Foundation of Cryptography, Lecture 10
Secure Computation

Handout Mode

Iftach Haitner
Tel Aviv University.

May 16, 2019

\(^1\)Last edited on: July 1, 2019.
Section 1

The Model
Multiparty Computation

- Multiparty Computation – computing a functionality $f$
- **Secure** Multiparty Computation: compute $f$ in a “secure manner”
  - Correctness
  - Privacy
  - Independence of inputs
  - Guaranteed output delivery
  - Fairness: corrupted parties should get their output iff the honest parties do
  - and ...
- Examples: coin-tossing, broadcast, electronic voting, electronic auctions
- How should we model it?
- Real Vs. Ideal paradigm
Real-model execution

For a pair of algorithms $\vec{A} = (A_1, A_2)$ and inputs $x_c, x_1, x_2 \in \{0, 1\}^*$, let $\text{REAL}_{\vec{A}}(x_c, x_1, x_2)$ be the joint output of $(A_1(x_c, x_1), A_2(x_c, x_2))$.

Given a two-party protocol $\pi$, an algorithm taking the role of one of the parties in $\pi$ is:

- **Malicious** — acts arbitrarily.
- **Honest** — acts exactly according to $\pi$.
- **Semi-honest** — acts according to $\pi$, but might output additional information.

$\vec{A} = (A_1, A_2)$ is an admissible with respect to $\pi$, if at least one party is honest.
Ideal model execution

For a pair of algorithms $\overline{B} = (B_1, B_2)$, inputs $x_c, x_1, x_2 \in \{0, 1\}^*$ and a function $f = (f_1, f_2)$, let $\text{IDEAL}^f_{\overline{B}}(x_c, x_1, x_2)$ be the joint output of the parties in the end of the following experiment:

1. The input of $B_i$ is $(x_c, x_i)$.
2. $B_i$ sends value $y_i$ to the trusted party.
3. Trusted party sends $z_i = f_i(y_0, y_1)$ to $B_i$ in an arbitrary order.
4. Each party outputs some value.

The actual definition allows a party after receiving its output, to instruct $f$ not to send the the output to the other party.

An oracle-aided algorithm $B$ taking the role of one of the parties is:

- **Malicious** — acts arbitrarily.
- **Honest** — sends its private input to the trusted party (i.e., sets $y_i = x_i$), and its only output is the value it gets from the trusted party (i.e., $z_i$).
- **Semi-honest**, sends its input to the trusted party, outputs $z_i$ plus possibly additional information.

$\overline{B} = (B_1, B_2)$ is admissible, if at least one party is honest.
Definition 1 (secure computation)

A protocol $\pi$ securely computes $f$, if $\forall$ admissible PPT pair $\overline{A} = (A_1, A_2)$ for $\pi$, exists admissible PPT pair $\overline{B} = (B_1, B_2)$, s.t.

$$\{\text{REAL}_{\overline{A}}(x_c, x_1, x_2)\}_{x_c, x_1, x_2 \in \{0,1\}^*} \approx_c \{\text{IDEAL}^f_{\overline{B}}(x_c, x_1, x_2)\}_{x_c, x_1, x_2 \in \{0,1\}^*}$$

In case $\overline{A}$ is honest, we require that $\overline{B}$ is honest, and the ensembles to be identical.

- Recall that the enumeration index (i.e., $x_c, x_1, x_2$) is given to the distinguisher.
- $\pi$ securely computes $f$ implies that $\pi$ computes $f$ correctly.
- Security parameter
- Auxiliary inputs
- We focus on semi-honest adversaries.
Section 2

Oblivious Transfer
Oblivious transfer

An (one-out-of-two) OT protocol securely computes the functionality
\( OT = (OT_S, OT_R) \) over \( (\{0, 1\}^* \times \{0, 1\}^*) \times \{0, 1\} \), where \( OT_S(\cdot) = \bot \) and
\( OT_R((\sigma_0, \sigma_1), i) = \sigma_i. \)

▶ “Complete” for multiparty computation
▶ We show how to construct for bit inputs.
Oblivious transfer from trapdoor permutations

Let \((G, f, \text{Inv})\) be a TDP and let \(b\) be an hardcore predicate for \(f\).

**Protocol 2 ((S, R))**

Common input: \(1^n\)

S’s input: \(\sigma_0, \sigma_1 \in \{0, 1\}\).

R’s input: \(i \in \{0, 1\}\).

1. S chooses \((e, d) \leftarrow G(1^n)\), and sends \(e\) to R.
2. R chooses \(x_0, x_1 \leftarrow \{0, 1\}^n\), sets \(y_i = f_e(x_i)\) and \(y_{1-i} = x_{1-i}\), and sends \(y_0, y_1\) to S.
3. S sets \(c_j = b(\text{Inv}_d(y_j)) \oplus \sigma_j\), for \(j \in \{0, 1\}\), and sends \((c_0, c_1)\) to R.
4. R outputs \(c_i \oplus b(x_i)\).

**Claim 3**

Protocol 2 securely computes OT (in the semi-honest model).
Proving Claim 3

We need to prove that ∀ semi-honest admissible PPT pair $\overline{A} = (A_1, A_2)$ for $(S, R)$, exists admissible oracle-aided PPT pair $\overline{B} = (B_1, B_2)$ s.t.

$$\{\text{REAL}_\overline{A}(1^n, (\sigma_0, \sigma_1), i)\} \approx_c \{\text{IDEAL}^\text{OT}_\overline{B}(1^n, (\sigma_0, \sigma_1), i)\},$$  \hspace{1cm} (1)

where the enumeration is over $n \in \mathbb{N}$ and $\sigma_0, \sigma_1, i \in \{0, 1\}$.
### R’s security

For a semi-honest $S'$, define semi-honest strategy $S'_I$ as follows.

#### Algorithm 4 ($S'_I$)

**input:** $1^n, \sigma_0, \sigma_1$

1. Send $(\sigma_0, \sigma_1)$ to the trusted party.
2. Emulate $(S'(1^n, \sigma_0, \sigma_1), R(1^n, 0))$.
3. Output the output that $S'$ does.

Let $\bar{A} = (S', R)$ and $\bar{B} = (S'_I, R_I)$, where $R_I$ is honest.

#### Claim 5

$$\{\text{REAL}_{\bar{A}}(1^n, (\sigma_0, \sigma_1), i)\} \equiv \{\text{IDEAL}_{\bar{B}}^{\text{OT}}(1^n, (\sigma_0, \sigma_1), i)\}.$$

Proof?
S’s security

For a semi-honest implementation $R'$ of $R$, define the oracle-aided semi-honest strategy $R'_I$ as follows.

**Algorithm 6 ($R'_I$)**

**input:** $1^n, i \in \{0, 1\}$,

1. Send $i$ to the trusted party, and let $\sigma$ be its answer.
2. Emulate $(S(1^n, \sigma_0, \sigma_1), R'(1^n, i))$, for $\sigma_i = \sigma$ and $\sigma_{1-i} = 0$.
3. Output the output that $R'$ does.

Let $\overline{A} = (S, R')$ and $\overline{B} = (S_I, R'_I)$, where $S_I$ is honest.

**Claim 7**

$$\{\text{REAL}_{\overline{A}}(1^n, (\sigma_0, \sigma_1), i)\} \approx_c \{\text{IDEAL}^{\text{OT}}_{\overline{B}}(1^n, (\sigma_0, \sigma_1), i)\}.$$

**Proof?**
Section 3

Yao Garbled Circuit
Before we start

- Fix a (multiple message) semantically-secure private-key encryption scheme \((G, E, D)\) with
  
  1. \(G(1^n) = U_n\).
  2. For any \(m \in \{0, 1\}^*\)
     
     \[
     \Pr_{d, d' \leftarrow \{0, 1\}^n} [D_d(E_{d'}(m)) \neq \perp] = \text{neg}(n).
     \]

- Can we construct such a scheme?
  
  Yes, append \(0^n\) at the end of the message.

- Boolean circuits: gates, wires, inputs, outputs, values, computation
The Garbled Circuit

Fix a Boolean circuit $C$ and $n \in \mathbb{N}$.

- Let $\mathcal{W}$ and $\mathcal{G}$ be the (indices) of wires and gates of $C$, respectively.
- For $w \in \mathcal{W}$, associate a pair of random ‘keys’ $k_w = (k_w^0, k_w^1) \in (\{0, 1\}^n)^2$.
- For $g \in \mathcal{G}$ with input wires $i$ and $j$, and output wire $h$, let $T(g)$ be the following table:

<table>
<thead>
<tr>
<th>input wire $i$</th>
<th>input wire $j$</th>
<th>output wire $h$</th>
<th>hidden output wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i^0$</td>
<td>$k_j^0$</td>
<td>$k_h^{g(0,0)}$</td>
<td>$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$</td>
</tr>
<tr>
<td>$k_i^0$</td>
<td>$k_j^1$</td>
<td>$k_h^{g(0,1)}$</td>
<td>$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$</td>
</tr>
<tr>
<td>$k_i^1$</td>
<td>$k_j^0$</td>
<td>$k_h^{g(1,0)}$</td>
<td>$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$</td>
</tr>
<tr>
<td>$k_i^1$</td>
<td>$k_j^1$</td>
<td>$k_h^{g(1,1)}$</td>
<td>$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$</td>
</tr>
</tbody>
</table>

**Figure:** Table for gate $g$, with input wires $i$ and $j$, and output wire $h$. 
## The Garbled Circuit, cont.

<table>
<thead>
<tr>
<th>input wire $i$</th>
<th>input wire $j$</th>
<th>output wire $h$</th>
<th>hidden output wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i^0$</td>
<td>$k_j^0$</td>
<td>$k_h^{g(0,0)}$</td>
<td>$E_{k_i^0}(E_{k_j^0}(k_h^{g(0,0)}))$</td>
</tr>
<tr>
<td>$k_i^0$</td>
<td>$k_j^1$</td>
<td>$k_h^{g(0,1)}$</td>
<td>$E_{k_i^0}(E_{k_j^1}(k_h^{g(0,1)}))$</td>
</tr>
<tr>
<td>$k_i^1$</td>
<td>$k_j^0$</td>
<td>$k_h^{g(1,0)}$</td>
<td>$E_{k_i^1}(E_{k_j^0}(k_h^{g(1,0)}))$</td>
</tr>
<tr>
<td>$k_i^1$</td>
<td>$k_j^1$</td>
<td>$k_h^{g(1,1)}$</td>
<td>$E_{k_i^1}(E_{k_j^1}(k_h^{g(1,1)}))$</td>
</tr>
</tbody>
</table>

Let $\mathcal{I}$ and $\mathcal{O}$ be the input and outputs wires of $C$.

- For $g \in \mathcal{G}$, let $\tilde{T}(g)$ be a random permutation of the fourth column of $T(g)$.
- For $w \in \mathcal{W}$, let $C(x)_w$ be the bit-value computation of $C(x)$ assigns to $w$.
- Given
  1. $\tilde{T} = \{(g, \tilde{T}(g))\}_{g \in \mathcal{G}}$.
  2. $\{k_w^{C(x)}\}_{w \in \mathcal{I}}$ for some $x$.
  3. $\{(w, k_w = (k_w^0, k_w^1))\}_{w \in \mathcal{O}}$.

One can efficiently compute $C(x)$.

- (essentially) The above leaks no additional information about $x$!
Example, GV for OR

On board...
The protocol

Let $f(x_A, x_B) = (f_A(x_A, x_B), f_B(x_A, x_B))$ be a function, and let $C$ be a circuit that computes $f$.

Let $I_A$ and $I_B$ be the input wires corresponds to $x_A$ and $x_B$ respectively in $C$, and let $O_A$ and $O_B$ be the output wires corresponds to $f_A$ and $f_B$ outputs respectively in $C$.

Recall that $C(x)_w$ is the bit-value the computation of $C(x)$ assigns to $w$.

Let $(S, R)$ be a secure protocol for OT.

---

Protocol 8 ($(A, B)$)

**Common input: $1^n$. A/B’s input: $x_A/x_B$**

1. A samples at random $\{k_w = (k_w^0, k_w^1)\}_{w \in W}$, and generate $\tilde{T}$.
2. A sends $\tilde{T}$ and $\{(w, k_w^{C(x_1, \cdot)_w})\}_{w \in I_A}$ to B.
3. $\forall w \in I_B$, A and B interact in $(S(k_w), R(C(\cdot, x_2)_w))(1^n)$.
4. B computes the (garbled) circuit, and sends $\{(w, k_w^{C(x_1, x_2)_w})\}_{w \in O_A}$ to A.
5. A sends $\{(w, k_w)\}_{w \in O_B}$ to B.
6. The parties compute $f_A(x_1, x_2)$ and $f_B(x_1, x_2)$ respectively.
Example, protocol for OR

On board...
Claim 9

Protocol 8 securely computes $f$ (in the semi-honest model)

Proof: We focus on the security of $A$. For a semi-honest $B'$, define

Algorithm 10 ($B'_I$)

**input:** $1^n$ and $x_B$.

1. Send $x_B$ to the trusted party, and let $o_B$ be its answer.
2. Emulate the first 4 steps of $(A(1^{|x_A|}), B'(x_B)(1^n))$.
3. For each $w \in O_B$: permute the order of the pair $k_w$ according to $o_B$, and the key of $w$ computed in the emulation.
4. Complete the emulation, and output the output that $B'$ does.

Claim: $B'_I$ is a good “simulator” for $B'$.

Security of $B$?
Extensions

- Efficiently computable $f$
  Both parties first compute $C_f$ – a circuit that compute $f$ for inputs of the right length
- Hiding $C$? All but its size
Malicious model

The parties prove that they act “honestly”:

1. Forces the parties to chose their random coin properly

2. Before each step, the parties prove in $\mathsf{ZK}$ that they followed the prescribed protocol (with respect to the random-coins chosen above)
Course summary

See diagram
What we did not cover

- Fully homomorphic encryption and obfuscation.
- “Few” reductions
- Environment security (e.g., UC)
- Information theoretic crypto
- Non-generic constructions: number theory, lattices
- Impossibility results
- “Real life cryptography” (e.g., Random oracle model)
- Security
- Differential privacy
- and....