Secure Computation & Yao’s Protocol

Mike Rosulek

crypt@b-it 2018
Roadmap

1. **Secure computation**: Concepts & definitions

2. **Yaos’ protocol**: semi-honest secure computation for boolean circuits
Secure computation

Premise:
- Mutually distrusting parties, each with a private input

Security guarantees:
- Privacy ("learn no more than" prescribed output)
- Input independence
- Output consistency, etc.

.. even if some parties cheat, collude!
Secure computation

Premise:
- Mutually distrusting parties, each with a private input
- Learn the result of agreed-upon computation
- Ex: election, auction, etc.

\[ f(x_1, x_2, x_3, x_4, x_5) \]
Secure computation

Premise:
- Mutually distrusting parties, each with a private input
- Learn the result of agreed-upon computation
- Ex: election, auction, etc.

Security guarantees:
- Privacy ("learn no more than" prescribed output)
- Input independence
- Output consistency, etc..

\[ f(x_1, x_2, x_3, x_4, x_5) \]
Secure computation

Premise:
- Mutually distrusting parties, each with a private input
- Learn the result of agreed-upon computation
- Ex: election, auction, etc.

Security guarantees:
- Privacy ("learn no more than" prescribed output)
- Input independence
- Output consistency, etc..

..even if some parties cheat, collude!

\[ f(x_1, x_2, x_3, x_4, x_5) \]
Examples: Sugar Beets

- Farmers make bids ("at price $X$, I will produce $Y$ amount")
- Purchaser bids ("at price $X$, I will buy $Y$ amount")
Examples: Sugar Beets

- Farmers make bids ("at price $X$, I will produce $Y$ amount")
- Purchaser bids ("at price $X$, I will buy $Y$ amount")
- **Market clearing price** (MCP): price at which total supply = demand
Examples: Sugar Beets

- Farmers make bids ("at price $X$, I will produce $Y$ amount")
- Purchaser bids ("at price $X$, I will buy $Y$ amount")
- Market clearing price (MCP): price at which total supply = demand
- 2009: MCP (+ bids at that price) computed via secure computation
Examples: Ad conversion

<table>
<thead>
<tr>
<th>Ad impressions</th>
<th>In-store purchases</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:alice@gmail.com">alice@gmail.com</a></td>
<td><a href="mailto:albert@gmail.com">albert@gmail.com</a> $80K</td>
</tr>
<tr>
<td><a href="mailto:bob@gmail.com">bob@gmail.com</a></td>
<td><a href="mailto:bob@gmail.com">bob@gmail.com</a> $160K</td>
</tr>
<tr>
<td><a href="mailto:charlie@gmail.com">charlie@gmail.com</a></td>
<td><a href="mailto:caroline@gmail.com">caroline@gmail.com</a> $99K</td>
</tr>
<tr>
<td><a href="mailto:dianne@gmail.com">dianne@gmail.com</a></td>
<td><a href="mailto:edwin@gmail.com">edwin@gmail.com</a> $99K</td>
</tr>
<tr>
<td><a href="mailto:edwin@gmail.com">edwin@gmail.com</a></td>
<td><a href="mailto:felipe@gmail.com">felipe@gmail.com</a> $85K</td>
</tr>
<tr>
<td><a href="mailto:frank@gmail.com">frank@gmail.com</a></td>
<td><a href="mailto:frank@gmail.com">frank@gmail.com</a> $77K</td>
</tr>
<tr>
<td><a href="mailto:gina@gmail.com">gina@gmail.com</a></td>
<td><a href="mailto:hilda@gmail.com">hilda@gmail.com</a> $113K</td>
</tr>
</tbody>
</table>
Examples: Ad conversion

```
SELECT SUM(amount)
FROM ads, purchases
WHERE ads.email = purchases.email
```

▶ Computed with secure computation by Google and its customers
How Boston Is Trying to Close the Gender Pay Gap

Through pay-negotiation workshops and partnerships with more than 100 companies, the city is trying to help female workers match the salaries of male counterparts.

DATA SUBMISSION PROCESS:
Part of the commitment employers make when signing the Boston 100% Talent Compact is to anonymously report employee data to the BWWC biennially. The Software & Application Innovation Lab at Boston University's Rafik B. Hariri Institute of Computing and Computational Science & Engineering, the BWWC's data partner, developed a completely confidential reporting system from which anonymous data from multiple independent sources can be analyzed in the aggregate.

During the submission process, Compact signers submit their wage data in the aggregate form over a unique, web-based software program that employs encryption using a technique known as secure multi-party computation. During this process, individual compensation data never leaves each organization's server. The BWWC then receives aggregate data unconnected to any firm.
What does it mean to “securely” compute $f$?
Security laundry list

- What if adversary learns more than $f(x, y)$?
- What if adversary learns $f(x, y)$ but then prevents honest party from learning it too?
- What if adversary forces several parties to have inconsistent outputs?
- What if adversary’s choice of input depends on honest party’s input?
- What if . . .
Defining security: ideal world

What can a corrupt party do in this ideal world?

▶ Choose any input $y$ (independent of $x$)
▶ Learn only $f(x; y)$, and nothing more
▶ Cause honest party to learn $f(x; y)$
Defining security: ideal world

What can a corrupt party do in this ideal world?

▶ Choose any input $y$ (independent of $x$)
▶ Learn only $f(x; y)$, and nothing more
▶ Cause honest party to learn $f(x; y)$
Defining security: ideal world

What can a corrupt party do in this ideal world?

▶ Choose any input $y$ (independent of $x$)
▶ Learn only $f(x; y)$, and nothing more
▶ Cause honest party to learn $f(x; y)$
Defining security: ideal world

What can a corrupt party do in this ideal world?
Defining security: ideal world

What can a corrupt party do in this **ideal world**?

- Choose any input \( y \) (independent of \( x \))
- Learn only \( f(x, y) \), and nothing more
- Cause honest party to learn \( f(x, y) \)
Security goal: real protocol interaction is as secure as the ideal-world interaction
Real-ideal paradigm \[\text{[GoldwasserMicali84]}\]

Security goal: real protocol interaction is \textbf{as secure as} the ideal-world interaction

For every “attack” against real protocol, there is a way to achieve “same effect” in ideal world
Real-ideal paradigm

What is the “effect” of a generic attack?
Real-ideal paradigm

What is the “effect” of a generic attack?

- Something the adversary learns / can compute about honest party
Real-ideal paradigm

What is the “effect” of a generic attack?

- Something the adversary learns / can compute about honest party
- Some influence on honest party’s output
Defining security

**Security definition:** For every real-world adversary $\mathcal{A}$
Defining security

Security definition: For every real-world adversary $A$, there exists an ideal adversary $A'$. 
Defining security

**Security definition**: For every real-world adversary $\mathcal{A}$, there exists an ideal adversary $\mathcal{A}'$ s.t. joint distribution $(\text{HonestOutput}, \text{AdvOutput})$ is indistinguishable
Defining security

Security definition: For every real-world adversary $\mathcal{A}$, there exists an ideal adversary $\mathcal{A}'$ s.t. joint distribution $(\text{HonestOutput}, \text{AdvOutput})$ is indistinguishable.

WLOG: $\exists$ simulator that simulates real-world interaction in ideal world.
Defining security

Role of simulator:

1. Send protocol messages that look like they came from honest party

2. **Extract** an $f$-input by examining adversary’s protocol messages
Defining security

Role of simulator:

1. Send protocol messages that look like they came from honest party
   ▶ Demonstrates that honest party’s messages leak no more than $f(x, y)$

2. **Extract** an $f$-input by examining adversary’s protocol messages
   ▶ “Explains” the effect on honest party’s output in terms of ideal world
Semi-Honest security

Special case: security against semi-honest (passive, honest-but-curious) adversary:

- Adversary assumed to follow the protocol on a given input
- Adversary may try to learn information based on what it sees
- No need to extract, only simulate transcript given ideal input+output
Disclaimer

Security definition here is **greatly oversimplified**

Universally Composable Security:
A New Paradigm for Cryptographic Protocols

Ran Canetti

July 16, 2013

Abstract

We present a general framework for representing cryptographic protocols and analyzing their security. The framework allows specifying the security requirements of practically any cryptographic task in a unified and systematic way. Furthermore, in this framework the security of protocols is preserved under a general protocol composition operation, called universal composition.

The proposed framework with its security-preserving composition operation allows for modular design and analysis of complex cryptographic protocols from relatively simple building blocks. Moreover, within this framework, protocols are guaranteed to maintain their security in any context, even in the presence of an unbounded number of arbitrary protocol instances.

(87-page security definition)
Secure computation: Concepts & definitions

Yao’s protocol: semi-honest secure computation for boolean circuits
Warm-up: garbled truth table

Alice does the following:

1. Write truth table of function $f$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$f(1, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$f(1, 1)$</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>$f(1, 2)$</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>$f(1, 3)$</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>$f(1, 4)$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$f(2, 1)$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$f(2, 2)$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$f(2, 3)$</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>$f(2, 4)$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$f(3, 1)$</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>$f(3, 2)$</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$f(3, 3)$</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>$f(3, 4)$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$f(4, 1)$</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>$f(4, 2)$</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>$f(4, 3)$</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>$f(4, 4)$</td>
</tr>
</tbody>
</table>
Warm-up: garbled truth table

Alice does the following:

1. Write truth table of function \( f \)
2. For each possible input, choose random cryptographic key

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B₁</td>
</tr>
<tr>
<td>2</td>
<td>B₂</td>
</tr>
<tr>
<td>3</td>
<td>B₃</td>
</tr>
<tr>
<td>4</td>
<td>B₄</td>
</tr>
<tr>
<td>5</td>
<td>B₁</td>
</tr>
<tr>
<td>6</td>
<td>B₂</td>
</tr>
<tr>
<td>7</td>
<td>B₃</td>
</tr>
<tr>
<td>8</td>
<td>B₄</td>
</tr>
<tr>
<td>9</td>
<td>B₁</td>
</tr>
<tr>
<td>10</td>
<td>B₂</td>
</tr>
<tr>
<td>11</td>
<td>B₃</td>
</tr>
<tr>
<td>12</td>
<td>B₄</td>
</tr>
<tr>
<td>13</td>
<td>B₁</td>
</tr>
<tr>
<td>14</td>
<td>B₂</td>
</tr>
<tr>
<td>15</td>
<td>B₃</td>
</tr>
<tr>
<td>16</td>
<td>B₄</td>
</tr>
</tbody>
</table>
Warm-up: garbled truth table

Alice does the following:

1. Write truth table of function $f$
2. For each possible input, choose random **cryptographic key**
3. Encrypt each output with corresponding keys

\[
\begin{align*}
\mathbb{E}_{A_1, B_1}(f(1, 1)) & \\
\mathbb{E}_{A_1, B_2}(f(1, 2)) & \\
\mathbb{E}_{A_1, B_3}(f(1, 3)) & \\
\mathbb{E}_{A_1, B_4}(f(1, 4)) & \\
\mathbb{E}_{A_2, B_1}(f(2, 1)) & \\
\mathbb{E}_{A_2, B_2}(f(2, 2)) & \\
\mathbb{E}_{A_2, B_3}(f(2, 3)) & \\
\mathbb{E}_{A_2, B_4}(f(2, 4)) & \\
\mathbb{E}_{A_3, B_1}(f(3, 1)) & \\
\mathbb{E}_{A_3, B_2}(f(3, 2)) & \\
\mathbb{E}_{A_3, B_3}(f(3, 3)) & \\
\mathbb{E}_{A_3, B_4}(f(3, 4)) & \\
\mathbb{E}_{A_4, B_1}(f(4, 1)) & \\
\mathbb{E}_{A_4, B_2}(f(4, 2)) & \\
\mathbb{E}_{A_4, B_3}(f(4, 3)) & \\
\mathbb{E}_{A_4, B_4}(f(4, 4)) & \\
\end{align*}
\]
Warm-up: garbled truth table

Alice does the following:

1. Write truth table of function $f$
2. For each possible input, choose random cryptographic key
3. Encrypt each output with corresponding keys
4. Randomly permute ciphertexts, send to Bob
Warm-up: garbled truth table

Alice does the following:
1. Write truth table of function $f$
2. For each possible input, choose random cryptographic key
3. Encrypt each output with corresponding keys
4. Randomly permute ciphertexts, send to Bob

Somehow Bob obtains “correct” $A_x, B_y$
Warm-up: garbled truth table

Alice does the following:

1. Write truth table of function \( f \)
2. For each possible input, choose random cryptographic key
3. Encrypt each output with corresponding keys
4. Randomly permute ciphertexts, send to Bob

Somehow Bob obtains “correct” \( A_x, B_y \)

Through trial decryption, Bob learns only \( f(x, y) \)
Suffices to show that Bob’s view in the protocol can be simulated given just Bob’s ideal input/output.

**Bob’s view (real):**

\[
\begin{align*}
& A_4, B_2 \\
& E_{A_3, B_4}(f(3, 4)) \\
& E_{A_4, B_3}(f(4, 3)) \\
& E_{A_3, B_3}(f(3, 3)) \\
& E_{A_2, B_3}(f(2, 3)) \\
& E_{A_4, B_2}(f(4, 2)) \\
& E_{A_2, B_4}(f(2, 4)) \\
& E_{A_4, B_4}(f(4, 4)) \\
& E_{A_1, B_4}(f(1, 4)) \\
& E_{A_2, B_2}(f(2, 2)) \\
& E_{A_1, B_2}(f(1, 2)) \\
& \vdots \\
\end{align*}
\]
Security of warm-up protocol

Suffices to show that Bob’s view in the protocol can be simulated given just Bob’s ideal input/output.

Bob’s view (real): \( \approx \) Simulated view:

\[
\begin{array}{c}
A_4, B_2 \\
\mathbb{E}_{A_3, B_4}(f(3, 4)) \\
\mathbb{E}_{A_4, B_3}(f(4, 3)) \\
\mathbb{E}_{A_3, B_3}(f(3, 3)) \\
\mathbb{E}_{A_2, B_3}(f(2, 3)) \\
\mathbb{E}_{A_4, B_3}(f(2, 3)) \\
\mathbb{E}_{A_2, B_2}(f(4, 2)) \\
\mathbb{E}_{A_4, B_2}(f(4, 2)) \\
\mathbb{E}_{A_2, B_4}(f(2, 4)) \\
\mathbb{E}_{A_4, B_4}(f(4, 4)) \\
\mathbb{E}_{A_1, B_4}(f(1, 4)) \\
\mathbb{E}_{A_2, B_2}(f(2, 2)) \\
\mathbb{E}_{A_1, B_2}(f(1, 2)) \\
\vdots
\end{array}
\]

\[
\begin{array}{c}
A^*, B^* \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A^*, B^*}(f(x, y)) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\mathbb{E}_{A?, B?}(0) \\
\vdots
\end{array}
\]
Security of warm-up protocol

Suffices to show that Bob’s view in the protocol can be simulated given just Bob’s ideal input/output.

Simulation is indistinguishable, as long as $E$ satisfies:

$$\mathbb{E}_{A,B}(C) \approx \mathbb{E}_{A',B'}(C')$$

if at least one of $\{A, B\}$ random and unknown to distinguisher.

Bob’s view (real): $\approx$ Simulated view:

<table>
<thead>
<tr>
<th>$A_4, B_2$</th>
<th>$A^<em>, B^</em>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{E}_{A_3,B_4}(f(3, 4))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(f(x, y))$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_4,B_3}(f(4, 3))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_3,B_3}(f(3, 3))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_2,B_3}(f(2, 3))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_4,B_2}(f(4, 2))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_2,B_4}(f(2, 4))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_4,B_4}(f(4, 4))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_1,B_4}(f(1, 4))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_2,B_2}(f(2, 2))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\mathbb{E}_{A_1,B_2}(f(1, 2))$</td>
<td>$\mathbb{E}_{A^<em>,B^</em>}(0)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
Problem: Cost scales with the **truth table size of** $f$.

Problem: How does Bob magically learn "correct" $A_x, B_y$?
Extending warm-up protocol

Problem: Cost scales with the truth table size of $f$!

- Idea: instead of encrypting outputs, encrypt keys to yet more garbled tables

\[
\begin{align*}
&E_{A_1, B_1}(C_1) \\
&E_{A_1, B_2}(C_1) \\
&E_{A_1, B_3}(C_2) \\
&E_{A_2, B_1}(C_1) \\
&E_{A_2, B_2}(C_3) \\
&\vdots \\
&E_{C_1, D_1}(\cdots) \\
&E_{C_1, D_2}(\cdots) \\
&E_{C_1, D_3}(\cdots) \\
&E_{C_2, D_1}(\cdots) \\
&E_{C_2, D_2}(\cdots) \\
&\vdots
\end{align*}
\]

Problem: How does Bob magically learn “correct” $A_x, B_y$?
Extending warm-up protocol

**Problem:** Cost scales with the truth table size of $f$!

- **Idea:** instead of encrypting outputs, encrypt keys to yet more garbled tables

\[
\begin{align*}
&\mathbb{E}_{A_1, B_1}(C_1) \\
&\mathbb{E}_{A_1, B_2}(C_1) \\
&\mathbb{E}_{A_1, B_3}(C_2) \\
&\mathbb{E}_{A_2, B_1}(C_1) \\
&\mathbb{E}_{A_2, B_2}(C_3) \\
&\vdots \\
&\mathbb{E}_{C_1, D_1}(\cdots) \\
&\mathbb{E}_{C_1, D_2}(\cdots) \\
&\mathbb{E}_{C_1, D_3}(\cdots) \\
&\mathbb{E}_{C_2, D_1}(\cdots) \\
&\mathbb{E}_{C_2, D_2}(\cdots) \\
&\vdots
\end{align*}
\]

**Problem:** How does Bob magically learn “correct” $A_x, B_y$?

- Discuss later (oblivious transfer)
Garbled circuit framework [Yao86]
Garbling a circuit:

▶ Pick random labels $W_0; W_1$ on each wire
▶ “Encrypt” truth table of each gate
▶ Garbled circuit: all encrypted gates
▶ Garbled encoding: one label per wire

Garbled evaluation:

▶ Only one ciphertext per gate is decryptable
▶ Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random **labels** $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
Garbled circuit framework [Yao86]

Garbling a circuit:

- Pick random **labels** $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
- **Garbled encoding** $\equiv$ one label per wire
Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- "Encrypt" truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
- **Garbled encoding** $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
Garbling a circuit:
- Pick random labels $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- Garbled circuit $\equiv$ all encrypted gates
- Garbled encoding $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire

Garbled circuit framework [Yao86]
Garbling a circuit:

- Pick random labels \( W_0, W_1 \) on each wire
- "Encrypt" truth table of each gate
- Garbled circuit \( \equiv \) all encrypted gates
- Garbled encoding \( \equiv \) one label per wire

Garbled evaluation:

- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random **labels** $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $\equiv$ all encrypted gates
- **Garbled encoding** $\equiv$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Garbled circuit framework [Yao86]

Garbling a circuit:
- Pick random **labels** $W_0, W_1$ on each wire
- “Encrypt” truth table of each gate
- **Garbled circuit** $≡$ all encrypted gates
- **Garbled encoding** $≡$ one label per wire

Garbled evaluation:
- Only one ciphertext per gate is decryptable
- Result of decryption = value on outgoing wire
Key idea: Given garbled circuit + garbled input . . .
Syntax & Security (informal)

Key idea: Given garbled circuit + garbled input . . .

- . . . Only thing you can do is (blindly) evaluate circuit on that input
Key idea: Given garbled circuit + garbled input . . .

- . . . Only thing you can do is (blindly) evaluate circuit on that input
- Learn only 1 label per wire: hard to guess “complementary” label
- Seeing a single label hides logical value on wire, although . . .
- Revealing both labels on output wires leaks only circuit output
Syntax & Security [BellareHoangRogaway12]

Garble \rightarrow f \rightarrow \text{Garble} \xrightarrow{\text{encoding info}} \text{Encode} \xrightarrow{x} \text{Eval} \xrightarrow{\text{garbled output}} \text{Decode} \xrightarrow{\text{decoding info}} f(x)

Formal security properties:

Privacy: \((F;X;d)\) reveals nothing beyond \(f\) and \(f(x)\)

Obliviousness: \((F;X)\) reveals nothing beyond \(f\)

Authenticity: given \((F;X)\), hard to find \(e\) such that \(\text{decode} < f f(x)\)

Other interesting notions we won't discuss:

Adaptive security: choice of input can depend on garbled circuit

Gate-hiding: \((F;X;d)\) reveals nothing beyond topology of \(f\) and \(f(x)\).
Syntax & Security

[BellareHoangRogaway12]

GarbleEncodeEvalDecode

Formal security properties:

Privacy:
\((F; X; d)\) reveals nothing beyond \(f\) and \(f(x)\)

Obliviousness:
\((F; X)\) reveals nothing beyond \(f\)

Authenticity:
given \((F; X)\), hard to find \(e\) \(Y\) that decodes < \(f(x)\) ?

Other interesting notions we won’t discuss:

Adaptive security:
choice of input can depend on garbled circuit

Gate-hiding:
\((F; X; d)\) reveals nothing beyond topology of \(f\) and \(f(x)\).
Syntax & Security

[BellareHoangRogaway12]

Formal security properties:

**Privacy:** \((F, X, d)\) reveals nothing beyond \(f\) and \(f(x)\)

**Obliviousness:** \((F, X)\) reveals nothing beyond \(f\)

**Authenticity:** given \((F, X)\), hard to find \(\tilde{Y}\) that decodes \(\notin \{f(x), \perp\}\)
Syntax & Security

Formal security properties:

**Privacy:** \((F, X, d)\) reveals nothing beyond \(f\) and \(f(x)\)

**Obliviousness:** \((F, X)\) reveals nothing beyond \(f\)

**Authenticity:** given \((F, X)\), hard to find \(\widetilde{Y}\) that decodes \(\notin \{f(x), \bot\}\)

Other interesting notions we won’t discuss:

**Adaptive security:** choice of input can depend on garbled circuit

**Gate-hiding:** \((F, X, d)\) reveals nothing beyond topology of \(f\) and \(f(x)\)
Oblivious transfer

How does evaluator (Bob) get the garbled input?

\begin{align*}
A_0, A_1 & \quad \text{Garbler's inputs:} \\
B_0, B_1 & \\
C_0, C_1 & \\
D_0, D_1 & 
\end{align*}
Oblivious transfer

How does evaluator (Bob) get the garbled input?

Alice \{ 
- \( A_0, A_1 \)
- \( B_0, B_1 \)
- \( C_0, C_1 \)

Bob \{ 
- \( D_0, D_1 \)
Oblivious transfer

How does evaluator (Bob) get the garbled input?

**Garbler’s inputs:** She knows both $A_0, A_1$, and which one is correct $\implies$ just send correct one to Bob
Oblivious transfer

How does evaluator (Bob) get the garbled input?

**Alice**
- \( A_0, A_1 \)
- \( B_0, B_1 \)
- \( C_0, C_1 \)
- \( D_0, D_1 \)

**Bob**
- \( A_0, A_1 \)
- \( B_0, B_1 \)
- \( C_0, C_1 \)
- \( D_0, D_1 \)

**Garbler’s inputs:** She knows both \( A_0, A_1 \), and which one is correct \( \Rightarrow \) just send correct one to Bob

**Evaluator’s inputs:** We need the following “gadget”:

\[
\begin{align*}
W_0, W_1 & \quad \rightarrow \quad \text{OT} \\
& \quad \rightarrow \quad c \in \{0, 1\} \\
& \quad \rightarrow \quad W_c
\end{align*}
\]
Oblivious transfer

How does evaluator (Bob) get the garbled input?

Garbler’s inputs: She knows both $A_0, A_1$, and which one is correct $\Rightarrow$ just send correct one to Bob

Evaluator’s inputs: We need the following “gadget” (oblivious transfer):

$$\begin{align*}
W_0, W_1 &\rightarrow \text{OT} & c \in \{0, 1\} \\
& & \rightarrow W_c
\end{align*}$$
How to construct OT?

$W_0, W_1 \rightarrow OT \leftarrow c \quad W_c$

$W_0, W_1 \quad c$

Need public-key encryption that supports blind key generation:

▶ sample a public key without knowledge of secret key
▶ E.g.: ElGamal (sample group element without knowing discrete log)
How to construct OT?

Need public-key encryption that supports **blind key generation**:

- sample a public key without knowledge of secret key
- E.g.: ElGamal (sample group element without knowing discrete log)
How to construct OT?

Need public-key encryption that supports **blind key generation**:

- sample a public key without knowledge of secret key
- E.g.: ElGamal (sample group element without knowing discrete log)
Yao’s Protocol: overview

$\text{garbled circuit } f$, garbled input $x$, output wire labels $\text{OT}(n)$, input wire labels $y$, garbled $f(x; y)$. Bob learns only $f(x; y)$. 
Yao’s Protocol: overview

- garbled circuit $f$,
- garbled input $x$,
- output wire labels

Given garbled $f$ + garbled inputs + all output labels

Bob learns only $f(x; y)$.
Yao’s Protocol: overview

Given garbled circuit $f$, garbled input $x$, output wire labels

Input wire labels $x$ → OT ($\times n$) → y → garbled $y$
Yao’s Protocol: overview

- Given garbled $f$ + garbled inputs + all output labels $\Rightarrow$ Bob learns only $f(x, y)$
Secure Computation allows parties to perform a computation on private input, learning only the output.

- market clearing price, advertising revenue, . . .

Security: every attack against the protocol can be “simulated” in an ideal world interaction.

Yao’s protocol:

- Garbled lookup table for each gate of boolean circuit
- Oblivious transfer for each input wire
Next lectures:

2

Garbled circuits are extremely large
  ▶ How to reduce their size by 10×

3

Yao’s protocol insecure against malicious attacks:
  ▶ How to harden the protocol against malicious adversaries

4

Oblivious transfer is prohibitively expensive:
  ▶ How to “amplify” OT instances using cheap crypto

5

Special-purpose protocols can be much faster than Yao’s
  ▶ How to securely compute set intersection