Secure Multi-Party Computation

Lecture 15
Must We Trust eBay?
Must We Trust eBay?

Can we have an auction without an auctioneer?!
Can we have an auction without an auctioneer?!
Can we have an auction without an auctioneer?!
Must We Trust eBay?

Can we have an auction without an auctioneer?!

Declared winning bid should be correct
Must We Trust eBay?

Can we have an auction without an auctioneer?!

- Declared winning bid should be correct
- Only the winner and winning bid should be revealed
Using data without sharing?
Using data without sharing?

Hospitals which can’t share their patient records with anyone
Using data without sharing?

- Hospitals which can’t share their patient records with anyone
- But want to data-mine on combined data
Secure Function Evaluation

$f(X_1, X_2, X_3, X_4)$
Secure Function Evaluation

A general problem

\[ f(X_1, X_2, X_3, X_4) \]
Secure Function Evaluation

A general problem

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

\[ f(X_1, X_2, X_3, X_4) \]
Secure Function Evaluation

A general problem

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

Beyond what is revealed by the function

\[ f(X_1, X_2, X_3, X_4) \]
Poker With No Dealer?
Poker With No Dealer?
Poker With No Dealer?

Need to ensure
Need to ensure

Cards are shuffled and dealt correctly
Poker With No Dealer?

Need to ensure
- Cards are shuffled and dealt correctly
- Complete secrecy
Poker With No Dealer?

Need to ensure

- Cards are shuffled and dealt correctly
- Complete secrecy
- No “cheating” by players, even if they collude
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
The Ambitious Goal
Without any trusted party, securely do

- Distributed Data mining
- E-commerce
- Network Games
- E-voting
- Secure function evaluation

....
The Ambitious Goal

Without any trusted party, securely do

- Distributed Data mining
- E-commerce
- Network Games
- E-voting
- Secure function evaluation

....
Emulating Trusted Computation
Encryption/Authentication allowed us to emulate a trusted channel
Emulating Trusted Computation

- Encryption/Authentication allowed us to emulate a trusted channel
- Secure MPC: to emulate a source of trusted computation
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
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
A Simple example

An auction, with Alice and Bob bidding
A Simple example

An auction, with Alice and Bob bidding

Rules:

- A bid is an integer in the range \([0,100]\)
- Alice can bid only even integers and Bob odd integers
- Person with the higher bid wins
A Simple example

An auction, with Alice and Bob bidding

Rules:

- A bid is an integer in the range [0,100]
- Alice can bid only even integers and Bob odd integers
- Person with the higher bid wins

Goal: find out the winning bid (winner & amount) without revealing anything more about the losing bid (beyond what is revealed by the winning bid)
A Simple example

Secure protocol:

- Count down from 100
- At each even round Alice announces whether her bid equals the current count; at each odd round Bob does the same
- Stop if a party says yes
A Simple example

Secure protocol:

- Count down from 100
- At each even round Alice announces whether her bid equals the current count; at each odd round Bob does the same
- Stop if a party says yes

Dutch flower auction
A Simple example

Secure protocol:

- Count down from 100
- At each even round Alice announces whether her bid equals the current count; at each odd round Bob does the same
- Stop if a party says yes

Dutch flower auction

What kind of security does this protocol get? (Later: “stand-alone” security)
SIM-Secure MPC
SIM-Secure MPC

IDEAL

Env

REAL

Env
SIM-Secure MPC
SIM-Secure MPC

Secure (and correct) if:

∀ s.t. output of is distributed identically in REAL and IDEAL
SIM-Secure MPC

Secure (and correct) if:
\[ \forall \exists \text{s.t.} \forall \text{output of is distributed identically in REAL and IDEAL} \]
SIM-Secure MPC

Secure (and correct) if:

∀ output of is distributed identically in REAL and IDEAL

∀ s.t.

IDEAL

Env

REAL

Env
Trust Issues Considered
Trust Issues Considered

Protocol may leak a party's secrets
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)
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
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
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
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
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
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
Adversary

- REAL-adversary can corrupt any set of players
Adversary

- REAL-adversary can corrupt any set of players
- In security requirement IDEAL-world adversary should corrupt the same set of players
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
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 corruptions players dynamically during/after the execution
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
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 vs. Active adversary: Passive adversary gets only read access to the internal state of the corrupted players. Active adversary overwrites their state and program.
Passive Adversary

Gets **only read access** to the internal state of the corrupted players (and can use that information in talking to environment)
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
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
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
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]
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
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.
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)
More Example Functionalities
More Example
Functionalities

Can consider “arbitrary” functionalities
More Example Functionalities

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

- Can consider “arbitrary” functionalities
  - i.e., arbitrary (PPT) program of the trusted party to be emulated
- Some simple (but important) examples:
More 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
More 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. Finding max, Oblivious Transfer (coming up)
More 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. Finding max, Oblivious Transfer (coming up)

Can be randomized: e.g. Coin-tossing
More 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. Finding max, Oblivious Transfer (coming up)
  - Can be randomized: e.g. Coin-tossing
  - “Reactive” functionalities (maintains state over multiple rounds)
More 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. Finding max, Oblivious Transfer (coming up)

Can be randomized: e.g. Coin-tossing

“Reactive” functionalities (maintains state over multiple rounds)

- e.g. Commitment (coming up)
Commitment
Commitment

Commit now, reveal later
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

We Predict STOCKS!!

Really?

IDEAL World
30 Day Free Trial
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

We Predict STOCKS!!

Really?

IDEAL World
30 Day Free Trial
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

IDEAL World
30 Day Free Trial
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

IDEAL World
30 Day Free Trial
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

IDEAL World
30 Day Free Trial

We Predict STOCKS!!

“REVEAL”

Really?

Next Day

Commit: m commit

up

up
Commitment

Commit now, reveal later

Intuitive properties: hiding and binding

We Predict STOCKS!!

“REVEAL”

IDEAL World
30 Day Free Trial
Oblivious Transfer

IDEAL World

We Predict STOCKS!!
Oblivious Transfer

Pick one out of two, without revealing which
Oblivious Transfer

- Pick one out of two, without revealing which

- Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information "obliviously"

IDEAL World
Oblivious Transfer

- Pick one out of two, without revealing which

- Intuitive property: transfer partial information "obliviously"

IDEAL World

We Predict STOCKS!!

All 2 of them! Sure

I need just one
Oblivious Transfer

- Pick one out of two, without revealing which

- Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information "obliviously"
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”
Can we REAL-ize them?
Can we REAL-ize them?

Are there protocols which securely realize these functionalities?
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
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”
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
Alternate Security Definitions

**Standalone security**: environment is not “live”: interacts with the adversary before and after (but not during) the protocol.
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.)
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)
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. input dependence, malleability, …)
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. input dependence, malleability, …)

- **Full-fledged SIM security**, but protocols allowed to use a real trusted entity for a basic functionality
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. input dependence, malleability, …)

- **Full-fledged SIM security**, but protocols allowed to use a real trusted entity for a basic functionality

- **Modified SIM definitions** (super-PPT adversary for ideal world)