \oplus \) and \( \otimes \) over finite field \( F \)
- Give output

**Simulating players . . .**

- \( n \) players: \( P = \{ P_1, \ldots, P_n \} \)
- Players can \( \oplus \) and \( \otimes \) in \( F \)
- Players can communicate
Sum Protocol III

Protocol:

\[
\begin{array}{cccccccc}
& x_1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\
& x_2 & x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\
& x_3 & x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\
& \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
& x_n & x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nn} \\
\end{array}
\]

Analysis: 1 passive cheater? 2 passive? 1 active? 2 active?

Secret-Sharing Schemes – Definition

Intuition

- Dealer D can share a secret s among parties \( P \)
- Qualified subsets of \( P \) can reconstruct s (w/o D)
- Access structure \( \Gamma \subseteq 2^P \)

Definition

A secret-sharing scheme for parties \( P \) and access structure \( \Gamma \) is a pair of protocols (SHARE, RECONSTRUCT), s.t.

- Correctness:
  1. After SHARE, there is a unique value \( s' \), where \( s' = s \) (the dealer's input) if the dealer is honest
  2. After RECONSTRUCT(\( M \)), if \( M \in \Gamma \), all players in \( M \) know \( s' \)
- Privacy: After SHARE, non-qualified sets have no information about \( s \)

Shamir’s Secret-Sharing Scheme (1/3)

Goal

- \( n \) parties, \( k \) needed for reconstruction
- Threshold access structure \( \Gamma = \{ M \subseteq P : |M| \geq k \} \)

Idea

- Random polynomial \( f \) of degree d is defined by \( d + 1 \) points
- \( s = f(0) \) = secret, party \( P_i \) gets share \( s_i = f(x_i) \) for fixed \( x_i \)
- Degree \( d = k-1 \) \( \Rightarrow \) \( k \) parties can reconstruct, \( k-1 \) cannot

Shamir’s Secret-Sharing Scheme (2/3)

Starting Point: To each party \( P_i \), some unique \( \alpha_i \in \mathbb{F} \setminus \{0\} \) is assigned.

SHARE

1. D: choose random \( f \) with \( f(0) = s \) and \( \deg(f) \leq d \)
   (i.e., choose random \( r_1, \ldots, r_d \)), let \( f(x) = s + r_1x + \cdots + r_dx^d \)
2. D: send \( s_i = f(\alpha_i) \) to \( \forall P_j \)

RECONSTRUCT

1. \( \forall P_j \): send \( s_j \) to \( P \)
2. \( P \): compute \( s \) with Lagrange interpolation:
   \[
   f(x) = \sum_{i=1}^{n} \lambda_i(x) \cdot s_i, \text{ where } \lambda_i(x) = \prod_{j 
eq i}^{n} \frac{x - \alpha_j}{\alpha_i - \alpha_j}
   \]
   hence \( s = \sum_{i=1}^{n} w_i s_i \), where \( w_i = \lambda_i(0) = \prod_{j 
eq i}^{n} \frac{-\alpha_j}{\alpha_i - \alpha_j} \)

Shamir’s Secret-Sharing Scheme (3/3)

Analysis for passive adversary:

Correctness

1. by inspection, \( s' = f(0) \)
2. due to Lagrange interpolation (given \( |M| \geq k = d + 1 \))

Privacy

- For \( d \leq k-1 \) shares, every secret \( s \) is “compatible” (same #polys)
- \( \Rightarrow \) adversary with \( < k \) shares obtains no information about \( s \).

Note

- Degree is at most \( d \), not exactly \( d \)
- Otherwise privacy violation
Linear Secret-Sharing Schemes

**Definition:** Secret-Sharing is linear, if each share $s_i = L_i(s, r_1, \ldots, r_{\ell})$:

$$
\begin{bmatrix}
  s_1 \\
  s_2 \\
  \vdots \\
  s_n
\end{bmatrix} =
\begin{bmatrix}
  A_{10} & A_{11} & \cdots & A_{1\ell} \\
  A_{20} & A_{21} & \cdots & A_{2\ell} \\
  \vdots & \vdots & \ddots & \vdots \\
  A_{n0} & A_{n1} & \cdots & A_{n\ell}
\end{bmatrix}
\begin{bmatrix}
  s \\
  r_1 \\
  \vdots \\
  r_{\ell}
\end{bmatrix}
$$

Shamir Sharing is linear

$$A =
\begin{bmatrix}
  1 & \alpha_1 & \cdots & \alpha_n \\
  1 & \alpha_1^2 & \cdots & \alpha_n^2 \\
  \vdots & \vdots & \ddots & \vdots \\
  1 & \alpha_1^n & \cdots & \alpha_n^n
\end{bmatrix}
$$
(Van der Monde Matrix)

MPC Passive: Secret-Sharing and Addition

**Setting**
- $n$ parties, $t$ corrupted (passive), $t < n/2$

**Secret Sharing**
- Shamir-Sharing with degree $t$
  - $a$, $b$, … shared by $a_1$, …, $a_n$, $b_1$, …, $b_n$, etc.
  - Every $P_i$ computes $c_i = L(a_i, b_i, \ldots)$
  - $c_1$, …, $c_n$ is a sharing of $c = L(a, b, \ldots)$

**Addition and Linear Functions**
- Shamir-Sharing is linear $\Rightarrow$ apply linear function on shares
- $a$, $b$, … shared by $a_1$, …, $a_n$, $b_1$, …, $b_n$, etc.
- Every $P_i$ computes $c_i = L(a_i, b_i, \ldots)$
- $c_1$, …, $c_n$ is a sharing of $c = L(a, b, \ldots)$

MPC Passive: Multiplication

**Starting Point:** $a, b$ shared by $a_1, \ldots, a_n$, $b_1, \ldots, b_n$

**Idea**
- Every $P_i$ computes $d_i = a_i \cdot b_i$
- Observe: $d_1, \ldots, d_n$ is some-kind-of sharing of $c = a \cdot b$
- Could compute $c$ from $d_1, \ldots, d_n$: $c = \sum_i w_id_i$ (Lagrange)
- Can compute $c$ as MPC: Every $P_i$ has input $d_i$, compute (sharing of) $c$

**Multiplication Protocol**
1. $P_i$: compute $d_i = a_i \cdot b_i$.
2. $P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $P_j$: compute $c_j = w_1d_{ij} + \cdots + w_nd_{nj}$.

Passive Protocol

**Share input**
0. $P_i$ has input $s$.
1. $P_i$: select $r_1, \ldots, r_t$ at random.
2. $P_i$: comp. $\left(\begin{array}{c}
  s_1 \\
  r_1
\end{array}\right) = A \left(\begin{array}{c}
  r_{\ell} \\
  r_{\ell}
\end{array}\right)$.
3. $P_i$: send $s_j$ to every $P_j$.

**Reconstruct Output**
0. $a$ is shared by $a_1, \ldots, a_n$.
1. $P_i$: send $a_i$ to $P_i$.
2. $P_i$: comp. $a = L(a_1, \ldots, a_n)$.
3. $P_i$: compute $c_i = L(a_i, b_i, \ldots)$.

**Addition and Linear Functions**
0. $a, b, \ldots$ are shared by $a_1, \ldots, a_n$, $b_1, \ldots, b_n$, etc.
1. $P_i$: compute $c_i = L(a_i, b_i, \ldots)$.

**Multiplication**
0. $a, b$ are shared by $a_1, \ldots, a_n$, $b_1, \ldots, b_n$.
1. $P_i$: compute $d_i = a_i b_i$.
2. $P_i$: share $d_i \rightarrow d_{i1}, \ldots, d_{in}$.
3. $P_j$: compute $c_j = L(d_{i1}, \ldots, d_{in})$.