Lecture: Secure Function Evaluation

Hemanta K. Maji

Contents

1 Overview

Suppose Alice has *private input* x and Bob has *private input* y. They are interested in jointly computing the function $f(x, y)$. If this is the only requirement, then it is extremely easy to solve. Alice sends her private input $\tau_1 = x$ to Bob and Bob computes the function $z = f(x, y)$ on their behalf, and sends $\tau_2 = z$ to Alice. There are some problems where $|\tau_1|$ can, in fact, be extremely small as compared to $|\tau_1|$, see for example the *pointer chasing problem*. Problems in *communication complexity* deal with minimizing $|\tau_1|$ such that Alice and Bob can perform this computation. We are not concerned with this aspect of the problem. We permit $|\tau_1\tau_2| = \text{poly}(|x| + |y|)$.

We are interested in computing $f(x, y)$ *securely*, i.e. we want Alice and Bob to learn at most $z = f(x, y)$. How do we make arguments of the form "Parties *at most* learn the value of $z = f(x, y)$"? Towards this end, in this lecture, we shall learn about *simulation*-based security arguments

2 Setting

We are performing:

1. Secure Function Evaluation: We only consider securely evaluating functions, i.e. a entity that obtains input from both parties and provides them with their respective outputs.
2. Symmetric Functions: Both parties receive the same output.
3. Deterministic Functions: The outcome z is a deterministic function of the inputs x and y.

What can our adversaries do:

1. Information-theoretic Setting: All parties have unbounded computational power.
2. Semi-honest Adversaries: Adversaries follow the protocol honest but, at a later time, become curious. So, they revist their *view* attempting to learn additional information that they were not permitted to learn.

3 Warmup: Encryption using One-time Pad

Suppose a party sends two messages m_0 and m_1 using the same random pad. We want to argue that an eavesdropper "only learns" $(m_0 \oplus m_1)$. This is a warmup example to introduce simulation based security arguments.

In the Ideal World, the "Encryption Functionality" takes as input m_0 and m_1 from Alice (the sender) and delivers them to Bob (the receiver). We construct a *simulator* in this ideal world who obtains an *appropriate information* about (m_0, m_1) from the ideal functionality and simulates the view for the eavesdropper.

In the Real World, Alice sends two encryptions of the same message using the shared key $k \sim U$ and sends the ciphertexts $(c_0, c_1) := (m_0 \oplus k, m_1 \oplus k)$ to Bob. The eavesdropper gets to see the messages (c_0, c_1).

The task of the simulator is to ensure that the eavesdropper cannot tell whether it is interacting with the simulator in the Ideal World or listening to the actual transcripts generated in the Real World.

The simulation strategy is as follows. The simulator asks the Ideal Functionality for $\delta = (m_0 \oplus m_1)$. Given this information, it outputs $(c_0, c_1) \sim (U, \delta \oplus U)$.[1]

Now, we need to argue that the eavesdropper cannot distinguish the simulated view from the real view. This follows from the following argument:

$$(m_0 \oplus U, m_1 \oplus U) \equiv (U, (m_0 \oplus m_1) \oplus U) \equiv (U, \delta \oplus U)$$

The leftmost view is the real view, and the rightmost view is the simulated view, and they are identical distributions. Therefore, we have shown that the eavesdropper can find out at most $m_0 \oplus m_1$.

A Note. The simulator could have even asked for (m_0, m_1) from the ideal functionality. In which case, it could again have simulated the views of the adversary successfully. This argument would have shown that the eavesdropper learns at most (m_0, m_1). This is also true!

But this is *wasteful* in the following sense. We know that the eavesdropper can find out $m_0 \oplus m_1$, and this conveys significantly less information as compared to (m_0, m_1). So, our aim as a simulator designer was to use the *least* amount of information to simulate the view of the eavesdropper.

4 Semi-honest Secure Symmetric-Function Evaluation

The Real World Experiment. In the real world version of this game, we have the following participants:

1. Alice: Left Party
2. Bob: Right Party
3. Environment ($\mathcal{Z}$): An external coordinating adversary.

The environment $\mathcal{Z}$ decides whether to corrupt a party or not. If it decides to corrupt a party, then it decides which party to corrupt. The environment decides the private input x of Alice and the private input y of Bob. Alice and Bob execute the protocol Π honestly. All parties return their

[1] Recall that the same variable used twice represents the sample same sample drawn according to the distribution. So, a sample from the distribution $(U, \delta \oplus U)$ is of the form $(u, \delta + u)$, where $u \sim U$. A sample from the distribution $(U, \delta \oplus U')$, on the other hand, would be of the form $(u, \delta \oplus u')$, where u and u' are independent samples drawn according to the uniform distribution.

respective output of the protocol to the environment. The corrupt party returns its complete view to the environment.

So, if the environment corrupts no party, then its view is (x, z_A, y, z_B), where z_A and z_B are outputs of Alice and Bob respectively in the real protocol. The distribution of the view of the environment, in this case, is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})} := (x, Z_A, y, Z_B)$. If the environment decides to corrupt Alice, then its view is (x, V_A, y, z_B). The distribution of the environment's view is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})} := (x, \mathbb{V}_A, y, Z_B)$. Similarly, if the environment decides to corrupt Bob, then the distribution of its view is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})} := (x, Z_A, y, \mathbb{V}_B)$.

The Ideal World Experiment. In the ideal world version of this game, we have the following participants:

1. Alice: Left Party
2. Bob: Right Party
3. Environment ($\mathcal{Z}$): And external coordinate adversary
4. Ideal Functionality ($\mathcal{F}_f$): A functionality that takes input x from Alice and y from Bob and outputs $z = f(x, y)$ to both parties.

The environment remains identical to the environment in the real world. Honest parties reply back with the output they receive. If the environment does not corrupt any party, then its view is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})} := (x, z, y, z)$.

If the environment decides to corrupt Alice, then we get to design a simulator Sim_A that takes over the control of Alice in the ideal world. It takes input x from the environment and forwards x to the external functionality $\mathcal{F}_f$ and receives the output z. Next, the simulator Sim_A prepares a view V_A based on its view, i.e. (x, z). Therefore, we can say that $V_A \sim \mathsf{Sim}_A(x, z)$. Now, the distribution of the view of the environment is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})} := (x, \mathsf{Sim}_A(x, z), y, z)$.

Similarly, if the environment decides to corrupt Bob, then we get to design a simulator Sim_B that takes over the control of Bob in the ideal world. It takes input y from $\mathcal{A}$ and forwards y to $\mathcal{F}_f$, and receives z. Next, it outputs a sample view according to the distribution $\mathsf{Sim}_B(y, z)$. The view of $\mathcal{Z}$, in this case, is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})} := (x, z, y, \mathsf{Sim}_B(y, z))$.

Proving Security. To prove the security of a protocol, we need to show that there exists $(\mathsf{Sim}_A, \mathsf{Sim}_B)$ such that for all $\mathcal{Z}$, the following holds:

$$\mathrm{SD}\left(\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})}, \mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})}\right) \leqslant \varepsilon$$

Then, we say that the protocol has ε *simulation error* or, equivalently, it is $(1-\varepsilon)$ secure. If $\varepsilon = 0$, then we say that the protocol is perfectly secure. We emphasize that the simulators $(\mathsf{Sim}_A, \mathsf{Sim}_B)$ depend on the protocol description, but *not* on $\mathcal{Z}$ because it is quantified after the instantiation of $(\mathsf{Sim}_A, \mathsf{Sim}_B)$. Therefore, intuitively, *even after we publish* the simulators as a proof of security of the protocol, no environment can break its security.

Observe that we need not make any ad hoc enforcement of *correct* evaluation of the output. Correct evaluation follows as a special case of simulation-based security requirement when the environment does not corrupt any party. Because, security, in that case, implies that $\mathsf{SD}\left((x, Z_A, y, Z_B), (x, z, y, z)\right) \leqslant \varepsilon$. Therefore, $\mathsf{SD}\left((Z_A, Z_B), (z, z)\right) \leqslant \varepsilon$, i.e. they agree with the correct output with probability ε.

5 Warmup: Example Protocols

1. XOR: The function is defined as $f(x, y) := x \oplus y$, where $x \in \{0, 1\}$ and $y \in \{0, 1\}$.

   Protocol: Alice sends her bit $\tau_1 = x$ to Bob and Bob replies back with his bit $\tau_2 = y$. Alice outputs $z_A := \tau_1 \oplus \tau_2$. Bob output $z_B := \tau_1 \oplus \tau_2$.

   Security Proof: When no party is corrupt it is easy to see that this protocol has simulation error $\varepsilon = 0$. When Alice is corrupt, $\mathsf{Sim}_A(x, z)$ computes the transcript $(\tau_1 = x, \tau_2 = x \oplus z)$ and sends the view V_A to $\mathcal{Z}$. The simulation error, in this case, is 0 as well. When Bob is corrupt, $\mathsf{Sim}_B(y, z)$ computes the transcript $(\tau_1 = z \oplus x, \tau_2 = y)$ and sends the view V_B to $\mathcal{Z}$. The simulation error, in this case, is 0 as well.

2. MAX: The function is defined as $f(x, y) := \max\{x, y\}$, where $x \in \{0, 2\}$ and $y \in \{1, 3\}$.

   Protocol: If Bob has input $y = 3$, then he sets $\tau_1 = 1$ else he sends $\tau_1 = 0$. If $\tau_1 = 0$ then Alice sends $\tau_2 = 0$, if $x = 0$; otherwise it sends $\tau_2 = 1$ (when $x = 2$. If $\tau_0 = 1$, then Alice does not reply with a message. To compute local output, Alice does the following: If $(\tau_0, \tau_1) = (0, 0)$ then she sets $z_A = 1$, if $(\tau_0, \tau_1) = (0, 1)$ then she sets $z_A = 2$, and if $\tau_0 = 1$ then she sets $z_A = 3$. Similarly, Bob also sets his local input z_B using the identical algorithm.

   Security Proof: When no party is corrupt the protocol is always correct for every input pair (x, y) and, hence, it has simulation error $\varepsilon = 0$. When Alice is corrupted, the simulator $\mathsf{Sim}_A(x, z)$ does the following. If $z = 1$ then output $(\tau_1, \tau_2) = (0, 0)$, if $z = 2$ then output $(\tau_1, \tau_2) = (0, 1)$ and if $z = 3$ then output $\tau_0 = 1$. It can be shown that this is perfectly secure. Simulator for the case of corrupt Bob follows the identical strategy. This establishes that the protocol is perfectly correct.

Question. Can we securely compute OR, i.e. the function $f(x, y) = x \ \vee \ y$, where $x, y \in \{0, 1\}$?

6 Characterization of Securely Realizable Functions

Theorem 1 ([Kus89, Bea89, MPR09, KMQR09])**.** *A secure (deterministic secure) function evaluation f has a semi-honest secure protocol in the information-theoretic setting if and only if f is* decomposable*.*

In Section 7, we shall define *decomposable functions* and then show one side of the proof that decomposable functions have semi-honest secure protocols. In Section 8, we shall show that a special types of *un-decomposable functions*, namely functions with an *OR-minor*, cannot be securely computed. For the full proof of this theorem refer to Appendix C of the full version of [MPR09].

7 Decomposable Functions

The concept of *decomposable functions* was defined by [Kus89, Bea89]. It is recursively defined using the concepts of *valid X-cut* and *valid Y-cut*.

Definition 1 (Valid X-Cut)**.** *A function $f\colon X \times Y \to Z$ has a* valid X-cut *if there exists a set $X' \subset X^*$ such that, for all $y \in Y$, $x' \in X'$ and $x'' \in X \setminus X'$, we have $f(x',y) \neq f(x'',y)$.*

Analogously, we can define the concept of *valid Y-cut.*

Definition 2 (Decomposable Function)**.** *A function $f\colon X \times Y \to Z$ is* decomposable *if:*

1. *The function f is a constant function, or*
2. *There exists a valid X-cut defined by $(X', X \setminus X')$ and the restricted functions $f|_{X' \times Y}$ and $f|_{X\setminus X' \times Y}$ are both decomposable, or*
3. *There exists a valid Y-cut defined by $(Y', Y \setminus Y')$ and the restricted functions $f|_{X \times Y'}$ and $f|_{X \times Y \setminus Y'}$ are both decomposable.*

Note that the functions XOR and MAX defined above are both decomposable. The function XOR has two different witness to its decomposability and the function MAX has a unique witness to its decomposability. But the function OR is not decomposable. In fact, any function f that has $x, x' \in X$ and $y, y' \in Y$ such that $f(x,y) = f(x,y') = f(x',y) \neq f(x',y')$ is said to have an *OR-minor* and is not decomposable.

Question. Are there functions that are non-decomposable but have no OR-minor? Think about whether they can exist for $|Z| = 2, 3, 4, 5, \dots$.

Note that if f is a decomposable function then it has a tree of decomposition, i.e. the witness of decomposition. One can easily show that traversing the tree is a secure protocol for the function f. Prove, inductively, that this gives a secure protocol. Hint: Use inspiration from the two secure protocols written in Section 5.

Question. If a function f has a unique witness for decomposition, then can you argue that *every* secure protocol for f, in essence, is isomorphic to this unique decomposition witness?

8 Impossibility Result for OR

There cannot be a secure protocol for any function f with an OR-minor. Otherwise, we can use the restriction of the function f to $x, x' \in X$ and $y, y' \in Y$ that witness the OR-minor and use it to securely compute OR. In this section, we shall show that there cannot exist any secure protocol for OR. This gives a contradiction; so, f cannot have a secure protocol to begin with.

Some mathematical notation: We represent the set $\{1, \ldots, n\}$ by $[n]$. Suppose there exists a secure protocol for OR with simulation error ε. Then we shall show that ε is at least a constant. To show this, we begin by proving some properties of views generated during protocol execution.

Claim 1 (Rectangle Rule for Transcripts). *Given a transcript τ, for all $x, x' \in X$ and $y, y' \in Y$, we have:*

$$\Pr[\tau|x, y] \cdot \Pr[\tau|x', y'] = \Pr[\tau|x, y'] \cdot \Pr[\tau|x', y]$$

Proof. Let $\tau = \tau_1 \cdots \tau_n$ (assume that n is even without loss of generality). Using Bayes' rule and Markov-chain property of views, respectively, we can write:

$$\begin{aligned}\Pr[\tau|x, y] &= \prod_{i \in [n]} \Pr\left[\tau_i | x, y, \tau_{[i-1]}\right] \\ &= \left(\prod_{\substack{i \in [n] \\ i \text{ is odd}}} \Pr\left[\tau_i | x, \tau_{[i-1]}\right] \right) \times \left(\prod_{\substack{i \in [n] \\ i \text{ is even}}} \Pr\left[\tau_i | y, \tau_{[i-1]}\right] \right) \qquad = \alpha(\tau, x) \cdot \beta(\tau, y)\end{aligned}$$

Now, we can write:

$$\begin{aligned}\Pr[\tau|x, y] \cdot \Pr[\tau|x', y'] &= \alpha(\tau, x)\beta(\tau, y) \cdot \alpha(\tau, x')\beta(\tau, y') \\ &= \alpha(\tau, x)\beta(\tau, y') \cdot \alpha(\tau, x')\beta(\tau, y) \\ &= \Pr[\tau|x, y'] \cdot \Pr[\tau|x', y]\end{aligned}$$

□

Claim 2 (Secure Views). *Let Π be a secure protocol for a function f with simulation error ε. If $f(x, y) = f(x, y')$, then the transcript distributions $(T|x, y)$ and $(T|x, y')$ satisfy:*

$$\mathrm{SD}\left((T|x, y), (T|x, y')\right) \leqslant 2\varepsilon$$

Proof. Consider $\mathcal{Z}$ that corrupts Alice and sends x to her and y to Bob. The distribution of view of this environment in the real world is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})} = (x, \mathbb{V}_A, y, z)$. The distribution of view of this environment in the ideal world is $\mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})} = (x, \mathsf{Sim}_A(x, z), y, z)$. By security we have:

$$\begin{aligned}&\mathrm{SD}\left(\mathbb{V}_{\mathcal{Z}}^{(\mathsf{real})}, \mathbb{V}_{\mathcal{Z}}^{(\mathsf{ideal})}\right) \leqslant \varepsilon \\ \implies \quad &\mathrm{SD}\left((T|x, y), T_A(x, z)\right) \leqslant \varepsilon\end{aligned}$$

Here, $T_A(x, z)$ is the marginal simulated transcript distribution in the distribution $\mathsf{Sim}_A(x, z)$.

Now consider a new environment $\mathcal{Z}$ that is identical as the above mentioned environment except that it sends y' to Bob. Analogous to the above argument, we get $\mathrm{SD}\left((T|x, y'), T_A(x, z)\right) \leqslant \varepsilon$.

By triangle inequality, we get: $\mathrm{SD}\left((T|x, y), (T|x, y')\right) \leqslant 2\varepsilon$. □

Claim 3 (Rectangle Rule Closeness). *Let A, B, C, D be four distributions such that:* $\mathrm{SD}(A, B) \leqslant \delta$, *and* $\mathrm{SD}(A, C) \leqslant \delta$. *Further, the four distributions satisfy the rectangle-rule, that is, $A_i D_i = B_i C_i$, where A_i, B_i, C_i and D_i is the probability of obtaining the sample i according to the distributions A, B, C and D, respectively. Then we have:* $\mathrm{SD}(A, D) \leqslant 2\delta$.

Proof. Try solving this puzzle yourself! In fact something tighter can be proven. Conjecture and prove the tighter bound! ☐

By Claim 2, we have: $\mathrm{SD}\left((T|1,1),(T|1,0)\right) \leqslant 2\varepsilon$ and $\mathrm{SD}\left((T|1,1),(T|0,1)\right) \leqslant 2\varepsilon$. Since transcript distribution follows the rentable rule (by Claim 1), we have: $\mathrm{SD}\left((T|1,1),(T|0,0)\right) \leqslant 4\varepsilon$, by Claim 3.

Note that we have the following Markov chain: $(T|x) \to (\mathbb{V}_A|x) \to (Z_A|x)$. Therefore, we get:

$$\mathrm{SD}\left((Z_A, Z_B|0,0),(Z_A,Z_B|1,1)\right) \leqslant 4\varepsilon \tag{1}$$

By correctness of the protocol, we have $\Pr[Z_A = Z_B = f(x,y)|x,y] \geqslant 1-\varepsilon$. So, we get $\Pr[Z_A = Z_B = 0|0,0] \geqslant 1-\varepsilon$ and $\Pr[Z_A = Z_B = 1|1,1] \geqslant 1-\varepsilon \iff \Pr[Z_A = Z_B = 0|1,1] \leqslant \varepsilon$. Therefore,

$$\mathrm{SD}\left((Z_A, Z_B|0,0),(Z_A,Z_B|1,1)\right) \geqslant 1-2\varepsilon \tag{2}$$

From Equation 1 and Equation 2, it must be the case that $4\varepsilon \geqslant 1-2\varepsilon \iff \varepsilon \geqslant 1/6$. This proves the following theorem:

Theorem 2. *If Π is a semi-honest secure protocol for* OR *with simulation error ε, then $\varepsilon \geqslant 1/6$.*

Question. Can you make this bound tighter? Can you construct a protocol for OR with simulation error matching this bound?

9 Summary

In this lecture, we saw that deterministic symmetric functions are one of the following three types:

1. Decomposable: These functions can be semi-honest securely realized [Kus89, Bea89, MPR09, KMQR09].
2. Non-Decomposable: These functions cannot be semi-honest securely realized.
   (a) Has OR-minor: These functions are the *hardest* functions to realize. If you are given a trusted box that securely implements a function with an OR-minor, then you can securely implement any other function. In this sense, similar to the terminology in computational complexity theory, they are called *complete functions* [Kil91].
   (b) Does not have an OR-minor: These functions are of intermediate complexity, neither securely realizable nor complete.

In the context of symmetric randomized secure function evaluation, very less is known. The characterization of the complete functions was provided by [Kil00]. But the characterization of functions that can be securely realized in the semi-honest setting is an open problem.

References

[Bea89] Donald Beaver. Perfect privacy for two-party protocols. In Joan Feigenbaum and Michael Merritt, editors, *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 65–77. American Mathematical Society, 1989.

[Kil91] Joe Kilian. A general completeness theorem for two-party games. In *23rd Annual ACM Symposium on Theory of Computing*, pages 553–560, New Orleans, Louisiana, USA, May 6–8, 1991. ACM Press.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd Annual ACM Symposium on Theory of Computing*, pages 316–324, Portland, Oregon, USA, May 21–23, 2000. ACM Press.

[KMQR09] Robin Künzler, Jörn Müller-Quade, and Dominik Raub. Secure computability of functions in the IT setting with dishonest majority and applications to long-term security. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 238–255. Springer, Heidelberg, Germany, March 15–17, 2009.

[Kus89] Eyal Kushilevitz. Privacy and communication complexity. In *30th Annual Symposium on Foundations of Computer Science*, pages 416–421, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press.

[MPR09] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 256–273. Springer, Heidelberg, Germany, March 15–17, 2009.