Secure Two-Party Computation

Anupam Datta
CMU
Fall 2014
Secure Two-Party Computation

- Bob’s Genome: ACTG...
- Markers (~1000): [0,1, ..., 0]

- Alice’s Genome: ACTG...
- Markers (~1000): [0, 0, ..., 1]

Can Alice and Bob compute a function of their private data, without exposing anything about their data besides the result?

$$x = f(g_A, g_B)$$
Roadmap

- Yao’s Classic Garbled Circuits
- Recent advances in practical secure two party computations
Yao’s Protocol

◆ Compute any function securely
  • ... in the semi-honest model
◆ First, convert the function into a boolean circuit

Truth table:

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

OR

Truth table:

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
1: Pick Random Keys For Each Wire

◆ Next, evaluate one gate securely
  • Later, generalize to the entire circuit
◆ Alice picks two random keys for each wire
  • One key corresponds to “0”, the other to “1”
  • 6 keys in total for a gate with 2 input wires
2: Encrypt Truth Table

Alice encrypts each row of the truth table by encrypting the output-wire key with the corresponding pair of input-wire keys.

![Diagram of an AND gate with keys]

Original truth table:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</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>

Encrypted truth table:

- $E_{k_0x}(E_{k_0y}(k_{0z}))$
- $E_{k_0x}(E_{k_1y}(k_{0z}))$
- $E_{k_1x}(E_{k_0y}(k_{0z}))$
- $E_{k_1x}(E_{k_1y}(k_{1z}))$
3: Send Garbled Truth Table

- Alice randomly permutes ("garbles") encrypted truth table and sends it to Bob

Garbled truth table:

$$E_{k_0x}(E_{k_0y}(k_{0z}))$$
$$E_{k_0x}(E_{k_1y}(k_{0z}))$$
$$E_{k_1x}(E_{k_0y}(k_{0z}))$$
$$E_{k_1x}(E_{k_1y}(k_{1z}))$$

Does not know which row of garbled table corresponds to which row of original table.
4: Send Keys For Alice’s Inputs

Alice sends the key corresponding to her input bit
- Keys are random, so Bob does not learn what this bit is

Garbled truth table:
- $E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
- $E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
- $E_{k_{1x}}(E_{k_{0y}}(k_{1z}))$
- $E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
- $E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

Learns $K_{b'x}$ where $b'$ is Alice’s input bit, but not $b'$ (why?)

If Alice’s bit is 1, she simply sends $k_{1x}$ to Bob; if 0, she sends $k_{0x}$
5: Use OT on Keys for Bob’s Input

- Alice and Bob run oblivious transfer protocol
  - Alice’s input is the two keys corresponding to Bob’s wire
  - Bob’s input into OT is simply his 1-bit input on that wire

Garbled truth table:

- $E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
- $E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
- $E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
- $E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

Alice knows $K_{b'x}$ where $b'$ is Alice’s input bit and $K_{by}$ where $b$ is his own input bit.

Bob learns $k_{by}$.

What does Alice learn?
6: Evaluate Garbled Gate

Using the two keys that he learned, Bob decrypts exactly one of the output-wire keys

- Bob does not learn if this key corresponds to 0 or 1
  - Why is this important?

Alice

Bob

k_{0z}, k_{1z}

k_{0x}, k_{1x}

k_{0y}, k_{1y}

AND

Suppose b' = 0, b = 1
This is the only row Bob can decrypt.
He learns k_{0z}

Knows K_{b'x} where b' is Alice's input bit and K_{by} where b is his own input bit

Garbled truth table:
In this way, Bob evaluates entire garbled circuit

- For each wire in the circuit, Bob learns only one key
- It corresponds to 0 or 1 (Bob does not know which)
  - Therefore, Bob does not learn intermediate values (why?)

Bob tells Alice the key for the final output wire and she tells him if it corresponds to 0 or 1

- Bob does not tell her intermediate wire keys (why?)
Brief Discussion of Yao’s Protocol

- Function must be converted into a circuit
  - For many functions, circuit will be huge
- If m gates in the circuit and n inputs, then need 4m encryptions and n oblivious transfers
  - Oblivious transfers for all inputs can be done in parallel
- Yao’s construction gives a constant-round protocol for secure computation of any function in the semi-honest model
  - Number of rounds does not depend on the number of inputs or the size of the circuit!
Acknowledgments

◆ Slides 4-12 from Vitaly Shmatikov