Secure Multi-Party Computation

Lecture 11
Can we have an auction without an auctioneer?!

- Declared winning bid should be correct
- Only the winner and winning bid should be revealed
Hospitals which can’t share their patient records with anyone
But want to data-mine on combined data

Using data without sharing?
A general problem

To compute a function of private inputs without revealing information about the inputs

Beyond what is revealed by the function
Poker With No Dealer?

- Need to ensure
  - Cards are shuffled and dealt correctly
  - Complete secrecy
  - No "cheating" by players, even if they collude
- No universally trusted dealer
Without any trusted party, securely do
- Distributed Data mining
- E-commerce
- Network Games
- E-voting
- Secure function evaluation
- ....

Any Task!
Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel

- Secure MPC: to emulate a source of trusted computation
  - Trusted means it will not “leak” a party’s information to others
  - And it will not cheat in the computation
SIM-Secure MPC

Secure (and correct) if:

\[ \forall \exists \text{s.t.} \forall \text{output of is distributed identically in REAL and IDEAL} \]
Trust Issues Considered

- Protocol may leak a party’s secrets
  - Clearly an issue -- even if we trust everyone not to cheat in our protocol (i.e., honest-but-curious)
  - Also, a liability for a party if extra information reaches it
    - Say in medical data mining
  - Protocol may give adversary illegitimate influence on the outcome
    - Say in poker, if adversary can influence hands dealt
- SIM security covers these concerns
  - Because IDEAL trusted entity would allow neither
Adversary

- REAL-adversary can corrupt any set of players
- In security requirement IDEAL-world adversary should corrupt the same set of players
  - i.e., environment gets to know the set of corrupt players
- More sophisticated notion: adaptive adversary which corrupts players dynamically during/after the execution
- We’ll stick to static adversaries
- Passive adversary: gets only read access to the internal state of the corrupted players
Passive Adversary

- Gets **only read access** to the internal state of the corrupted players (and can use that information in talking to environment)
- Also called “Honest-But-Curious” adversary
- Will require that simulator also corrupts passively
- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]
- Oddly, sometimes security against a passive adversary is more demanding than against an active adversary
  - Active adversary: too pessimistic about what guarantee is available even in the IDEAL world
  - e.g. 2-party SFE for OR, with output going to only one party (trivial against active adversary; impossible without computational assumptions against passive adversary)
Example Functionalities

- Can consider “arbitrary” functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated

- Some simple (but important) examples:
  - **Secure Function Evaluation**
    - e.g. *Oblivious Transfer* (coming up)
    - Can be randomized: e.g. *Coin-tossing*
  - “Reactive” functionalities (maintains state over multiple rounds)
    - e.g. *Commitment* (coming up)
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

IDEAL World
30 Day Free Trial

Commit now, reveal later

Intuitive properties: hiding and binding

We Predict STOCKS!!

“COMMIT”

“REVEAL”

Really?

Next Day

Commit: m

reveal: m
Oblivious Transfer

- Pick one out of two, without revealing which

- Intuitive property: transfer partial information "obliviously"

IDEAL World

We Predict STOCKS!!

A: up, B: down

I need just one
But can't tell you which

Sure

All 2 of them!

A → OT

B → OT

\(x_0 \ x_1 \ b \ x_b\)
Can we REAL-ize them?

- Are there protocols which securely realize these functionalities?
  - Securely Realize: A protocol for the REAL world, so that SIM security definition satisfied

- Turns out SIM definition “too strong”

- Unless modified carefully...
Alternate Security Definitions

- **Standalone security**: environment is not “live”: interacts with the adversary before and after (but not during) the protocol.

- **Honest-majority security**: adversary can corrupt only a strict minority of parties. (Not useful when only two parties involved)

- **Passive (a.k.a honest-but-curious) adversary**: where corrupt parties stick to the protocol (but we don’t want to trust them with information)

- **Functionality-specific IND definitions**: usually leave out several attacks (e.g. malleability related attacks)

- **Protocols on top of a real trusted entity for a basic functionality**

- **Modified SIM definitions** (super-PPT adversary for ideal world)
2-Party Secure Function Evaluation

- Functionality takes (X;Y) and outputs f(X;Y) to Alice, g(X;Y) to Bob
- OT is an instance of 2-party SFE
  - \[ f(x_0, x_1; b) = \text{none}; g(x_0, x_1; b) = x_b \]
- Symmetric SFE: both parties get the same output
  - e.g. \[ f(x_0, x_1; b, z) = g(x_0, x_1; b, z) = x_b \oplus z \] [OT from this! How?]
- More generally, any SFE from an appropriate symmetric SFE
  - i.e., there is a protocol securely realizing SFE functionality G, which accesses a trusted party providing some symmetric SFE functionality F
- Exercise
2-Party Secure Function Evaluation

Randomized Functions: \( f(X;Y;r) \)

- \( r \) is chosen randomly by the trusted party.
- Neither party should know \( r \) (beyond what is revealed by output).

Consider evaluating \( f'(X,a;Y,b) := f(X;Y;a \oplus b) \)

- Note \( f' \) is deterministic.

- If either \( a \) or \( b \) is random, \( a \oplus b \) is random and hidden from each party.

- Gives a protocol using access to \( f' \), to securely realize \( f \).
An OT Protocol (passive receiver corruption)

Using a TOWP

- Depends on receiver to pick \( x_0, x_1 \) as prescribed
- Simulation for passive corrupt receiver: simulate \( z_0, z_1 \) knowing only \( x_b \) (use random \( z_{1-b} \))
- Simulation for corrupt sender: Extract \( x_0, x_1 \) from interaction (pick \( s_{1-b} \) also)
Today

Secure MPC: formalized using IDEAL world with trusted computational entity

Examples: poker, auction, privacy-preserving data-mining

Basic Examples: SFE, Oblivious Transfer, Commitment

Weaker security requirements: security against passive (honest-but-curious) adversary, standalone security

Example of a protocol: OT secure against passive adversary