CS 598DH Secure Computation – Homework 2

Professor David Heath

Due: March 21

Problem 1. Sometimes it is not clear whether a certain behavior is an "attack" against a protocol. Recall that to decide whether a behavior violates security, we must determine whether the behavior is possible in the ideal world.

For each of the following, consider a **malicious** adversary that successfully carries out the described behavior *as part of a real-world protocol*. Specify whether this behavior indicates that the real-world protocol is *insecure* (yes or no). If the procotol is insecure, give a brief explanation.

Hint: recall the malicious model ideal execution. What additional capabilities do we allow malicious adversaries as compared to semi-honest adversaries?

1. Alice holds x and Bob holds y, where $x, y \in \{0, ..., N-1\}$. They wish to compute:

   $$(x + y) \bmod N$$

   An adversary corrupts Alice and learns Bob's input in its entirety.

2. Alice holds x and Bob holds y, where $x, y \in \{0, ..., N-1\}$. They wish to compute:

   $$(x + y) \bmod N$$

   An adversary corrupts Alice and forces Bob to output zero.

3. Alice holds x and Bob holds y, where $x, y \in \{0, ..., N-1\}$. They wish to compute:

   $$(x + y) \bmod N$$

   An adversary corrupts Alice and prevents Bob from learning the output.

4. Alice holds x and Bob holds y, where $x, y \in \{0, ..., 7\}$. They wish to compute $x + y$. An adversary corrupts Alice and forces Bob to output a value greater than or equal to 7.

5. Alice holds x and Bob holds y, where $x, y \in \{0, ..., 7\}$. They wish to compute:

   $$\begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$$

   An adversary corrupts Alice and learns the most significant bit of y (assume y is written as a three bit value, e.g. $4 = 100_2$).

Answer 1.

Problem 2. It is often useful to build a secure computation from another already-designed protocol. In class, we have discussed 1-out-of-2 OT, where the receiver selects one of two secrets; it is natural to consider generalizations to 1-out-of-N OT, where the receiver selects one of N secrets.

1. **Warm-up:** Suppose you have a semi-honest secure 1-out-of-4 OT protocol. Construct a semi-honest 1-out-of-2 OT protocol.

2. Suppose you have a semi-honest secure 1-out-of-4 OT protocol. Construct a semi-honest protocol that simultaneously executes two 1-out-of-2 OTs. Your protocol may not make more than one call to the 1-out-of-4 protocol.

3. Suppose you have a semi-honest secure 1-out-of-2 OT protocol. Construct a semi-honest 1-out-of-4 OT protocol.

Prove your protocols are secure in the **semi-honest model** by constructing simulators and arguing indistinguishability.

Answer 2.

Problem 3. Consider an arbitrary two-party protocol Π, and suppose that Π is secure in the malicious model. One might think that Π is *also* secure in the *semi-honest* model. Perhaps surprisingly, this is not necessarily the case.

Consider the following one-sided AND functionality:

PARAMETERS:

1. Let P_0, P_1 be two parties.
2. Each party P_i has input $x_i \in \{0, 1\}$.

FUNCTIONALITY:

1. P_0 outputs $\perp$.
2. P_1 outputs $x_0 \wedge x_1$.

I.e., the parties compute AND, but only P_1 receives output. Consider the following protocol Π_{AND} for the above functionality:

PROTOCOL:

1. P_0 sends x_0 to P_1 and outputs $\perp$.
2. P_1 outputs $x_0 \wedge x_1$.

Π_{AND} is not secure in the semi-honest model, but it *is secure* in the malicious model.

1. Give a brief and informal argument that explains why this protocol is secure in the presence of a malicious adversary, but not a semi-honest adversary.

2. Prove that Π_{AND} is not secure in the semi-honest model.

3. Prove that Π_{AND} is secure in the malicious model by constructing simulators for P_0 and P_1.

4. **Bonus.** Suppose we are willing to adjust our definition of semi-honest security to ensure that malicious security *does* imply semi-honest security. How would you adjust the definition? Informally argue (1) that your change still captures the notion of an adversary that is honest but curious and (2) that malicious security implies security under your adjusted definition.

Answer 3.

Problem 4. In this problem, suppose we have access to a maliciously secure protocol Π for 1-out-of-2 OT.

1. Suppose we would like to use Π to design a maliciously secure protocol for computing arbitrary functions. To do so, we take the semi-honest GMW protocol as specified in class, and we substitute semi-honest OT by malicious OT. Is this modified GMW protocol maliciously secure? If so, argue why. If not, briefly describe an attack by a malicious adversary.

2. Recall the XOR functionality, where Alice inputs a bit x, Bob inputs a bit y, and each party outputs $x \oplus y$. Recall that in class we showed some malicious protocols by using commitments. Construct a maliciously secure protocol for the XOR protocol the *does not* use commitments, but you may invoke the OT protocol Π once. Argue informally that your protocol is secure. There is no need to construct formal simulators. *Hint: It is okay if the malicious adversary can launch an "attack" that succeeds only with negligible probability.*

Answer 4.

Problem 5. The definition of semi-honest security requires us to simulate *each possible subset* of corrupted parties.

Consider the 3-party setting. One might naively assume that it suffices to simulate each *size-two* subset of corrupted parties, without needing to simulate subsets of size one. This is not true.

1. Construct a three-party functionality and a corresponding protocol such that the protocol (1) is secure against an adversary who corrupts exactly two parties, but (2) is insecure under the full definition of semi-honest security.

   Hint: There exist very simple protocols that meet these requirements.

2. Construct simulators that prove that your protocol securely achieves your functionality when a semi-honest adversary corrupts an arbitrary size-two subset of the parties.

3. Prove that your protocol is insecure under the full definition of semi-honest security (by demonstrating that some single party cannot be simulated).

Answer 5.

Problem 6. Consider the Garbled Circuit (GC) protocol. (You can see a formalization in Figures 3.1 and 3.2 of `https://securecomputation.org/docs/ch3-fundamentalprotocols.pdf`.) Recall that basic GC does not offer protection against a malicious garbler.

Consider a circuit $\mathcal{C}$ with a single output wire and 20 input wires, 10 of which are the garbler's input and 10 of which are the evaluator's input. Consider the following scenario:

- The garbler garbles the circuit: $\hat{\mathcal{C}}, X \leftarrow \text{Garble}(\mathcal{C})$. Here, $\hat{\mathcal{C}}$ is the garbled circuit and X is a vector of pairs of labels (keys) for the circuit input wires. That is, X_i^0 is the zero label for wire i; X_i^1 is the one label for wire i. Each label is a λ-bit key (when garbling in practice, λ is often set to 128).
- The evaluator obtains (via a TTP) input labels from X that correspond to her input.
- The garbler sends to the evaluator both $\hat{\mathcal{C}}$ and the labels from X that correspond to his input.
- The evaluator evaluates the garbled circuit and obtains a single length-λ label on the single output wire.
- The evaluator sends this output label to the garbler.

Let us consider the case where the garbler is malicious.

1. Describe a malicious adversary $\mathcal{A}$ that can learn a single bit of the evaluator's input. Namely, your adversary does not call Garble as prescribed, but instead runs arbitrary code to build a "garbled circuit".
2. Describe a new adversary $\mathcal{A}$ that can with overwhelming probability learn the evaluator's *entire input*.

   Hint: Suppose that the security parameter $\lambda \gg 10$. You may assume that $\mathcal{C}$'s single output wire depends on every input wire.

Answer 6.