Cryptographic Protocols
Spring 2018
Part 6

Multi-Party Computation: Goal

Specification Protocol

Sum Protocol

\[ r \in \mathbb{Z}_m \]
\[ y_1 = r + x_1 \]
\[ y_2 = y_1 + x_2 \]
\[ y_3 = y_2 + x_3 \]
\[ y_4 = y_3 + x_4 \]
\[ y_5 = y_4 + x_5 \]
\[ y_6 = y_5 + x_6 \]
\[ y_7 = y_6 + x_7 \]
\[ s = y_7 - r \]

Known Results

<table>
<thead>
<tr>
<th>Setting</th>
<th>Adv. Type</th>
<th>Condition</th>
<th>Literature</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryptographic</td>
<td>passive</td>
<td>( t &lt; n )</td>
<td>[GMW87]</td>
</tr>
<tr>
<td>cryptographic</td>
<td>active</td>
<td>( t &lt; n/2 )</td>
<td>[GMW87]</td>
</tr>
<tr>
<td>information-theoretic</td>
<td>passive</td>
<td>( t &lt; n/2 )</td>
<td>[BGW88, CCD88]</td>
</tr>
<tr>
<td>information-theoretic</td>
<td>active</td>
<td>( t &lt; n/3 )</td>
<td>[BGW88, CCD88]</td>
</tr>
<tr>
<td>i.-t. with broadcast</td>
<td>active</td>
<td>( t &lt; n/2 )</td>
<td>[RB89, Bea91]</td>
</tr>
</tbody>
</table>

Oblivious Transfer

1-2-OT

\[ s \]
\[ r \in \mathbb{Z}_m \]
\[ r = 0: s \]
\[ r = 1: i \]

1-i-OT

\[ s_1 \ldots s_k \]
\[ i \]
\[ s_i \]
1-2-OST based on RSA and DES

Sender
messages \( x_0, x_1 \)
gerate RSA-Keys
\( n_0, e_0, d_0 \) and \( n_1, e_1, d_1 \)
with \( n_0 \approx n_1 \)
selector \( b \in \{0, 1\} \)
generate RSA-Keys
\( n_0, e_0, d_0 \) and \( n_1, e_1, d_1 \)
\( k \) at random,
\( u = k^b \pmod{n_0} \)
\( y_0 = DES_{k_0}(s_0) \)
\( y_1 = DES_{k_1}(s_1) \)
\( y_0, y_1 \)

Receiver

Multi-Party Computation: Goal III

⇒
Specification Protocol

Multi-Party Computation: Goal IV

⇒
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

Operations

⇒
Protocol

Input:

⇒
Compute:

\( \oplus \)
\( \otimes \)

⇒
\( \oplus \)
\( \otimes \)

Output:

⇒

Sum Protocol III

\( x_1 \)
\( x_{11} x_{12} \ldots x_{1n} \)
\( x_2 \)
\( x_{21} x_{22} \ldots x_{2n} \)
\( \vdots \)
\( \vdots \)
\( x_n \)
\( x_{n1} x_{n2} \ldots x_{nn} \)
\( y_1 y_2 \ldots y_n \)
\( y = \sum_{i=1}^{n} y_i \)

Passive Protocol

Reconstruct Output

• \( a \) is shared by \( a_1, \ldots, a_n \).
• every \( P_j \) sends \( a_j \) to \( P_i \).
• \( P_i \) comp. \( a = L(a_1, \ldots, a_n) \).

Share input

• \( P_i \) has input \( x_i \).
• \( P_i \) selects \( r_1, \ldots, r_t \) at random.
• \( P_i \) comp. \( \left( \frac{x_1}{a_1} \right) = A \left( \frac{r_1}{r_i} \right) \).
• \( P_i \) sends \( a_j \) to every \( P_j \).

Addition and linear functions \( L \):

• \( a, b, \ldots \) shared by \( a_1, \ldots, a_n, b_1, \ldots, b_n \), etc.
• every \( P_i \) computes \( c_i = L(a_1, b_1, \ldots) \).

Multiplication

• \( a, b \) are shared by \( a_1, \ldots, a_n, b_1, \ldots, b_n \).
• every \( P_i \) computes \( d_i = a_i b_i \).
• every \( P_i \) shares \( d_i \rightarrow d_{i1}, \ldots, d_{in} \).
• every \( P_j \) computes \( c_j = L(d_{1j}, \ldots, d_{nj}) \).