Today: Secure computation

So far: Our focus was on securing communication:

- Encryption (for privacy) & signatures for authenticity.

From now on: We will move towards securing computation.

The setting: There are $n$ parties $P_1, \ldots, P_n$ with private inputs $x_1, \ldots, x_n$, respectively.

They wish to compute a predetermined function $f$ on their joint inputs: $(Z_1, \ldots, Z_n) = f(x_1, \ldots, x_n)$.

Party $P_i$ should learn $Z_i$, but nothing more.

Today: We focus on the 2-party setting ($n=2$).

We will consider two adversarial models:

1. Honest-but-Curious: Both parties follow the
protocol, but try to learn as much as possible from the (honest) transcript.

2. **Malicious Adversaries**: An adversarial party can deviate arbitrarily from the protocol.

* In the multi-party setting \((n \geq 3)\) further distinguish between polynomially bounded adversaries & all powerful adversaries.

In the 2-party setting achieving information theoretic security is impossible!

**Today**: We focus on Honest-but-curious (HBC) adversaries.

How do we define security?

Intuitively: The only information revealed by an (honest) transcript is the output.

A similar security requirement was formalized in the context of zero knowledge.
As in ZK, we formalize security via the simulation paradigm.

**Def:** A 2-party protocol $\pi = (A, B)$ computes $f = (f_1, f_2)$ securely against HBC adversaries if there exists PPT simulators $S_A, S_B$ such that:

$\forall n \in \mathbb{N} \quad \forall x, y \in \{0, 1\}^n$

$\left( \text{View}_{A(x)}(\pi(x, y)), \text{out}_A \right) \equiv \left( S_A(x, f(x, y)), f(x, y) \right)$

$\left( \text{View}_{B(y)}(\pi(x, y)), \text{out}_B \right) \equiv \left( S_B(y, f(x, y)), f_2(x, y) \right)$

the entire view of B after running $\pi(x, y)$

This definition is very similar to that of ZK.

Note that we include the output in the indist. above. This is to ensure correctness.

An alternative, yet equivalent def. is via Real/Ideal world.
Real world: \( \Pi \) is run on the inputs \( x, y \).

Ideal world: \( A \) & \( B \) do not run \( \Pi \).

Rather, they send their inputs to a trusted party (or ideal functionality).

The security guarantee is that whatever \( A \) (resp. \( B \)) learns in the "real world" can be efficiently simulated given only the information that \( A \) (resp. \( B \)) learned in the ideal world, where the simulated view is required to be "computationally indistinguishable from the real view."
One can also consider security in the case where the parties may deviate from the protocol (malicious parties): Stay tuned!

Q: For which functions $f$ do we have a secure protocol against HBC adversaries?

Thm: If oblivious Transfer (OT) exists then any poly-size function $f=(f_1, f_2)$ has a 2-party protocol that is secure against HBC adversaries.

**Oblivious Transfer**

Oblivious Transfer is a 2-party protocol between a sender $S$ and a receiver $R$.

The sender $S$ has two inputs $x_0, x_1$, & the receiver has a bit $b \in \{0, 1\}$.

After the protocol is executed, the receiver
learns $x_b$ and nothing else, and the sender learns nothing.

Namely: $\text{protocol}(x_0, x_1, b) = (1, x_b)$

An OT protocol is a secure 2-party protocol for OT.

Namely: The theorem above says that for is complete for 2-party comp. (in the HBC setting).

**OT Construction:**

We will next construct an OT protocol that is secure against HBC adv., assuming the existence of public-key encryption with the following property:

The encryption scheme $(\text{Gen, Enc, Dec})$ has an oblivious key generation alg. $\text{OGen}$.
which samples only a public key s.t. \( \forall m_0, m_1 \in M \)

\[
(\gamma, \text{PK} = \text{OGen}(\gamma), \text{Enc}(m_0)) \quad \&
\]

\[
(\gamma, \text{PK} = \text{OGen}(\gamma), \text{Enc}(m_1)).
\]

\[
\{ \text{PK} : \text{PK} \leftarrow \text{OGen}(\Lambda^n) \} \cong \{ \text{PK} : (\text{PK}, \text{SK}) \leftarrow \text{Gen}(\Lambda^n) \}
\]

**Examples:**

**ElGamal:** \( \text{PK} = (g, g^s) \quad \text{SK} = s \)

\( \text{OGen} \): Samples a random element from the group

**LWE:** \( \text{PK} = (A, As + e) \quad \text{SK} = s \)

\( \text{OGen} \): Samples randomly \( (A, u) \)

**Construction:**

\[
S \quad (x_0, x_1)
\]

\[
R \quad _b
\]

\[
\text{PK}_0, \text{PK}_1 \quad (\text{PK}_b, \text{SK}_b) \leftarrow \text{Gen}(\Lambda^n)
\]

\[
\text{PK}_{1-b} \leftarrow \text{OGen}(\Lambda^n)
\]

\[
\text{Co} = \text{Enc}_{\text{PK}_0}(x_0)
\]

\[
C_1 = \text{Enc}_{\text{PK}_1}(x_1) \quad \text{Co}, C_1 \rightarrow
\]
**Claim**: This OT protocol is secure against HBC adversaries if the underlying encryption scheme is semantically secure &

\[ \{ \text{PK} : \text{PK} \in \text{OGen}(\dagger^n) \} \equiv \{ \text{PK} : (\text{PK}, \text{SK}) \in \text{OGen}(\dagger^n) \} \]

**Pf**: \( S_{\text{sender}}( (x_0, x_1, \dagger^n), 1) \) : Sample \( \text{PK}_0, \text{PK}_1 \in \text{OGen}(\dagger^n) \)

and output \( (\text{PK}_0, \text{PK}_1) \)

Note that by \( \oplus \),

\[ \{ S_{\text{sender}}( (x_0, x_1, \dagger^n), 1) \}, 1 \} \equiv \{ \text{View}_{\text{sender}}, 1 \} \]

\( x_0, x_1, \dagger^n, \text{PK}_0, \text{PK}_1, \)

\( \text{PK}_{1-b} \in \text{OGen}(\dagger^n) \)

\( (\text{PK}_0, \text{SK}_b) \in \text{Gen}(\dagger^n, \Gamma_0) \) for random \( \Gamma_0 \)

\( S_{\text{receiver}}( b, x_b, \dagger^n) : \text{Generate} \ (\text{PK}_0, \text{PK}_1) \)

honestly. \( (\text{PK}_{1-b} = \text{OGen}(\dagger^n, \Gamma_{1-b}) \) for random \( \Gamma_{1-b} \)

\( (\text{PK}_0, \text{SK}_b) = \text{Gen}(\dagger^n, \Gamma_b) \) for random \( \Gamma_b \)

Generate \( C_b = \text{Enc}_{\text{PK}_b}(x_b) \)

\( C_{1-b} = \text{Enc}_{\text{PK}_b}(0) \)
Output \( (b, \tau_0, \tau_1, \text{CT}_0, \text{CT}_1) \)

\[ \frac{PK_0}{PK_1} \]

\[ \{ \text{S\_receiver}(b, x_b, 1^n), x_b^2 \} \equiv \{ \text{View}_{\text{receiver}(b)}(b, x_0, x_1, 1^n), x_b \} \]

Follows from semantic security

**Remark:** Security against HBC seems very weak! Why would R choose \( PK_{1-b} \) obliviously?

The reason we consider HBC is because we will later show how to convert HBC secure protocols into malicious secure ones, using ZK (and more).

**Next lecture:** A secure OT protocol (against HBC)

implies secure protocol \( \oplus \) poly size \( s=(s_1, s_2) \) (against HBC)