Generic protocols for secure multi-party computation

Yao’s garbled circuits and the GMW protocol
What is secure multi-party computation (MPC)?
What is MPC?

**Setup:**

- In "parties" $P_1, P_2, \ldots, P_n$ with inputs $x_1, \ldots, x_n$

**Goal:**

- Evaluate a function $F$ on the inputs $x_1, \ldots, x_n$

**Constraint:**

- Keep the inputs "private"
- Learn nothing more about the inputs than what can be inferred through $F(x_1, x_2, \ldots, x_n)$. 
Ideal world

**Trusted third party:**

\[ y = F(x_1, x_2, \ldots, x_n) \]

**N.B.**

- Third party only shares back \( y \)
- No guarantee that inputs remain private:
  - \( y, x_1, \ldots, x_n \) can leak into about \( x_n \)

whole other field to address this issue for statistical applications
We want a protocol $\pi$ to compute $F(x_1, \ldots, x_n)$ without the need for a third party.

- Yao's garbled circuits
- GMW protocol
Primitives
Oblivious transfer

\[ \text{Sender } S_l \]
secrets \( s_1, s_2, \ldots, s_k \)

\[ \text{Receiver } R \]
chooses \( i \in \{ 1, \ldots, k \} \)

\[ \rightarrow s_i \]

\( R \) obtains \( s_i \) without learning about \( s_j, j \neq i \), and without \( S \) learning \( i \).
Oblivious transfer

Example:
You're taking an exam and the professor has hints for the different questions, but:
- they're only willing to give at most one hint to each student
- as a student for fair grading, you don't want them to know which hint (if any) you requested.

\[
\begin{align*}
\text{hint 1} & \quad \text{hint 2} & \quad \text{hint 3} \\
(\text{student choose 1 in secret})
\end{align*}
\]
Secret sharing

- secret $s$ shares $s_A$ $s_B$

- Individually, $s_A$ and $s_B$ give no information about $s$

- Together, $s_A$ and $s_B$ can be used to reconstruct $s$. 
Secret sharing

Example:

Sharing a secret bit $s \in \{0,1\}$:

1. Sample $k \in \{0,1\}$

2. Share $s_A = k \oplus s$ and $s_B = k$.

$\Rightarrow s = s_A \oplus s_B$
Secret sharing

\((t, n)\) - sharing:

- Share secret \(s\) in \(n\) parts such that
  - Any \(t\) parts can reconstruct \(s\)
  - No fewer than \(t\) parts give information about \(s\).
Example (s \in \mathbb{F}):

- Pick random \( p \in \mathbb{F}^{t-1} \) such that \( p(0) = s \).
- Consider the shares \( p(i), i = 1, 2, \ldots, n \).

\( \Rightarrow \) A polynomial of degree \( t-1 \) is uniquely defined by its values at \( t \) points.

\( \Rightarrow \) Some thinking is needed to show this is secure.
GMW protocol
2-party computation of logic gate

\[ x, y \in \{0, 1\} \]
2-party computation of logic gate

\[ x = x_1 \oplus x_2 \]

\[ y = y_1 \oplus y_2 \]
2-party computation of XOR

\[
\begin{align*}
x &= x_1 \oplus x_2 \\
y &= y_1 \oplus y_2
\end{align*}
\]
2-party computation of AND

\[ x \land y = x_1 \land y_1 \oplus x_2 \land y_2 \]
\[ \quad \oplus x_1 \land y_2 \oplus x_2 \land y_1 \]

\[ P_1 : \text{Keep } r \oplus x_1 \land y_1 \]
\[ P_2 : \text{Keep } f(x_1, y_2) \oplus x_2 \land y_2 \]
\[ P_2 : \text{obtain } f(x_1, y_2) \text{ using OT.} \]

secret sharing

\[ x = x_1 \oplus x_2 \]
\[ y = y_1 \oplus y_2 \]
Generalization to n party

secrets \( x, y \) in shares \( x_1, \ldots, x_n \) and \( y_1, \ldots, y_n \), \( x = \bigoplus_{i=1}^{n} x_i \), \( y = \bigoplus_{i=1}^{n} y_i \).

- \( x_i \oplus y_i \) is a share of \( x \oplus y \)

- \( x \wedge y = \left( \bigoplus_{i=1}^{n} x_i \wedge y_i \right) \oplus \left( \bigoplus_{i \neq j} x_i \wedge x_j \right) \)

*Note: this can be shared between pairs of parties.*
Yao’s garbled circuits
Yao’s garbled circuits

- Two-party computation $(f(x, y))$
- Compute facts represented as logic circuits.
Yao’s garbled circuits

$P_1$ generator

$P_2$ executor

$x_1$, $x_2$
Intuition

- Suppose \( x, y \in \{0, 1\} \).

\[ P_2 \text{ will:} \]
- compute \( f(x, 0), f(x, 1) \).

\[ P_2 \text{ will:} \]
- request \( f(v_x, v_y) \) from \( P_1 \) using OT.
Intuition

For more complex functions, we use composability:

\[ h(f(x_1, y_1), g(x_2, y_2)) \]

P (the generator) doesn't know any input to \( h \), so he has to compute all possible values in advance.
More general intuition

2. View $f(x,y)$ as a table

3. Encrypt:

$$e_{x_2}^{x_3} = Enc_{K_2}(e_{x_1}^{x_3})$$

1. Generate keys

$k_0$, $k_1$, $k_2$, $k_3$

$y, x_1, x_2, x_3$

Sort \((p_1, p_2, p_3, p_4)\) \(\in \mathbb{B}^4\).

$\text{Permute: } p_0 \in \{0,1\}, p_1 \in \{0,1,2,3\}, p_1 = 1 - p_2 - p_3 - p_4$
More general intuition

What you obtain:

<table>
<thead>
<tr>
<th>$v_x$</th>
<th>$v_y$</th>
<th>$f(v_x, v_y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$f(0,0)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$f(0,1)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$f(1,0)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$f(1,1)$</td>
</tr>
</tbody>
</table>

random sorting

key

$\text{encryption key}$

$\text{cyphertext}$

$\begin{array}{c|c|c}
\text{random sorting} & \text{encryption key} & \text{cyphertext} \\
\hline
p_x^o & p_y^o & k_x^o \quad k_y^o \\
p_x^o & p_y^i & k_x^o \quad k_y^i \\
p_x^i & p_y^o & k_x^i \quad k_y^o \\
p_x^i & p_y^i & k_x^i \quad k_y^i \\
\end{array}$

$\begin{array}{c|c|c}
\text{random sorting} & \text{encryption key} & \text{cyphertext} \\
\hline
e_{0,0} & e_{0,0} & e_{0,0} \\
e_{0,1} & e_{0,1} & e_{0,1} \\
e_{1,0} & e_{1,0} & e_{1,1} \\
e_{1,1} & e_{1,1} & e_{1,1} \\
\end{array}$
More general intuition

- $P_1$ sends the sorted $(p_x, p_y), (e_x, y)$, $x, y \in \sigma(a,b)$ to $P_2$.
- $P_1$ can send $K_x, p_x$ corresponding to its input value.
- $P_2$ can request $K_y, p_y$ using OT.
- $P_2$ can decrypt: $\text{Dec}_{K_x, K_y}(e_x, y)$. 

Visually

\[ (\rho_x, K_x)_{x \in \{0, 1\}} \]
\[ (\rho_y, K_y)_{y \in \{0, 1\}} \]

\[ (e_{u_x, v_y})_{u_x, v_y \in \{0, 1\}} \]

\[ \text{garbled table} \]

\[ \text{wire labels} \]

\[ x \]
\[ y \]

\[ z \]

\[ this \ is \ composable \]
Hide intermediate results
Composability
Security

- $P_1$ shares the garbled tables and $(p_{x}^{v_x}, k_x^{v_x})$ corresponding to its input $v_x$
- $P_2$ obtains $(p_{y}^{v_y}, k_y^{v_y})$ for its input $v_y$ using OT
- $P_2$ executes the computation
- $P_4$ can decrypt the result.