CMSC 858K — Introduction to Secure Computation September 9, 2013

Lecture 2

Lecturer: Jonathan Katz *Scribe(s): Aishwarya Thiruvengadam*

1 Summary

In this lecture, we introduce secure n-party computation in the semi-honest setting. We present the definition for n-party computation in this setting and state the composition theorem.

2 Secure n-party computation

Consider the scenario where a set of n parties $P_1, \ldots, P_n$ wish to compute a function f on their n inputs and receive the resulting outputs. The problem of secure computation is to guarantee security of this function evaluation in the presence of an adversary. The security guarantees here would be to ensure correctness of the output and secrecy of the data held by the parties. We formalize this below.

We start by looking at n-party computation in the semi-honest setting. A *semi-honest* adversary is a passive adversary, wherein it looks only at the protocol transcript and tries to learn additional information if possible. It does not actively deviate from the protocol.

First, let us formalize what happens in the real world. All parties interact with each other according to the protocol in the presence of a real-life adversary who controls a certain set of parties. At the end of the computation, the uncorrupted parties output whatever is specified in the protocol. The adversary, controlling the corrupted parties, outputs some arbitrary value. This value may include information that the adversary gathered during the computation.

Let f be a randomized functionality from n inputs to n outputs (i.e.) $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$. A protocol Π *computes* f if, when parties $P_1, \ldots P_n$ run Π on inputs $k, x_1, \ldots, x_n$, they output $y_1, \ldots y_n$ that is distributed according to the output function $f(x_1, \ldots, x_n)$. Note that k is the security parameter. Let $\mathcal{A}$ be a semi-honest adversary that corrupts some set of t parties.

Define

$$\text{REAL}_{\Pi,\mathcal{A}}(k, x_1, \ldots, x_n, z) = \text{OUT}_\Pi(k, x_1, \ldots, x_n), \text{VIEW}_{\Pi,\mathcal{A}}(k, x_1, \ldots, x_n, z) \tag{1}$$

where $\text{OUT}_\Pi(k, x_1, \ldots, x_n)$ is the output of the honest parties and $\text{VIEW}_{\Pi,\mathcal{A}}(k, x_1, \ldots, x_n, z)$ denotes the view of the adversary $\mathcal{A}$. The view of the adversary includes the inputs of the corrupted parties, the randomness of $\mathcal{A}$ (if $\mathcal{A}$ is randomized), the auxiliary input z, incoming protocol messages to the corrupted parties.[1] Note that $\text{OUT}_\Pi(k, x_1, \ldots, x_n)$ and

[1] Note that messages between honest parties are included in $\mathcal{A}$'s view if the adversary can eavesdrop. Typically, this is ignored since it is assumed that the messages are encrypted by keys the honest parties have shared.

$\text{VIEW}_{\Pi,\mathcal{A}}(k, x_1, \ldots, x_n, z)$ are correlated random variables.[2]

We will compare this to the *ideal world* setting in order to capture the security we need in the real world. An ideal world adversary $\mathcal{S}$ controls a set of corrupted parties, and learns their inputs. All parties then hand their (possibly corrupted) inputs to an incorruptible trusted party. The trusted party evaluates the given function at these inputs and hands each party its designated output. Finally, the uncorrupted parties output whatever they receive from the trusted party and the adversary outputs some arbitrary value. The adversary's output may contain any information gathered by the adversary in the ideal process. However, this information is very limited. It consists only of the adversary's random input, the identities of the corrupted parties, their inputs and the values they received from the trusted party.

Let the adversary $\mathcal{S}$ output $\text{OUT}_{\mathcal{S}}(k, x_1, \ldots, x_n, z)$ that is some arbitrary function of the information gathered during the computation in the ideal world. Let $\text{OUT}_f(k, x_1, \ldots, x_n)$ be the output of the honest parties in the ideal process. Define

$$\text{IDEAL}_{f,\mathcal{S}}(k, x_1, \ldots, x_n, z) = \text{OUT}_f(k, x_1, \ldots, x_n), \text{OUT}_{\mathcal{S}}(k, x_1, \ldots, x_n, z) \tag{2}$$

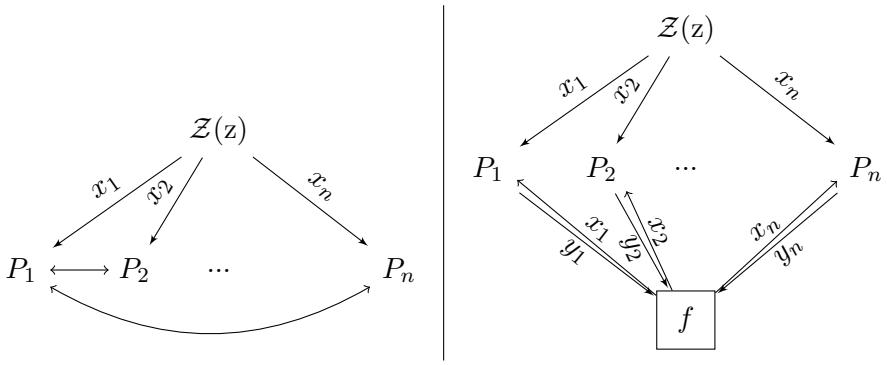
We say that a protocol Π evaluating a function f is secure if any effect achieved by a real-life adversary $\mathcal{A}$ can also be achieved by some ideal world adversary $\mathcal{S}$.

Definition 1[[Can00]] Let f be an n-party function and let Π be a protocol for n parties. We say that a protocol Π computing f is t-private if for all probabilistic polynomial time $\mathcal{A}$ corrupting at most t parties, there exists a polynomial time $\mathcal{S}$ such that $\{\text{REAL}_{\Pi,\mathcal{A}}(k, x_1, \ldots, x_n, z)\}$ and $\{\text{IDEAL}_{f,\mathcal{S}}(k, x_1, \ldots, x_n, z)\}$ are computationally indistinguishable. $\diamondsuit$

Remark As mentioned earlier, it is important that we consider the joint distribution of the honest parties' outputs and the adversary $\mathcal{A}$'s view. Consider the function f that takes no inputs and sends a uniform bit to party P_1. Let protocol Π be such that P_1 chooses a random bit and sends it to P_2 and then outputs the bit. If the joint distribution were not considered, Π would be considered as securely computing f according to the definition above. Similarly, consider the function f that takes no inputs and sends $p.q$ to P_1 and p, q to P_2. Consider the protocol Π where P_1 generates p, q, sends that to P_2 and outputs $p.q$.

Let us consider a pictorial definition where there is an environment $\mathcal{Z}$ with auxiliary input z. The environment $\mathcal{Z}$ sends a bunch of inputs to the parties P_1, $P_2, \ldots, P_n$ and gets back their output and the view. In the real world, a set of t parties are corrupted by an adversary $\mathcal{A}$ and the parties run the protocol Π to determine their outputs. In the ideal world, a set of t parties are controlled by the simulator $\mathcal{S}$ and there is a trusted functionality f that returns the output to each party on receiving their inputs. The environment $\mathcal{Z}$ gets back the view and the parties' outputs in both cases. Informally, the security requirement is that the environment cannot distinguish whether it is in the real world or in the ideal world.

[2]Note that if f is randomized, it is important to consider the joint distribution of honest outputs and the adversary $\mathcal{A}$'s view.



Definition 2 A protocol Π is t-private if for any set of at most t corrupted parties, there exists a polynomial time $\mathcal{S}$ such that

$$|\Pr[\mathcal{Z}(\text{real}, z) = 1] - \Pr[\mathcal{Z}(\text{ideal}_{\mathcal{S}}, z) = 1]| \tag{3}$$

is negligible. $\diamondsuit$

3 Hybrid model

We state a composition theorem that says that we can replace any ideal evaluation calls made by a function Π with sub-protocols that securely evaluate the corresponding functions and this results in a protocol that has the same input-output functionality as Π.

Theorem 1 (Composition [Can00]) *Let Π be a t-private protocol computing f in the g-hybrid model, and let σ be a t-private protocol computing g. Assume Π makes only one call to g per round. Then, the composed protocol Π^{σ} t-privately computes f.*

References

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.