Lecture: Secure 2-party Computation

Hemanta K. Maji

Contents

1 Overview

Alice and Bob are interested in performing general two-party secure computation where their private inputs are, respectively, x and y. We are in the information-theoretic setting and parties are interested in computing a deterministic symmetric function $f(x, y)$. Previously, we had seen that only decomposable functions can be securely computed in the plain model, i.e. parties have access to secure point-to-point communication channels only. In this section, we shall use *oblivious transfer* to perform *any* computation securely against information-theoretic semi-honest adversaries.

2 Interesting Functionalities

First we define some interesting functionalities.

2-choose-1 Bit Oblivious Transfer. This is a two-party functionality, where Alice has inputs $(x_0, x_1) \in \{0,1\}^2$ and Bob has input $c \in \{0,1\}$. The functionality provides x_c to Bob. Alice is referred to as the sender and Bob is referred to as the receiver. Alice's private input is referred to as *input bits* and Bob's private input is referred to as the *choice bit.* This functionality is also referred to as the oblivious transfer functionality or OT functionality.

n-choose-k ℓ-bit String Oblivious Transfer. This is a two-party functionality, where Alice has inputs $(x_0, \dots, x_{n-1}) \in \left(\{0,1\}^\ell\right)^n$ and Bob has input $i \in \{0, \dots, n-1\}$. The functionality provides $x_i \in \{0,1\}^\ell$ to Bob.

Random Oblivious Transfer Correlation. This is an input-less randomized functionality, i.e. parties do not have any inputs. Alice receives $(x_0, x_1) \overset{\$}{\leftarrow} \{0,1\}^2$ and Bob receives (c, x_c), where $c \overset{\$}{\leftarrow} \{0,1\}$.

3 Interesting Protocols

2-choose-1 Bit OT from Random OT Correlation. Suppose the private inputs of Alice is (x_0, x_1) and the private input of Bob is c.

Protocol. Alice and Bob invoke the Random OT correlation to obtain (r_0, r_1) and (b, r_b), respectively. Bob sends $f = b \oplus c$ to Alice. Alice computes $\alpha_0 = m_0 \oplus r_f$ and $\alpha_1 = m_1 \oplus r_{1-f}$ and sends (α_0, α_1) to Bob. Bob computes $z = \alpha_c \oplus r_b$.

Note that $z = m_c$ and this is a two-message protocol.

Intuition. Think of two (one-time pad) encryptions generated using two secret keys r_0 and r_1. Bob knows exactly one secret key r_b and Alice does not know which secret key Bob knows.

If $b = c$, then α_0 is the encryption of the bit x_0 using the secret key r_0, and α_1 is the encryption of the bit x_1 using the secret key r_1. Bob can recover exactly the bit which was encrypted using the secret key r_b.

If $b \neq c$, then α_0 is the encryption of the bit x_0 using the secret key r_1, and α_1 is the encryption of the bit x_1 using the secret key r_0. Bob can recover exactly the bit which was encrypted using the secret key r_b.

n-choose-1 Bit OT from 2-choose-1 Bit OT [BCR86]. Suppose the private input of Alice is $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ and the private input of Bob is $i \in \{0, \ldots, n-1\}$. Here we assume, without loss of generality, that $n = 2^t$, i.e. n is a power of two. The protocol presented below also works for n which is not a power of 2.

Protocol. Consider a binary tree of depth k. The root of the binary tree is named ε. Any internal node v has $v0$ and $v1$ as the left and right child of the node v. The leaves of the binary tree are, respectively, the nodes corresponding to the binary representations of $\{0, \ldots, 2^t - 1\}$.

Corresponding to every edge $e = (v, vb)$, where v is an internal node and $b \in \{0,1\}$, Alice picks a uniform random bit $r_e \xleftarrow{\$} \{0,1\}$.

For every internal node node v, Alice and Bob perform a 2-choose-1 bit OT where Alice uses input $(r_{(v,v0)}, r_{(v,v1)})$. Bob's input is determined as follows: If v is an ancestor of the leaf i, then Bob's input is c, such that the edge (v, vc) lies on the path from the root to the leaf i; otherwise, $c \xleftarrow{\$} \{0,1\}$. Using this OT, Bob obtains $r_{(v,vc)}$.

For every $j \in \{0, \ldots, n-1\}$, define $R(j) = \oplus_{s=1}^{t} r_{(v_{s-1}, v_s)}$, where $v_0 v_1 \ldots v_t$ is the path from the root to the leaf j. Alice sends $\alpha_j = x_j \oplus R(j)$, for all $j \in \{0, \ldots, 2^t - 1\}$, to Bob.

Bob computes $z = \alpha_i + R(i)$. Note that $z = x_i$. We shall argue that Bob does not learn any other Alice input x_j, where $i \neq j \in \{0, \ldots, n-1\}$. Let a be the first node in the the path from the root to the leaf j that it differs from the path from the root to the leaf i. Let a' be the parent of a and $\bar{a}$ be the sibling of a. Note that Bob only got $r_{(a',\bar{a})}$ and not $r_{(a',a)}$, so $R(j)$ is perfectly hidden from Bob. Further, this is only a 2-message protocol.

Intuition. Alice is sends n encryptions with respective secret keys $R(0), \ldots, R(n-1)$. Bob has only one secret key $R(i)$ and hence is able to decrypt only that encryption.

n-choose-1 ℓ-bit String OT from 2-choose-1 ℓ-bit string OT. This is identical to the protocol constructing n-choose-1 bit OT from 2-choose-1 bit OT.

2-choose-1 ℓ-bit string OT from 2-choose-1 bit OT. Alice's private input is $(x_{0,1} \cdots x_{0,\ell}, x_{1,1} \cdots x_{1,\ell}) \in (\{0,1\}^\ell)^2$ and Bob's private input is $c \in \{0,1\}$.

Protocol. For every $i \in [\ell]$, Alice and Bob invoke a 2-choose-1 bit OT with inputs $(x_{0,i}, x_{1,i})$ and c, respectively. Bob gets $x_{c,i}$, for all $i \in [\ell]$. Bob outputs the string $x_{c,1} \cdots x_{c,\ell}$.

3.1 Overall Summary

Starting from $\ell(n-1)$ random OT correlation instances, we can securely compute one n-choose-1 ℓ-bit string OT. Further, all the random OT correlation instances can be computed in an offline phase. These protocols are only 2-message protocols: Bob sends one message and Alice replies with one message. And, these protocols are *perfectly semi-honest secure* even against adversaries with unbounded computational power.

4 Secure 2-party Computation of Small Functions

Let $f \colon X \times Y \to \{0,1\}^\ell$, where $|Y| = n$. Alice's private input is $x \in X$ and Bob's private input is $y \in Y$.

Protocol. Alice constructs strings $\alpha_j = f(x,i)$, for all $j \in \{0,\dots,n-1\}$. Alice uses $(\alpha_0,\dots,\alpha_{n-1})$ as input and Bob uses $i = y$ as input to n-choose-1 ℓ-bit string OT. Bob receives $z = f(x,y)$ as output. Bob sends z to Alice.

Note. Starting from $\ell(n-1)$ random OT correlations, this is a 3-message protocol. The first two rounds are used to implement the n-choose-1 ℓ-bit string OT. And, in the third message, Bob sends the output z to Alice.

5 Yao's Garbled Circuit Construction [Yao82]

Assumption. In this presentation, we will assume the existence of a special encryption scheme with the following properties:

1. Given two keys (k,k') we can encrypt a message m efficiently and produce the ciphertext $c = \mathsf{Enc}_{(k,k')}(m)$.
2. Given two keys (t,t') the decryption algorithm has the following property: if $(k,k') \neq (t,t')$ then $\mathsf{Dec}_{(t,t')}(c) = \perp$; else $\mathsf{Dec}_{(t,t')}(c) = m$.

Notation. Suppose Alice and Bob are interested in computing a function f represented by a circuit C. The input wires corresponding to Alice inputs are $\{0,\dots,t-1\}$ and the input wires corresponding to Bob inputs are $\{k,\dots,2t-1\}$. Let $\{u+1,\dots,u+\ell\}$ be the output wires of the circuit. For any gate G of the circuit, let u_G and v_G represent the indices of the input wires and w_G represent the index of the output wire. Let W be the set of all input, internal and output wires of the circuit.

Protocol. Alice picks two random keys $K_{w,0}$ and $K_{w,1}$, for all wires $w \in W$. For every gate G of the circuit, she computes the following strings, for every $a,b \in \{0,1\}$:

$$\alpha_{a,b} = \mathsf{Enc}_{(K_{u_G,a},K_{v_G,b})}(K_{w_G,G(a,b)})$$

Alice picks a random permutation π_G of four elements and computes the string $\alpha_G = \pi_G(\alpha_{0,0},\alpha_{0,1},\alpha_{1,0},\alpha_{1,1})$. Alice sends α_G to Bob for all gates G. This is called the *garbled circuit*.

For every input wire $w \in \{0, \ldots, t-1\}$, Alice sends the key K_{w,x_w} to Bob. For every input wire $w \in \{t, \ldots, 2t-1\}$, Alice performs a 2-choose-1 string OT with inputs $(K_{w,0}, K_{w,1})$ and Bob uses $c = y_{w-t}$ as input and obtains the key $K_{w,y_{w-t}}$.

For every input wire $w \in \{0, \ldots, 2t-1\}$, Bob has a key K_w. Bob iteratively performs the following computation. For any internal gate G, if it has K_{u_G} and K_{v_G}, it tries to decode all four entries of α_G. Three of them output $\perp$ and one outputs K_{w_G}. In this manner, Bob computes K_w, for all $w \in \{u+1, \ldots, u+\ell\}$. Bob sends $\{K_w : w \in \{u+1, \ldots, u+\ell\}\}$ to Alice.

Let $K_{w,\theta_w} = K_w$, for every $w \in \{u+1, \ldots, u+\ell\}$. Alice sends $\{\theta_w : w \in \{u+1, \ldots, u+\ell\}\}$ to Bob.

Alice and Bob output $\theta_{u+1} \cdots \theta_{u+\ell}$ as the output of the circuit.

Note. If all the random OT correlation instances are generated in the offline phase, then this is a 4-message protocol (starting with Bob). The first two messages are used to perform key transfers. The garbled circuit can be send by Alice in the second message. The last two messages are used by Alice and Bob to obtain the output.

6 GMW Construction [GMW87]

We use the same circuit notation as in Section 5.

Protocol. For all wires $w \in \{0, \ldots, t-1\}$, Alice picks random $r_w \xleftarrow{\$} \{0,1\}$ and computes $s_w = x_w \oplus r_w$. Alice sends s_w to Bob.

For all wires $w \in \{t, \ldots, 2t-1\}$, Bob picks random $s_w \xleftarrow{\$} \{0,1\}$ and computes $r_w = y_{w-t} \oplus s_w$. Bob sends r_w to Alice.

The following step is performed iteratively. Let G be an internal gate such that Alice already has (r_{u_G}, r_{v_G}), and Bob already has (s_{u_G}, s_{v_G}). Define the following function f_G:

$$f_G(r_{u_G}, r_{v_G}, r_{w_G}, s_{u_G}, s_{v_G}) := r_{w_G} \oplus G(\ (r_{u_G} \oplus s_{u_G})\ ,\ (r_{v_G} \oplus s_{v_G})\ )$$

Alice uses $(r_{u_G}, r_{v_G}, r_{w_G} \xleftarrow{\$} \{0,1\})$ as input and Bob uses (s_{u_G}, s_{v_G}) as input to this function. Parties use the protocol in Section 4 to perform this computation where only Bob obtains the output $z = f_G(r_{u_G}, r_{v_G}, r_{w_G}, s_{u_G}, s_{v_G})$. Bob defines $s_{w_G} = z$.

For all output wires $w \in \{u+1, \ldots, u+\ell\}$, Alice sends r_w to Bob and Bob sends s_w to Alice. Both parties can compute $\theta_w = r_w \oplus s_w$. Both parties output $\theta_{u+1} \cdots \theta_{u+\ell}$ as the output of the circuit.

Note. The round complexity of the protocol depends on the depth of the circuit. The communication complexity of the protocol depends on the size of the circuit. This protocol is a perfectly secure protocol if random OT correlations are provided.

Further optimizations are possible. If the circuit only uses XOR, NOT, AND gates, then f_G needs random OT correlations to be evaluated only when G is an AND gate. If G is NOT or XOR gates then f_G can be computed securely without using random OT correlations.

7 Security Proofs

Students are encouraged to prove the security of the protocols on their own. The instructor will help the students develop these proofs in one-on-one meetings.

References

[BCR86] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. Information theoretic reductions among disclosure problems. In *27th Annual Symposium on Foundations of Computer Science*, pages 168–173, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City,, New York, USA, May 25–27, 1987. ACM Press.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.