Secure Computation: Why, How and When

Mariana Raykova
Yale University
Predictive Model

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood Count</th>
<th>Heart Conditions</th>
<th>Digestive Track</th>
<th>...</th>
<th>Medicine Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3.9</td>
<td>10.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>4.5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4.3</td>
<td>5.3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Given samples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$
- Learn a function $f$ such that $f(\mathbf{x}_i) = y_i$
Linear Regression

- Given samples \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
  \(\quad x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\)
- Learn a function \(f\) such that \(f(x_i) = y_i\)

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood Count</th>
<th>Heart Conditions</th>
<th>Digestive Track</th>
<th>...</th>
<th>Medicine Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBC</td>
<td>WBC</td>
<td>Murmur</td>
<td>Arrhythmia</td>
<td>Inflammation</td>
</tr>
<tr>
<td>A</td>
<td>3.9</td>
<td>10.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>4.5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4.3</td>
<td>5.3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

\(f\) is well approximated by a linear map

\[ y_i \approx \theta^T x_i \]
Distributed Data

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood Count</th>
<th>Heart Conditions</th>
<th>Digestive Track</th>
<th>...</th>
<th>Medicine Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBC</td>
<td>WBC</td>
<td>...</td>
<td>Murmur</td>
<td>Arrhythmia</td>
</tr>
<tr>
<td>A</td>
<td>3.9</td>
<td>10.0</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>4.5</td>
<td>...</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>11</td>
<td>...</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4.3</td>
<td>5.3</td>
<td>...</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Shared database** - \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) do not belong to the same party
### Horizontally Partitioned Database

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood Count</th>
<th>Heart Conditions</th>
<th>Digestive Track</th>
<th>...</th>
<th>Medicine Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RBC</td>
<td>WBC</td>
<td>Murmur</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3.9</td>
<td>10.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>4.5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4.3</td>
<td>5.3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**• Different rows belong to different parties**
  
  - E.g., each patient has their own information
**Vertically Partitioned Database**

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood Count</th>
<th>Heart Conditions</th>
<th>Digestive Track</th>
<th>Medicine Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RBC</td>
<td>WBC</td>
<td>...</td>
<td>Murmur</td>
</tr>
<tr>
<td>A</td>
<td>3.9</td>
<td>10.0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5.0</td>
<td>4.5</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2.5</td>
<td>11</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4.3</td>
<td>5.3</td>
<td>...</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Different columns belong to different parties**
  - E.g., different specialized hospitals have different parts of the information for all patients
Can the parties holding the distributed data construct the predictive model on the whole database **while** protecting the privacy of their inputs?

- Without sending their own data to other parties
- Without revealing more about their own inputs
Secure Computation

Alice and Bob want to compute $F(X,Y)$ without revealing their inputs.
Secure Computation

Security: the parties cannot learn more than what is revealed by the result.
Secure Multiparty Computation (MPC)

Security: the parties cannot learn more than what is revealed by the result
Applications

- Auctions:
  - inputs: bids; output: winner, price to pay
  - Sugar beet auction in Denmark, 2008
  - Energy trade auctions
What Does and Does Not MPC Guarantee?

Guarantee: The computation does not reveal more than what the output reveals.

No Guarantee: How much does the output reveal.
Security

Real World

\[ F(X_1, \ldots, X_5) \]

Ideal World

\[ F(X_1, \ldots, X_5) \sim \approx \]

Simulator
Adversarial Models

Adversary behavior:
• **Semi-honest** – corrupt parties follow the MPC protocol
• **Malicious** – corrupt parties deviate arbitrarily from the MPC protocol

Party corruption:
• **Static** – corrupted parties are chosen before the start of the MPC protocol execution
• **Adaptive** – parties can be corrupted during the execution
What Can We Compute Securely?

- We can compute securely any function!
  
  • [Yao82, GMW87, CDv88, BG89, BG90, Cha90, Bea92, CvT95, CFGN96, Gol97, HM97, CDM97, FHM98, BW98, KOR98, GRR98, FvHM99, CDD+99, HMP00, CDM00, SR00, CDD00, HM00, Kil00, FGMO01, HM01, CDN01, Lin01, FGMv02, Mau02, GIKR02, PSR02, NNP03, FHHW03, KOS03, CFIK03, Lin03c, DN03, MOR03, CKL03, Pin03, PR03, NMQO+03, Lin03b, Lin03a, Lin03d, FWW04, FHW04, Pas04, IK04, HT04, ST04, KO04, MP04, ZLX05, CDG+05, HNP05, FGMO05, GL05, HN05, DI05, JL05, Kol05, WW05, vAHL05, LT06, CC06, DFK+06, BTH06, NK06, IKLP06, DI06, FFP+06, ADGH06, Dam06, MF06, CKL06, DPSW07, Kat07b, CGOS07, HIK07, DN07, Pen07, NO07, Kat07a, IKOS07, BMQU07, HK07, LP07, Woo07, BDNP08, QT08, PR08, HNP08, GKO8, GMS08, SYTO8, DIK+08, PCR08, KS08, Lin08, LPS08, GHKL08, CEMY09, GP09, GK09, MPR09, ZHM09, AKL+09, Tof09, BCD+09, DGKN09, DNW09, Lin09b, PSSW09, Lin09a, CLS09, LP09, Unr10, DO10, IKP10, DIK10, GK10, ..........]
Computation Over Circuits

Boolean Circuits

• Yao Gabled Circuits

Arithmetic Circuits

• BGW Construction
  ○ Ben-Or, Goldwasser, Widgerson
Yao Garbled Circuits

Two Party Computation
Circuit Evaluation
Circuit Evaluation

1 0 0 0 0 1 1
AND OR OR AND
0 0 1 1
OR AND
0 1
AND
0
F
Evaluation

<table>
<thead>
<tr>
<th>In1</th>
<th>In2</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

AND

0 1 0

0
Yao Garbled Evaluation

\[
\text{ENC}_k(m) = m \oplus k
\]

<table>
<thead>
<tr>
<th>AND</th>
<th>0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>In1</td>
<td>In2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Garbled Evaluation

\[ K_{00} \]
\[ K_{11} \]
\[ \bot \leftarrow \text{DEC}_{k_{00}} \text{DEC}_{k_{10}} (k_{20}) \]
\[ K_{20} \leftarrow \text{DEC}_{k_{00}} \text{DEC}_{k_{11}} (k_{20}) \]
\[ \bot \leftarrow \text{DEC}_{k_{01}} \text{DEC}_{k_{10}} (k_{20}) \]
\[ \bot \leftarrow \text{ENC}_{k_{01}} \text{ENC}_{k_{11}} (k_{21}) \]

**AND**

<table>
<thead>
<tr>
<th>In1</th>
<th>In2</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Secure Computation

\[ F (X_{\text{alice}}, Y_{\text{bob}}) \]
Oblivious Transfer (OT)

Sender
Inputs: \(m_0, m_1\)

Receiver
Inputs: \(b\)

Output: \(\bot\)

Output: \(m_b\)

For each inputs wire corresponding to evaluator’s input execute OT

\[\overrightarrow{m_0 \lor m_1} \quad \overleftarrow{b} \quad \overrightarrow{m_b}\]
### The Evolution Of Garbled Circuits

<table>
<thead>
<tr>
<th></th>
<th>Size (x sec.param)</th>
<th>Garble cost</th>
<th>Eval cost</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AND</td>
<td>XOR</td>
<td>AND</td>
<td>XOR</td>
</tr>
<tr>
<td>Classical [Yao86]</td>
<td>large</td>
<td>8</td>
<td>5</td>
<td>PKE</td>
</tr>
<tr>
<td>P&amp;P [BMR90]</td>
<td>4</td>
<td>4/8</td>
<td>1/2</td>
<td>hash/PRF</td>
</tr>
<tr>
<td>GRR3 [NPS99]</td>
<td>3</td>
<td>4/8</td>
<td>1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>Free XOR [KS08]</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>circ. hash</td>
</tr>
<tr>
<td>GRR2 [PSSW09]</td>
<td>2</td>
<td>4/8</td>
<td>1/2</td>
<td>PRF/hash</td>
</tr>
<tr>
<td>FlexOR [KMR14]</td>
<td>2</td>
<td>{0,1,2}</td>
<td>1</td>
<td>circ. symm</td>
</tr>
<tr>
<td>HalfGates [ZRE15]</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>circ. hash</td>
</tr>
</tbody>
</table>

Threshold gates, garbling arithmetic operations [BMR16]

- Asymptotic and concrete improvements
BGW Protocol

Multi Party Computation for Arithmetic Circuits
Shamir’s Secret Sharing

\textbf{t-out-of-n} sharing: random degree \( t \) polynomial

\( t \) shares reveal nothing about the secret

\( t+1 \) shares interpolate the secret
Multi-Party Computation

\[ f_x(0) = x \]
\[ f_x(5) = a_1 \]
\[ f_x(10) = a_2 \]
\[ f_x(15) = a_3 \]
\[ f_x(20) = a_4 \]

\[ f_y(0) = y \]
\[ f_y(5) = b_1 \]
\[ f_y(10) = b_2 \]
\[ f_y(15) = b_3 \]
\[ f_y(20) = b_4 \]

\[ f_z(0) = z \]
\[ f_z(5) = c_1 \]
\[ f_z(10) = c_2 \]
\[ f_z(15) = c_3 \]
\[ f_z(20) = c_4 \]

\[ f_w(0) = w \]
\[ f_w(5) = d_1 \]
\[ f_w(10) = d_2 \]
\[ f_w(15) = d_3 \]
\[ f_w(20) = d_4 \]

\[ f_{x+y}(0) = x+y \]
\[ f_{x+y}(5) = a_1 + b_1 \]
\[ f_{x+y}(10) = a_2 + b_2 \]
\[ f_{x+y}(15) = a_3 + b_3 \]
\[ f_{x+y}(20) = a_4 + b_4 \]

\[ f_{zw}(0) = zw \]
\[ f_{zw}(5) = c_1 d_1 \]
\[ f_{zw}(10) = c_2 d_2 \]
\[ f_{zw}(15) = c_3 d_3 \]
\[ f_{zw}(20) = c_4 d_4 \]

\[ F(x, y, z, w) = (x+y)zw+zw+w \]
Multi-Party Computation

\[ f_x(0) = x \]
\[ f_y(20) = b_4 \]

\[ f_x(5) = a_1 \]
\[ f_x(10) = a_2 \]
\[ f_x(15) = a_3 \]
Multi-Party Computation

\[ f_z(20) = c_4 \]
\[ f_z(5) = c_1 \]
\[ f_z(0) = z \]
\[ f_z(10) = c_2 \]
\[ f_x(15) = c_3 \]
Multi-Party Computation

\[ f_y(0) = y \]

\[ f_w(0) = w \]
Multi-Party Computation

$F(a_1, b_1, c_1, d_1) \quad \text{ Shares of the output. Are they enough to reconstruct? } \quad F(a_2, b_2, c_2, d_2)$

$F(a_3, b_3, c_3, d_3)$

$a_4, b_4, c_4, d_4$

$a_3, b_3, c_3, d_3$
How Many Shares?

• If we allow $t$ corrupt parties, we need polynomials of degree $t$
  o The secret can be reconstructed by at least $t+1$ parties

• Addition gates:
  o Output shares lie on a polynomial of degree $t$

• Multiplication gates:
  o Output shares lie on a polynomial of degree $2t$
  o We need at least $2t+1$ parties to reconstruct the secret

• Does the degree increase exponentially with the multiplicative depth of the circuit?
  o “Luckily” not – we can reduce the degree after each multiplication gate
  o For any $n>2t+1$ and points $\alpha_1, \ldots, \alpha_n$, there exists an $n \times n$ matrix $A$ such that for all polynomial $p(x)$ of degree $2t$
    \[
    A \left( p(\alpha_1), \ldots, p(\alpha_n) \right) = \left( p'(\alpha_1), \ldots, p'(\alpha_n) \right)
    \]
    where
    - $p'(x)$ is of degree $t$
    - $p'(x) = p(x)$
How to Reduce the Degree?

Vandermonde matrix

\[
p(\alpha_1) \\
p(\alpha_2) \\
\vdots \\
p(\alpha_n)
\]

\[
\begin{bmatrix}
1 & \alpha_1 & \ldots & \alpha_1^{n-1} \\
1 & \alpha_2 & \ldots & \alpha_2^{n-1} \\
\vdots \\
1 & \alpha_n & \ldots & \alpha_n^{n-1}
\end{bmatrix}
\]

\[
p_0 \\
p_t \\
0
\]

\[
\begin{bmatrix}
1 & \alpha_1 & \ldots & \alpha_1^{n-1} \\
1 & \alpha_2 & \ldots & \alpha_2^{n-1} \\
\vdots \\
1 & \alpha_n & \ldots & \alpha_n^{n-1}
\end{bmatrix}
\]

\[
p_0 \\
p_2t \\
0
\]

\[
\begin{bmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots \\
0 & 0 & \ldots & 1 \\
0 & 0 & \ldots & 0
\end{bmatrix}
\]

\[
p_0 \\
p_t \\
0
\]

\[
\begin{bmatrix}
1 & \alpha_1 & \ldots & \alpha_1^{n-1} \\
1 & \alpha_2 & \ldots & \alpha_2^{n-1} \\
\vdots \\
1 & \alpha_n & \ldots & \alpha_n^{n-1}
\end{bmatrix}
\]

\[
\begin{bmatrix}
-1 \\
p'(\alpha_1) \\
p'(\alpha_n)
\end{bmatrix}
\]

\[
A
\]
Multi-Party Computation

- BGW security guarantees for n party computation
  - Semi-honest model: up to $n/2$ corrupt parties
  - Malicious model: up to $n/3$ corrupt parties
  - Information theoretic/perfect security

- Security against arbitrary number (up to $n-1$) of corrupt parties
  - Computational security (relies on computational assumptions)
  - Constructions:
    - GMW Protocol [GMW87] (Goldreich-Micali-Wigderson)
    - Preprocessing model: SPZD [DPSZ12], SPZD-BMR [LPSY15], BMR-SHE [LSS16], Mascot [KOS16]
Computation Over Circuits

- **Boolean Circuits**
- **Yao Gabled Circuits**

- **Arithmetic Circuits**
- **BGW Construction**
  - Ben-Or, Goldwasser, Widgerson
How Efficient is Computation with Circuits?

• Linear in the circuit size!
Binary Search

Binary search has logarithmic complexity in plaintext computation.
Is MPC inherently linear?

Yes, if you do not touch some part of the data, you reveal it is not used in the computation.

No, in the amortized setting.
Random Access Machine (RAM)

- LOAD #5
- STORE 15
- LOAD #0
- EQUAL 15
- JUMP #6
- HALT
- ADD #1
- JUMP #3
RAM Computation

While
state ≠ stop

Access memory
- fetch next program instruction
- read/write data

Computation:
- update state
- compute next memory instruction
Binary Search RAM

While
item not found and non-empty search range

Access memory
Read data from address $p$

Computation
check for match, compute next address $p$ to access
Secure Computation for RAMs

- Binary Search

- Oblivious RAM [GO96]
  - access pattern hiding

- WHOLE DATABASE N
  - Log N steps
  - Address
  - Constant size
  - Read from memory
**ORAM Properties**

- **Access pattern hiding**
  - The physical accesses in memory for any two query sequences of equal length are indistinguishable.

- **Efficiency** - random access (logarithmic)
  - Note: trivial solution is to read the whole memory at each access. Very expensive!

- **ORAM Initialization** – one time linear computation

- **Constructions:**
  - **Hierarchical-based:** [GO96], [KLO12]
  - **Tree-based:** Tree ORAM [SCSL11], Path ORAM [SDSCFRYD13], Circuit ORAM [WCS15]

Example:
read 1, read 1, read 1
write 3, read 1, read 5

Logarithmic number of subqueries for memory part of constant size
MPC for RAMs enables secure computation with sublinear complexity in the amortized setting!
What Does and Does Not MPC Guarantee?

**Guarantee:** The computation does not reveal more than what the output reveals.

**Secure Computation for Approximations:**
An approximation may reveal more than the exact output of the computation. One needs to argue that such leakage does not exist. [FIMNSW06]

**No Guarantee:**
How much does the output reveal.
The Impact of Cryptography

Practical Impact

80’s – public key cryptography

2016

Practical MPC

Developments in cryptography