Secure Computation - I

CS 601.642/442 Modern Cryptography

Fall 2017

Motivating Example

Consider two billionaires Alice and Bob with net worths x and y, respectively:

- They want to find out who is richer by computing the following function

$$f(x,y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{otherwise} \end{cases}$$

- Potential Solution: Alice sends x to Bob, who sends y to Alice. They each compute f on their own.
- Problem: Alice learns Bob's net worth (and vice-versa). No privacy!
- Main Question: Can Alice and Bob compute f in a "secure manner" s.t. they only learn the output of f, and *nothing more*?

General Setting

Two parties A and B, with private inputs x and y, respectively:

- They want to "securely" compute a function f
- If both A and B are honest, then they should learn the output $f(x, y)$
- Even if one party is adversarial, it should not learn anything beyond the output (and its own input)
- Think: How to formalize this security requirement?

Types of Adversaries

Two types of adversaries:

- **Honest but curious (a.k.a. semi-honest):** Such an adversary follows the instructions of the protocol, but will later analyze the protocol transcript to learn any "extra information" about the input of the other party
- **Malicious:** Such an adversary can deviate from the protocol instructions and follow an arbitrary strategy

Note: We will only consider *semi-honest* adversaries

Secure Computation: Intuition

- Want to formalize that no semi-honest adversary learns anything from the protocol execution beyond its input and the (correct) output
- Idea: Use simulation paradigm, as in zero-knowledge proofs
- View of adversary in the protocol execution can be efficiently simulated given only its input and output, and without the input of the honest party

Secure Computation: Definition

Definition (Semi-honest Secure Computation)

A protocol $\pi = (A, B)$ securely computes a function f in the semi-honest model if there exists a pair of non-uniform PPT simulator algorithms $\mathcal{S}_A, \mathcal{S}_B$ such that for every security parameter n, and all inputs $x, y \in \{0,1\}^n$, it holds that:

$$\left\{\mathcal{S}_A(x, f(x,y)), f(x,y)\right\} \approx \left\{e \leftarrow [A(x) \leftrightarrow B(y)] : \mathsf{View}_A(e), \mathsf{Out}_B(e)\right\},$$

$$\left\{\mathcal{S}_B(y, f(x,y)), f(x,y)\right\} \approx \left\{e \leftarrow [A(x) \leftrightarrow B(y)] : \mathsf{View}_B(e), \mathsf{Out}_A(e)\right\}.$$

Remarks on Definition

- Recall: In zero-knowledge, we only require indistinguishability of simulated view and adversary's view in the real execution
- Here, indistinguishability is w.r.t. the *joint distribution* over the adversary's view and the honest party's output
- This is necessary for **correctness**: it implies that output of the honest party in the protocol execution must be indistinguishable from the correct output $f(x, y)$
- If we remove this requirement, then a clearly wrong protocol where parties are instructed to output y would be trivially secure!

Oblivious Transfer

Consider the following functionality, called, 1-out-of-2 oblivious transfer (OT):

- Two parties: Sender A, and Receiver B
- Inputs: A's input is a pair of bits (a_0, a_1), and B's input is a bit b
- Outputs: B's output is a_b, and A receives no output

Note: Definition of secure computation promises that in a secure OT protocol, A does not learn b and B does not learn a_{1-b}

Importance of Oblivious Transfer

- Can be realized from physical channels [Wiener,Rabin]
- **OT is complete:** given a secure protocol for OT, any function can be securely computed
- **OT is necessary:** OT is the minimal assumption for secure computation

Oblivious Transfer: Construction

Let $\{f_i\}_{i \in \mathcal{I}}$ be a family of trapdoor permutations with sampling algorithm Gen. Let h be a hardcore predicate for any f_i.

Sender's input: (a_0, a_1) where $a_i \in \{0,1\}$

Receiver's input: $b \in \{0,1\}$

Protocol $\mathsf{OT} = (A, B)$**:**

$A \rightarrow B$: A samples $(f_i, f_i^{-1}) \leftarrow \mathsf{Gen}(1^n)$ and sends f_i to B

$B \rightarrow A$: B samples $x \overset{\$}{\leftarrow} \{0,1\}^n$ and computes $y_b = f_i(x)$. It also samples $y_{1-b} \overset{\$}{\leftarrow} \{0,1\}^n$. B sends (y_0, y_1) to A

$A \rightarrow B$: A computes the inverse of each value y_j and XORs the hard-core bit of the result with a_j:

$$z_j = h(f_i^{-1}(y_j)) \oplus a_j$$

A sends (z_0, z_1) to B

$B(x, b, z_0, z_1)$: B outputs $h(x) \oplus z_b$

$\mathsf{OT} = (A, B)$ is Semi-honest Secure : Intuition

- Security against A: Both y_0 and y_1 are uniformly distributed and therefore independent of b. Thus, b is hidden from A
- Security against B: If B could learn a_{1-b}, then it would be able to predict the hardcore predicate

Note: A *malicious* B can easily learn a_{1-b} by deviating from the protocol strategy

$\mathsf{OT} = (A, B)$ is Semi-honest Secure : Simulator $\mathcal{S}_A$

Simulator $\mathcal{S}_A((a_0, a_1), \perp)$:

1. Fix a random tape r_A for A. Run honest emulation of A using (a_0, a_1) and r_A to obtain the first message f_i
2. Choose two random strings $y_0, y_1 \in \{0,1\}^n$ as B's message
3. Run honest emulation of A using (y_0, y_1) to obtain the third message (z_0, z_1)
4. Stop and output $\perp$

Claim: The following two distributions are identical:
$\left\{\mathcal{S}_A((a_0, a_1), \perp), a_b\right\}$ and
$\left\{e \leftarrow [A(a_0, a_1) \leftrightarrow B(b)] : \mathsf{View}_A(e), \mathsf{Out}_B(e)\right\}$

Proof: The only difference between $\mathcal{S}_A$ and real execution is in step 2. However, since f is a permutation, y_0, y_1 are identically distributed in both cases.

Simulator $\mathcal{S}_B(b, a_b)$:

1. Sample f_i
2. Choose random tape r_B for B. Run honest emulation of B using (b, r_B, f_i) to produce (x, y_0, y_1) s.t. $y_b = f_i(x)$ and $y_{1-b} \xleftarrow{\$} \{0,1\}^n$
3. Compute $z_b = h(x) \oplus a_b$ and $z_{1-b} \xleftarrow{\$} \{0,1\}$
4. Output (z_0, z_1) as third message and stop

Claim: The following two distributions are indistinguishable:
$\left\{\mathcal{S}_B(b, a_b), \perp\right\}$ and $\left\{e \leftarrow [A(a_0, a_1) \leftrightarrow B(b)] : \mathsf{View}_B(e), \mathsf{Out}_A(e)\right\}$

Proof: The only difference is in step 3, where $\mathcal{S}_B$ computes z_{1-b} as a random bit. However, since $h(f_i^{-1}(y_{1-b}))$ is indistinguishable from random (even given y_{1-b}), this change is indistinguishable

Remarks

1-out-of-k OT:

- The previous protocol can be easily generalized to construct 1-out-of-k OT for $k > 2$

Semi-honest vs Malicious:

- In reality, adversary may be malicious and not semi-honest
- Goldreich-Micali-Wigderson [GMW] gave a compiler to transform *any* protocol secure against semi-honest adversary into one secure against malicious adversary
- The transformation uses coin-flipping (to make sure that adversary's random tape is truly random) and zero-knowledge proofs (to make sure that adversary is following the protocol instructions)
- Details outside the scope of this class