Advanced Tools from Modern Cryptography

Lecture 4

Secure Multi-Party Computation:
Passive Corruption,
Linear Functions
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?

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

\[ f(X_1, X_2, X_3, X_4) \]
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 that uses a trusted party!

Secure Multi-Party Computation (MPC)
Mental Poker

Adi Shamir, Ronald L. Rivest
and Leonard M. Adleman

Massachusetts Institute of Technology

ABSTRACT

Can two potentially dishonest players play a fair game of poker without using any cards—for example, over the phone? This paper provides the following answers:

1. No. (Rigorous mathematical proof supplied.)
2. Yes. (Correct and complete protocol given.)
Emulating Trusted Computation

- Encryption/Authentication allow 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 tool for mutually distrusting parties to collaborate
Is it for Real?

Getting there!

Many implementations/platforms

Fairplay, VIFF, Sharemind, SCAPI, Obliv-C, JustGarble, SPDZ/MASCOT, ObliVM, ...

See multipartycomputation.com
Is it for Real?

And many practical systems using some form of MPC:
- Danish company Partisia with real-life deployments (since 2008)
  - sugar beet auction, electricity auction, spectrum auction, key management
- A prototype for credit rating, supported by Danish banks
- A proposal to the Estonian Tax & Customs Board
- A proposal for Satellite Collision Analysis
- Legislation in the US to use MPC for applications like a “higher education data system”
- MPC Alliance
- ...
MPC

Several dimensions

- Passive (Semi-Honest) vs. Active corruption
  - Passive: corrupt parties still follow the protocol
- Honest-Majority vs. Unrestricted corruption
- Information-theoretic vs. Computational security
- ...
Security Definition

- Simplest case: Passive corruption, Information-theoretic security
- In general, need honest-majority (or similar restriction)
- In passive corruption, the adversary can see the internals of all the corrupt parties, but cannot control their actions
  - Main concern will be secrecy (correctness is automatic, provided the protocol is correct in the absence of corruption)
- Will ask for Perfect Secrecy
  - Similar to secret-sharing
Multiple parties in a protocol could be corrupt

Collusion

Modelled using a single adversary who corrupts the parties

Its view contains all the corrupt parties’ views

Security guarantee given against an “adversary structure”

Sets of parties that could be corrupt together
For secret sharing we needed to formalise “x is secret”

Now want to say: x is **secret except for** f(x) which is revealed

∀ x, x’ s.t. f(x)=f(x’), { view | input=x} = { view | input=x’ }

Here f(x) consists of the coordinates of input as well as the coordinates of outputs that correspond to corrupted parties

i.e., what the collusion is **allowed** to learn about x

Later: More complicated when considering active corruption and/or computational security
MPC for Linear Functions

Client-server setting

\[ f_1(x_1, \ldots, x_5) \]

\[ f_2(x_1, \ldots, x_5) \]
MPC for Linear Functions: Using Linear Secret-Sharing

Using Linear Secret-Sharing

\[ f_1(x_1, \ldots, x_5) \]

\[ f_2(x_1, \ldots, x_5) \]
MPC for Linear Functions:
Using Linear Secret-Sharing

Each row given to a server

Each column with an input client

Each column sent to an output client
MPC for Linear Functions: Using Linear Secret-Sharing

Each column with an input client

Each row given to a server

Each column sent to an output client

View of the adversary (corrupt parties)
Security

- Adversary allowed to corrupt any set of input and output clients and any subset T of servers s.t. T is not a privileged set (i.e., not in the access structure) for the secret-sharing scheme.

- View of adversary should reveal nothing beyond the inputs and outputs of the corrupted clients.

Claim: Consider any input y of corrupt clients. If x, x' of uncorrupted clients such that for each corrupt output client i, \( f_i(x, y) = f_i(x', y) \), then the view of the adversary in the two cases are identically distributed.

Because for any given view of the adversary, in each of the two cases (x and x'), the solution space of randomness is non-empty and then it has the same dimension.

Exercise
MPC for General Functions?

- So far: a 2-round protocol for any *linear* function
  - Could use additive secret-sharing
- How about other functions?
- Any function over a finite field can be computed using addition and multiplication
  - Interested in functions which are efficiently computable
- Arithmetic circuit: representation of the computation using addition and multiplication
- Goal: MPC Protocol for $f$, which is efficient if we are given an efficient arithmetic circuit for $f$
MPC from Shamir Secret-Sharing: Overview

A function $f$ given as a program with linear steps and multiplications: arithmetic circuit (over a finite field)

Locally multiplying degree $d$ shares of $M_1$ and $M_2$ gives a degree $2d$ share of $M_1 \cdot M_2$. Then securely switch back to a degree $d$ share (involves communicating degree $d$ shares of degree $2d$ shares)

Need $n > 2d$ parties. Security against $d$ colluding parties

Involves rerandomisation for "refreshing" shares