Lecture 9: Multiparty Computation
Plan

* Multi-party computation
  - Concept
  - Applications
* Stretch break
* Simulation-based defi
* Breakout rooms

Logistics

- Unexpected Tuesday class next week
- Guest lecture next week from Susan MacKay on privacy challenges for journalists. HU 3 due next Friday
- OH today
Multiparty Computation

As far as I know, this idea goes back to Yao (1986)...

Presented idea in conf talk, but not in paper

Historical aside: Yao is a legendary computer scientist
- Two phds: Physics & CS
- A number of papers that launched entire fields
  Comm complexity, cell probe, MRC, ...
- It’s worth checking out some of these papers
e.g. “Should tables be sorted?”
Yao's "Millionaire's Problem" (1986)

somewhere on a yacht in French polynesia....

- Each player $i$ has private input $x_i \in \{0,1\}^n$
- Public function $f: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$

Intuitively, both parties learn $f(x_{mark}, x_{oprah})$, but they learn "nothing else" about the other's input.

as always, devil is in the details
Multiparty Computation, Generally GMW, ...

Parties \( P_1, \ldots, P_n \)

Each party \( P_i \) has a secret input \( x_i \in \{0,1\}^\ell \)

A public function \( f : \{0,1\}^\ell \times \cdots \times \{0,1\}^\ell \to \{0,1\}^\ell \) \( n \) times

Goal: Jointly compute \( f(x_1, \ldots, x_n) \) while "leaking nothing" to other parties about input.

A few things to notice:

- Easy to extend to many-bit output
  
  (Black-box: run many times in parallel)

- Each party can get private output

\[
f(x_1, \ldots, x_n) \to (y_1, \ldots, y_n)
\]

\[
\hat{f}((x_1, r_1), \ldots, (x_n, r_n)) \to (y_1 + r_1, \ldots, y_n + r_n)
\]

Private random blinding

One-time pad encryption
MPC is powerful/general

PIR:

\[
\text{Client (i) } \xrightarrow{\text{DB}_i} \text{Server (DB)} \]

+ Restriction on comm complexity.

\[ S(i, DB) := \{ \text{output ith bit of DB} \} \]

Proofs:

\[ N = p \cdot q \]

\[ \text{Factors of } N \]

\[ S(N, p, q) = \{ p, q \div N \} \]

Privacy:

\[ S(\text{data}_1, \ldots, \text{data}_n) = \{ \text{train model on data} \} \]
There are serious problems with making this work in practice (Bluetooth, incentives, etc.) but I want to raise this example b/c it highlights both strengths & weaknesses of MPC.

- Every phone has:
  - a list of (ID, timestamp) pairs
  - a timestamp (if any) of when they tested positive for COVID.

- Every hour, every phone wants to know:
  "Did I have contact with other phone whose owner tested positive within a time window that would require me to quarantine?"

→ Learn "nothing else."
Types of MPC

You should remember: There are many types.

Things to specify:

- **How many parties?**
  - Two, three, many?
  - Interesting special cases for 2, 3.

- **How many parties are adversarial?**
  - ... of \( n \)
  - \(< \frac{n}{3}\) "Byzantine" setting
  - \(< \frac{n}{2}\) "Honest majority"
  - \(< n - 1\) "Dishonest majority"

- **What type of misbehavior can adversary do?**
  - Collude and try to learn something while still following the protocol "semi-honest"
  - Arbitrarily deviate from the protocol "malicious"

- **When does the adversary corrupt the parties?**
  - Before protocol begins? "Static"
  - While protocol runs? "Adaptive"

- **What type of security do we want?**
  - No assumptions "info theoretic"
  - Crypto assumptions "computational"

- **Do we want any "fairness" guarantee?**
  - "Fair": all parties get output at same time
  - "Sec w/ abort": adv may get output, but parties don't
Types of MPC

Not all combinations of these properties are feasible.

- e.g. Fairness requires an honest majority (Cleve 1986)

Good things to know:

Computational security:

- GMW protocol: $n$ parties, $\leq n-1$ malicious $\leftrightarrow$ abort (uses auth channels)  
  - GMW'87

- Yao's Garbled Ckts: 2 parties, better constants

Info theoretic:

- BGW protocol: $n$ parties $<\frac{1}{2}$ malicious (uses auth channels)  
  - BGW’88, CCD’88
  
  or $<\frac{1}{3}$ malicious  
  - (with broadcast)
Complexity Metrics

* Computational costs for parties
* Communication (# bits exchanged)
* Round complexity (# sequential msgs)

Problem: Most MPC protocols are not very concretely efficient. Let's more when we see e.g. protocol.
Stretch

Break
Key Idea: Simulation

The design of MPC uses (arguably) the most beautiful idea in modern cryptography: using simulation to capture the idea of “learning nothing.”

Idea: You have “learned nothing” from an interaction if you can write down a transcript of your interaction w/o actually interacting.

My dad in sixth grade:

- Q: How was school, son?
- A: Fire
- Q: What happened in there today?
- A: Not much

Dad can simulate a transcript of our interaction on his own...
Another example you see...

Reporter: “Was your govt responsible for....”

Govt official: “We cannot confirm or deny...”

Q: Have you been...
A: We cannot confirm...

Can simulate!
Might as well not have asked!

[There's a life lesson here about how to ask good questions...]
In MPC context we want to capture the idea of "learning nothing" except $f(x_1, \ldots, x_n)$.

$\Rightarrow$ Adversary should be able to simulate its "view" of the protocol given only $f(x_1, \ldots, x_n)$.

"View" of a party in a protocol is
- input
- randomness
- msgs sent/received.

**Definition**

An MPC protocol is secure against semi-honest adversaries if there exists an efficient simulator $\text{Sim}$ such that for every subset $C \subseteq [n]$ (e.g. $|C| < n/2$) and every choice of inputs $x_1, \ldots, x_n$,

$$\left\{ \text{Views of parties in } C \right\} \subseteq \left\{ \text{Sim}(C, \{x_i : i \in C\}) \right\} \subseteq \left\{ f(x_1, \ldots, x_n) \right\}$$

Key idea: Input to simulator captures what leaks to adv.
Real World

$S(x_1, \ldots, x_4)$

Ideal World

$\text{Sim} \left( \{x_1, x_3, x_4\}, \{x_3, x_4, S(x_1, \ldots, x_4)\} \right)$

Views capture everything adv learns during protocol execution.

Can simulate views $\Rightarrow$ "learns nothing except $S(x_1, \ldots, x_n)$"
Breakout Rooms
Definition gets more complicated

- in malicious model
- when $f(\cdot)$ is a stateful functionality
- when many instances of protocol may run concurrently.

Important sanity checks?

When you see a “privacy-preserving protocol” ask yourself:

* What flavor of MPC is at work? (type of adv, type of guarantee)

* What is the ideal functionality $f(\cdot)$ computing?

* Is protocol leaks “nothing more than $f(x, \ldots, x_r)$” is that good enough?

EW example