Lecture 3

Gidon Rosalki
2026-05-06

Notice: If you find any mistakes, please open an issue in the github repository

1. Secure Computation

Here we will discuss a problem, where instead of one side having more information than the other, the two sides are relatively symmetrical, and are attempting to perform a shared computation. Alice and Bob would like to know which one is richer. Must they reveal their inputs? They could both say how much money they each have, but that is *boring*. We can instead solve this without leaking any information. Much like Zero Knowledge, this does not sound possible, but let us be surprised. Let us say that Alice has input x, and Bob input y. We want to create a function

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

Alice and Bob want to jointly compute $f(x,y)$. We have a few adversarial models:

- **Semi-honest**: Parties that follow the protocol's instructions, but try to learn additional information
- **Malicious**: Parties that may deviate from the protocol's instructions, in order to learn additional information

2. Security In The Semi-Honest Model

2.1. Notation

- $\langle \mathcal{A}(x), \mathcal{B}(y) \rangle (1^n)$ is the distribution of the transcript on inputs x and y
- $\text{out}_{\mathcal{A}}[\langle \mathcal{A}(x), \mathcal{B}(y) \rangle (1^n)]$ is the distribution of the transcript on inputs x and y

2.2. Security

Definition An efficient protocol $\langle \mathcal{A}, \mathcal{B} \rangle$ **securely computes** a deterministic function $f = (f_{\mathcal{A}}, f_{\mathcal{B}})$ in the semi-honest model if there exist expected PPT simulators $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{B}}$, such that for every $x, y \in \{0,1\}^*$ the following hold:

- Correctness: $\Pr[(\text{out}_{\mathcal{A}}[\langle \mathcal{A}(x), \mathcal{B}(y) \rangle], \text{out}_{\mathcal{B}}[\langle \mathcal{A}(x), \mathcal{B}(y) \rangle]) = f(x,y)] = 1$
- Security against a semi-honest Alice: $\mathcal{S}_{\mathcal{A}}(x, f_{\mathcal{A}}(x,y)) \overset{c}{\approx} \text{view}_{\mathcal{A}}[\langle \mathcal{A}(x), \mathcal{B}(y) \rangle]$
- Security against a semi-honest Bob: $\mathcal{S}_{\mathcal{B}}(y, f_{\mathcal{B}}(x,y)) \overset{c}{\approx} \text{view}_{\mathcal{B}}[\langle \mathcal{A}(x), \mathcal{B}(y) \rangle]$

Theorem (Yao 1986): Assuming the existence of a secure Oblivious Transfer (OT) protocol in the semi honest model, any efficiently computable deterministic two-output function can be securely computed in the semi-honest model.

This was a groundbreaking result initiating research on secure computation, inspired fundamental protocols for the multi-party & malicious models, and various applications beyond secure computation. It quickly became one of the modern cryptographic cornerstones.

2.3. Exercise

Let us consider Alice, and Bob, both with 1 bit $b_A, b_B \in \{0,1\}$. Let us consider that they want to securely compute $f(b_A, b_B)$, where $f =$AND. We need to make the following simulators $\text{view}_A \approx S_{A(b_A \wedge b_B, b_A)}$, and $\text{view}_A \approx S_{B(b_A \wedge b_B, b_B)}$.

Prove that there is no such protocol that enables perfect secrecy for this problem: Let us assume towards contradiction that there is such a protocol. Let us begin by assuming $b_A = 1$. So, from the output, we can learn b_B. The transcript result is either 0 or 1. We will note that this is because there is no intersection between the transcripts for when $b_B = 0$ and when $b_B = 1$. However, if $b_A = 0$, then the output is *always* 0, and if there is a secure protocol, then Alice learns nothing about the bit from Bob, so the results of the transcript are exactly equal no matter what Bob's input was. So, if $b_B = 0$, then Bob

may look at all the possible transcripts, and view from where it came, and so may compute Alice's bit. This is not efficient, since you must run over all the possible transcripts, but it does work.

3. Computing Any Function In The Semi-Honest Model

3.1. Main Tool: Obivious Transfer (OT)

3.1.1. 1 out of 2 oblivious transfer

- Input: $\mathcal{A}$ holds (x_0, x_1), $\mathcal{B}$ holds $\sigma \in \{0,1\}$
- Output: $\mathcal{B}$ learns x_σ, (does **not** learn $x_{1-\sigma}$), $\mathcal{A}$ learns nothing

We want to create a protocol which creates the above. You may consider this to be a form of computing "if". It also turns out that this is the hardest function, and if you can make "if", you can make everything.

To do this, we will use **trapdoor permutations** (TDPs): One way permutations which can be efficiently inverted using a trapdoor (RSA is an example).
Definition A tuple (Gen, Sam, f, f^{-1}) of PPT algorithms is a **trapdoor permutation family** if:

- Gen (1^n) outputs pairs (I, td) defining a domain $\mathcal{D}_I$
- $(\text{Gen}_1, \text{Samp}, f)$ is a one way permutation family, where Gen_1

Intuition: Function that is easy to compute forwards, very difficult to compute backwards, *unless* you have the trapdoor.

3.1.2. If

So, using this we may create the following protocol (Alice on the left, Bob on the right).

$$
\begin{aligned}
\text{Alice input: } (x_0, x_1) \mid & \text{ Bob input: } \sigma \in \{0,1\} \\
(I, td) \leftarrow \text{Gen}(1^n) \mid & \\
I \longrightarrow & \\
\mid & \; w_\sigma = f_I(v_\sigma), w_{1-\sigma} \leftarrow \mathcal{D}_I : v_\sigma \leftarrow \mathcal{D}_I \\
v_0 = f_{td}^{-1}(w_0) \mid & \\
b_0 = h(v_0) \oplus x_0 \mid & \\
v_1 = f_{td}^{-1}(w_1) \mid & \\
b_1 = h(v_1) \oplus x_1 \mid & \\
(b_0, b_1) \longrightarrow & \\
\mid & \; x_\sigma = b_\sigma \oplus h(v_\sigma)
\end{aligned}
$$

Theorem Assuming that (Gen, f, f^{-1}) is an enhanced TDP family, with a hard core predicate h, the above is a secure OT protocol in the semi-honest model.

Correctness: is easy, just follow the XORs.
Security: We need to make simulators for Alice, and for Bob. Let us construct $\mathcal{S}_\mathcal{A}(x_0, x_1)$:

- Sample $r_\mathcal{A} \leftarrow \{0,1\}^*$, and compute $(I, td) = \text{Gen}(1^n; r_\mathcal{A})$
- Sample $w_0 \leftarrow \mathcal{D}_I$ and $w_1 \leftarrow \mathcal{D}_I$ independently
- Output $((x_0, x_1, r_\mathcal{A}, w_0, w_1))$

$\mathcal{S}_\mathcal{B}(\sigma, x_\sigma)$:

- Sample $r_\mathcal{A} \leftarrow \{0,1\}^*$, and compute $(I, td) = \text{Gen}(1^n; r_\mathcal{A})$
- Sample $r_\mathcal{B} = (r_0, r_1) \leftarrow \{0,1\}^*$
- Let $v_\sigma = \mathcal{D}_I(r_\sigma), w_\sigma = f_I(v_\sigma), w_{1-\sigma} = \mathcal{D}_I(r_{1-\sigma})$
- Let $b_\sigma = h(v_\sigma) \oplus x_\sigma$, and sample $b_{1-\sigma} \leftarrow \{0,1\}$
- Output $(\sigma, r_\mathcal{B})$

We can also construct from DDH, rather than from RSA (in the style of Michael Rabin at HUJI at the end of the 70s): We may construct **Random OT**, which is a special case of a real OT, where we have a sender S, with input m_0, m_1, s_0, s_1, and the receiver $b \in \{0,1\}, r \in \{0,1\}, s_r$, where s_0, s_1, r are all

random. The idea is that if we know to do OT for these random values, then we know to do it for anything.

$$\begin{aligned} S \mid R \\ m_0, m_1 \mid b \in \{0,1\} \\ s_0, s_1 \mid r \in \{0,1\}, s_r \\ \longleftarrow b \oplus r \\ c_0 = m_0 \oplus s_{b \oplus r} \mid \\ c_1 = m_1 \oplus s_{b \oplus r \oplus 1} \mid \\ (c_0, c_1) \longrightarrow \\ \mid m_b = c_b \oplus s_r \end{aligned}$$

So, we are assuming that we begin from the situation where S has s_0, s_1, and R has $r \in \{0,1\}$, and the appropriate s_r. We have thus changed the problem from OT to beginning in that situation.

So, how do we construct the situation where we have the shared information?

$$\begin{aligned} S \mid R \\ x \in \mathbb{Z}_p \mid \\ g^x \longrightarrow y \in \mathbb{Z}_p \\ \mid p_{k_r} = g^y \\ \mid p_{k_{1-r}} = g^{x-y} \\ \longleftarrow p_{k_0}, p_{k_1} \\ \left(\text{Enc}_{p_{k_1}}(s_0), \text{Enc}_{p_{k_2}}(s_1)\right) \longrightarrow \end{aligned}$$

Since R knows y, it may decrypt $\text{Enc}_{p_{k_r}}(s_r)$, and only that one, to receive s_r.

3.2. Additional Tool "Special" Encryption

We will use a CPA-secure private key encryption scheme (G, E, D) with two additional properties. Firstly some notation:

$$\text{Range}_n(k) \stackrel{\text{def}}{=} \{E_k(x) : x \in \{0,1\}^n\}$$

Property 1: Elusive range. For every PPT A, there exists a negligible function $v(n)$ such that

$$\Pr_{k \leftarrow G(1^n)}\left[A(1^n) \in \text{Range}_{n(k)}\right] \leq v(n)$$

Property 2: Efficiently verifiable range: there exists a PPT algorithm M such that $M(1^n, k, c) = 1$ **if and only if** $c \in \text{Range}_n(k)$.

Construction: Let F be a PRF, where $F_k : \{0,1\}^n \to \{0,1\}^{2n}$ for $k \in \{0,1\}^n$.

$$E_k(x; r) = \left(r, F_{k(r)} \oplus x0^n\right)$$

So given the ciphertext, and the key, we can verify that the ciphertext is in fact a valid ciphertext.

3.3. Garbling Boolean Circuits

Input: Boolean circuit $C : \{0,1\}^l \to \{0,1\}$
Output: Garbled circuit $G(C)$, and input labels $\left\{(L_1^0, L_1^1), ..., \left(L_l^0, L_l^1\right)\right\}$

Our goal is given $G(C)$ and

Let us begin with a single OR gate. We want to assign to random label (L_w^0, L_w^1) to each wire w:

- $L_w^0 \leftarrow G(1^n)$ corresponds to value 0 on wire w
- $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w

For each gate g, construct a doubly encrypted translation table with randomly permuted rows. So for OR, we have $(L_u^0, L_u^1), (L_v^0, L_v^1), (L_w^0, L_w^1)$, where w is the output wire, and u, v are the input wires. We may now construct the OR table:

L_u^0	L_v^0	$\text{Enc}_{L_u^0}\left(\text{Enc}_{L_v^0}(L_w^0)\right)$
L_u^0	L_v^1	$\text{Enc}_{L_u^0}\left(\text{Enc}_{L_v^1}(L_w^1)\right)$
L_u^1	L_v^0	$\text{Enc}_{L_u^1}\left(\text{Enc}_{L_v^0}(L_w^1)\right)$
L_u^1	L_v^1	$\text{Enc}_{L_u^1}\left(\text{Enc}_{L_v^1}(L_w^1)\right)$

So, we need a system such that if Alice has input u, and Bob input v, then together they may find the result L_w^σ for $\sigma = f(u, v)$, without finding out the other value. We cannot just use this table, since it would leak the other value, but we could randomise the order, have both try and decode the entire set of 4, and one of them will be a correct decryption, from which they may then check the actual value. We now just need to create the table.

So:

- Alice creates $L_u^0, L_u^1, L_v^0, L_v^1$

3.4. Yao's Garbled Circuit Protocol

- Alice has input $x \in \{0,1\}^n$, computes $G(C)$, and labels $\left\{(L_i^0, L_i^1)\right\}_{i \in [2n]}$.
- Bob has input $y \in \{0,1\}^n$
- Alice sends Bob the garbled circuit $G(C)$, and input labels $L_1^{x_1}, ..., L_n^{x_n}$ for x
- Alice and Bob perform OT for each $i \in [n]$, in parallel:
  - Alice's input: (L_{n+i}^0, L_{n+i}^1)
  - Bob's input: y_i
- Bob computes $C(x, y)$, using $G(C)$, and $L_1^{x_1}, ..., L_n^{x_n}, L_{n+1}^{y_1}, ..., L_{2n}^{y_n}$
- Bob sends Alice $C(x, y)$

Let us approximate the security proof. We need two simulators:

- $S_A(f(x,y), x) \to \text{view}_A$
  - Except for the OTs, Alice's view can be generated given x, and $C(x, y)$
- $S_B(f(x,y), y) \to \text{view}_B$
  - Bob expects $G(C)$, together with $2n$ input labels on which $G(C)$ evaluates to $C(x, y)$
  - Problem: S_B can generate $G(C)$, but will not know which input labels correspond to x
  - Solution: Generate "fake" $\tilde{G}(C)$
  - Steps:

1. h

Efficiency: The garbled circuit is about $4 \cdot |C|$, and then we perform OT for each of the tables. There are around 5 communication steps, (the best for OT is 3), so… it's a lot. There are lots of papers on making each part more efficient. There are half gates, which mean we can encode each table with 2 rows, rather than 4.

3.5. Universal Circuit

There exists a universal circuit (like a UTM), which given a circuit f, and an input x, will compute $f(x)$. Fun fact, for $n = |f|$, $|u| \approx n \log n$.

4. Forcing Semi-Honest Behaviour

What if one of our sides is not honest? We will have them begin with a commitment for their input, and then at every step related to their inputs, we prove with ZK that this is in fact based off their input. This has given us the ability to convert any semi-honest method, and convert it to malicious, simply by using ZK as a black box at every step, to prove that each step was carried out correctly and honestly.