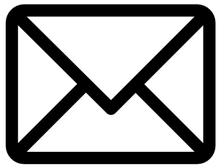
CS 350S: Privacy-Preserving Systems

Multi-party computation I

Outline

1. **MPC introduction**
2. MPC for arithmetic circuits
3. Logistics

What is multi-party computation (MPC)?



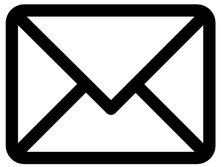
Trusted third-party

Spam Classifier

Trusted third-party is central point of attack

What is multi-party computation (MPC)?

(Informal) Any computation that can be performed with a trusted third party can be securely computed *without* one!



Multi-party computation

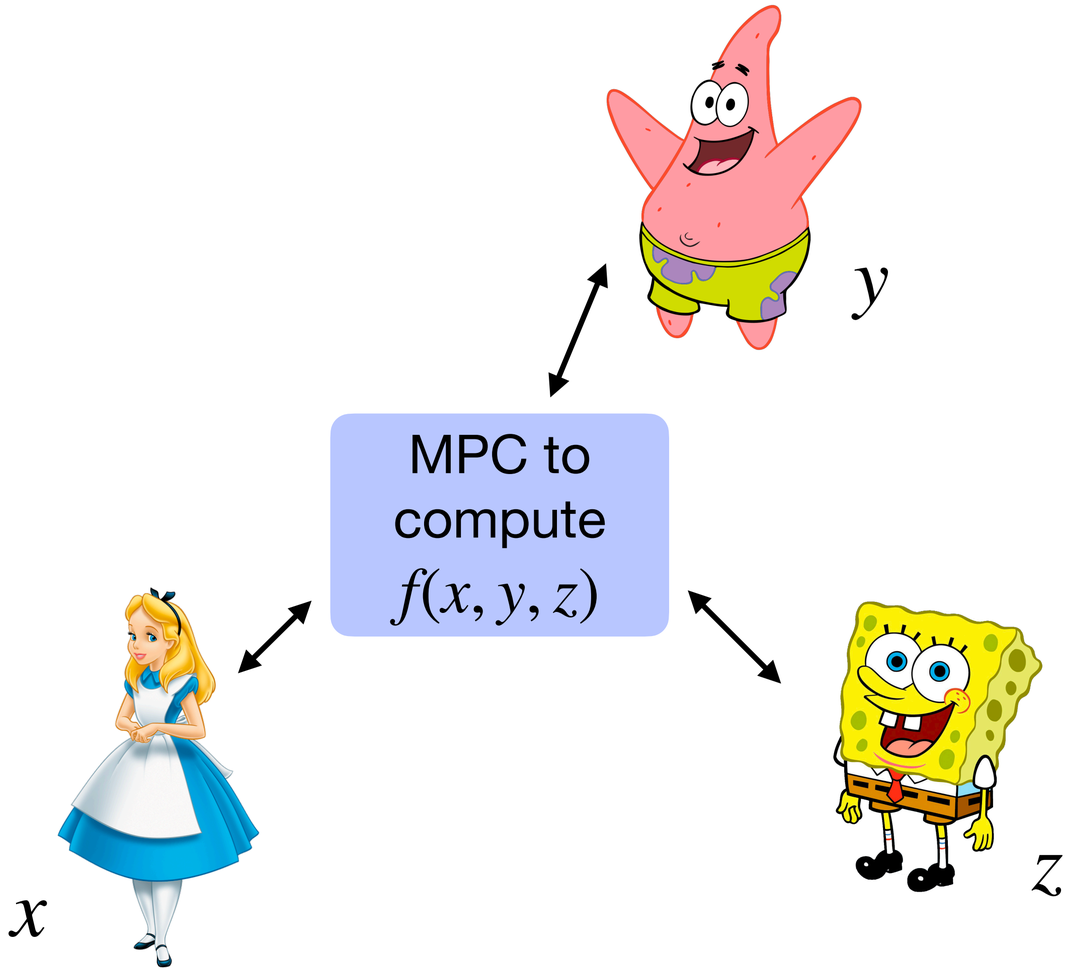
Spam Classifier

No central point of attack!

Defining MPC

Parties with inputs x, y, z that want to jointly compute the function $f(x, y, z)$

- Assume encrypted, authenticated channels between parties
- Generalize to any number of parties



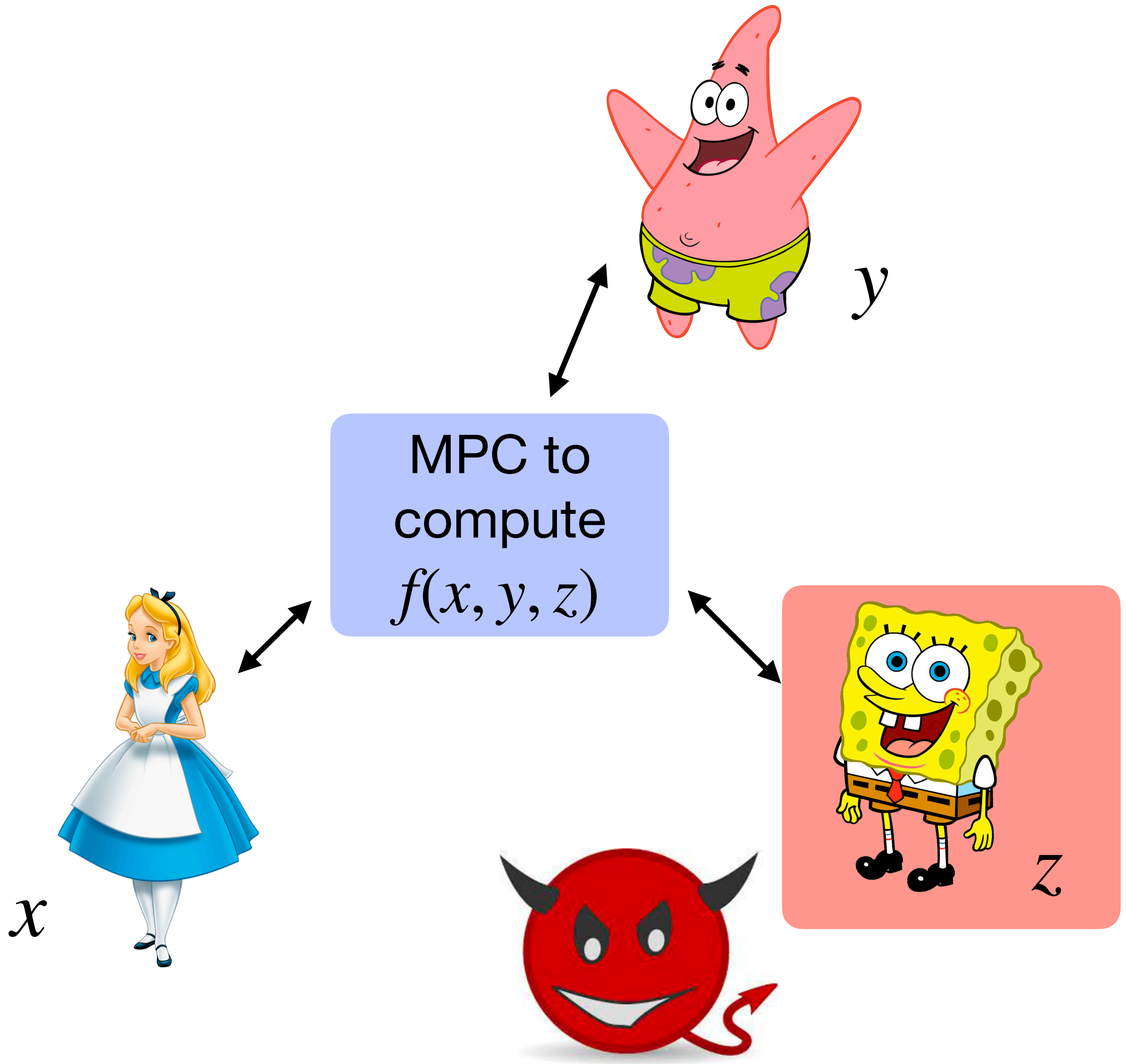
Defining MPC

Parties with inputs x, y, z that want to jointly compute the function $f(x, y, z)$

- Assume encrypted, authenticated channels between parties
- Generalize to any number of parties

Defends against attacker that compromises a subset of the parties

- Exact security properties depends on MPC protocol



Applications?

- Analytics across users' medical records
- Private auctions
- Cryptocurrency wallets
- Genome-wide association analysis
- Voting
- Detecting fraudulent transactions (e.g., money laundering)
- Training a machine learning model across many users' data
- Etc.

Two adversary models in MPC

Semihonest: The corrupted parties *follow the protocol specification*. After the protocol completes, they look at the transcript and try to extract info about the honest parties' input.

Malicious: The corrupted parties may *arbitrarily deviate from the protocol specification* to learn extra info about the honest parties' inputs or trick them into producing the wrong output.

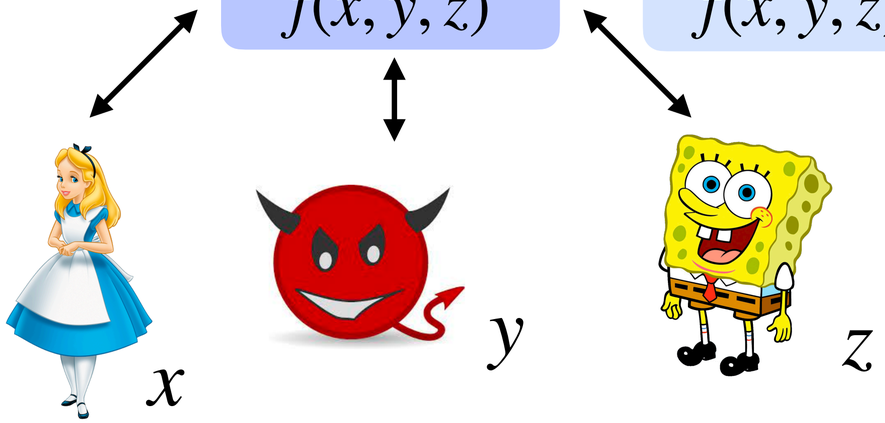
Defining semihonest security

Informally: Anything the adversary learns in an execution of the MPC, it could also have learned if all the parties were interacting with a trusted third party

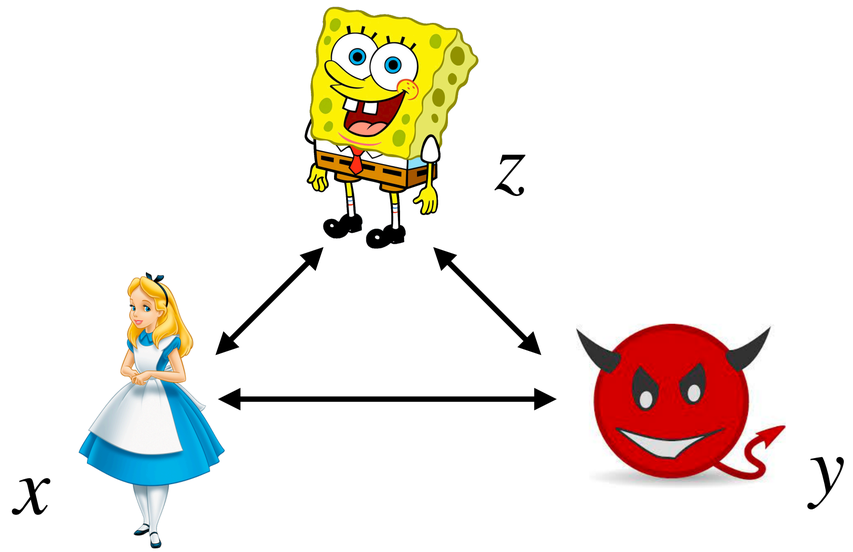
Ideal world

In ideal world, attacker learns
- Input of corrupted party y
- Output of computation $f(x, y, z)$

Trusted third-party for $f(x, y, z)$



Real world



Defining semihonest security

Ideal world

x

z

Trusted third-party for $f(x, y, z)$

Simulator

y

$\approx$

The adversary cannot tell whether it is in the real or ideal world

Real world

z

x

y

Defining semihonest security

Ideal world

x

z

Ideal functionality for $f(x, y, z)$

Simulator

y

Simulator only receives $f(x, y, z)$

$\approx$

The adversary cannot tell whether it is in the real or ideal world

Real world

z

x

y

Outline

1. MPC introduction
2. **MPC for arithmetic circuits**
3. Logistics

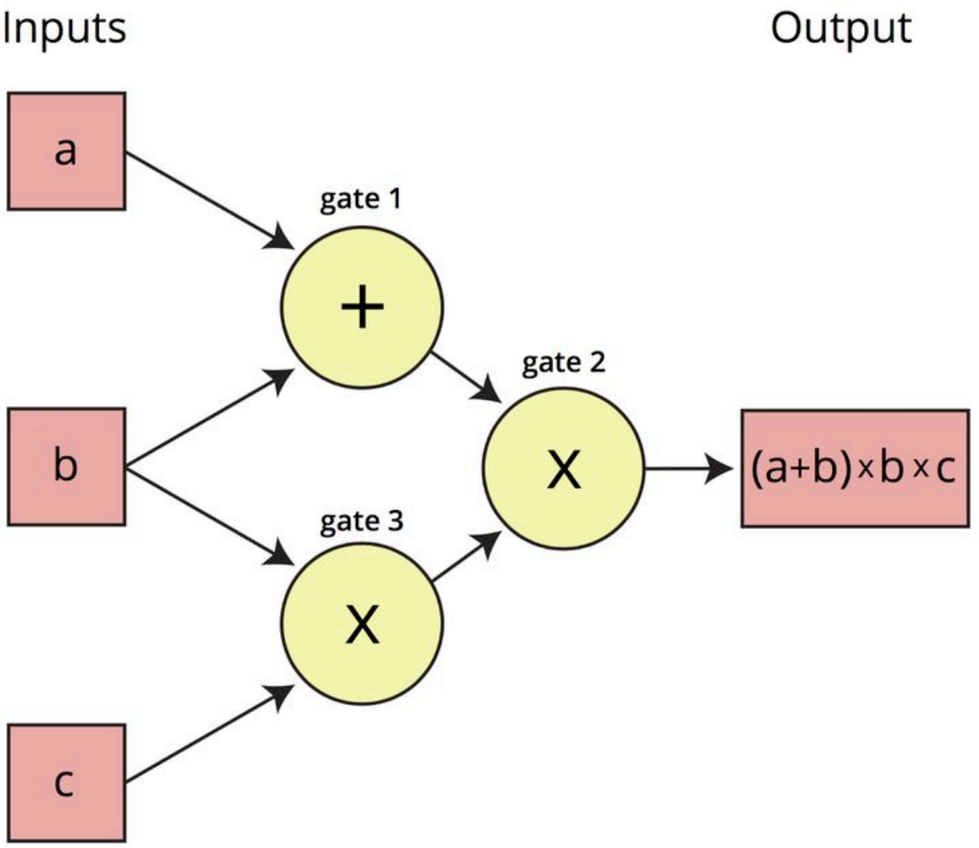
Background: Arithmetic circuits

Can express any computation via an arithmetic circuit.

Two types of gates

- Multiplication and addition gates

Idea: construct multi-party computation protocol for generic arithmetic circuit



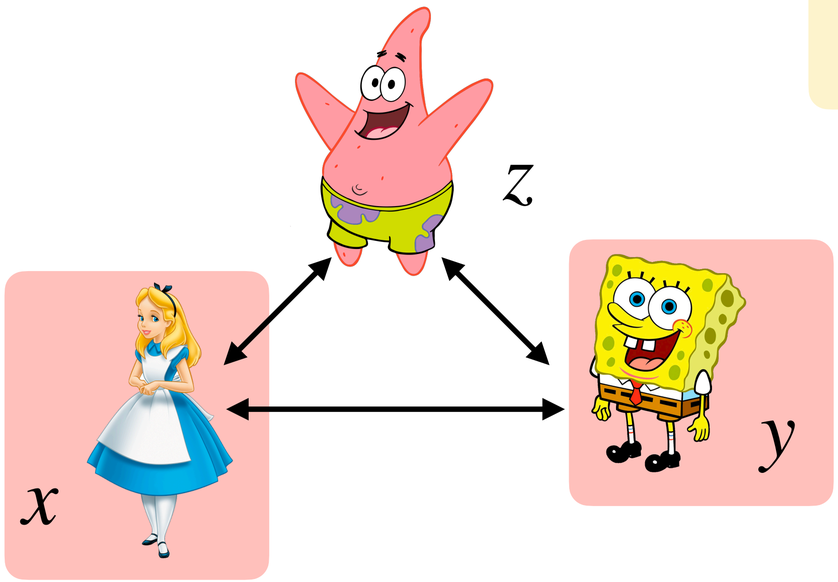
Security guarantees

With n parties, defends against $n - 1$ corruptions.

Semihonest security right now (compromised parties follow protocol)

- Can extend to malicious security

Will show for 2 parties but techniques generalize to n parties



High-level overview

If we have a way to compute addition and multiplication gates:

1. Each party shares its input with every other party.
2. For each addition gate with inputs $[x], [y]$, parties compute shares of $[x + y]$ 🤔
3. For each multiplication gate with inputs $[x], [y]$, parties compute shares of $[x \cdot y]$ 🤔
4. Parties publish shares of output.

Addition gates: add shares locally (no communication required)

Multiply shares

Parties have shares of x, y

How to compute shares of $z = x \cdot y$?

x_1, y_1

$z_1 \leftarrow x_1 \cdot y_1$ ✗

x_2, y_2

$z_2 \leftarrow x_2 \cdot y_2$ ✗

$$(x_1 + x_2)(y_1 + y_2) = x_1y_1 + \underline{x_2y_1 + x_1y_2} + x_2y_2 \neq x_1y_1 + x_2y_2$$

Multiply shares using Beaver triples

Want to compute shares of
$z = x \cdot y$

a_1, b_1, c_1

x_1, y_1

$d_1 \leftarrow x_1 - a_1$

$e_1 \leftarrow y_1 - b_1$

d_1, e_1

d_2, e_2

a_2, b_2, c_2

x_2, y_2

$d_2 \leftarrow x_2 - a_2$

$e_2 \leftarrow y_2 - b_2$

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_1 \leftarrow de + b_1 d + a_1 e + c_1$

$z_2 \leftarrow b_2 d + a_2 e + c_2$

Idea: Give Alice and Bob shares of random triple (a, b, c) where $a \cdot b = c$

Multiply shares using Beaver triples

Want to compute shares of $z = x \cdot y$

a_1, b_1, c_1

x_1, y_1

$d_1 \leftarrow x_1 - a_1$

$e_1 \leftarrow y_1 - b_1$

d_1, e_1 →

← d_2, e_2

Why is it safe to send shares of d, e?

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_1 \leftarrow de + b_1 d + a_1 e + c_1$

a_2, b_2, c_2

x_2, y_2

$d_2 \leftarrow x_2 - a_2$

$e_2 \leftarrow y_2 - b_2$

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_2 \leftarrow b_2 d + a_2 e + c_2$

Idea: Give Alice and Bob shares of random triple (a, b, c) where $a \cdot b = c$

Multiply shares using Beaver triples

Want to compute shares of
$z = x \cdot y$

a_1, b_1, c_1

x_1, y_1

$d_1 \leftarrow x_1 - a_1$

$e_1 \leftarrow y_1 - b_1$

d_1, e_1

d_2, e_2

a_2, b_2, c_2

x_2, y_2

$d_2 \leftarrow x_2 - a_2$

$e_2 \leftarrow y_2 - b_2$

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_1 \leftarrow de + b_1 d + a_1 e + c_1$

$z_2 \leftarrow b_2 d + a_2 e + c_2$

Why is it safe to send shares of d, e?

d, e don't reveal any information about x, y: a, b act as one-time pads

Idea: Give Alice and Bob shares of random triple (a, b, c) where $a \cdot b = c$

Multiply shares using Beaver triples

Want to compute shares of
$z = x \cdot y$

a_1, b_1, c_1

x_1, y_1

$d_1 \leftarrow x_1 - a_1$

$e_1 \leftarrow y_1 - b_1$

d_1, e_1

d_2, e_2

Is it safe to use the same
(a, b, c) for multiple gates?

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_1 \leftarrow de + b_1 d + a_1 e + c_1$

a_2, b_2, c_2

x_2, y_2

$d_2 \leftarrow x_2 - a_2$

$e_2 \leftarrow y_2 - b_2$

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_2 \leftarrow b_2 d + a_2 e + c_2$

Idea: Give Alice and Bob shares of random triple (a, b, c) where $a \cdot b = c$

Multiply shares using Beaver triples

Want to compute shares of
$z = x \cdot y$

a_1, b_1, c_1

x_1, y_1

$d_1 \leftarrow x_1 - a_1$

$e_1 \leftarrow y_1 - b_1$

d_1, e_1 →

← d_2, e_2

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_1 \leftarrow de + b_1 d + a_1 e + c_1$

a_2, b_2, c_2

x_2, y_2

$d_2 \leftarrow x_2 - a_2$

$e_2 \leftarrow y_2 - b_2$

$d \leftarrow d_1 + d_2$

$e \leftarrow e_1 + e_2$

$z_2 \leftarrow b_2 d + a_2 e + c_2$

Is it safe to use the same (a, b, c) for multiple gates?

No! Values d, e with same (a, b, c) but different (x, y) will leak information about different (x, y) (re-using one-time pad)

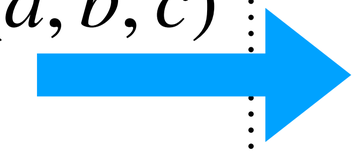
Online-offline model

Offline phase (input-independent)

How to generate shares of (a, b, c) where $a \cdot b = c$?

1. From another party (dealer)
2. Run a protocol to generate these

Shares of (a, b, c)



Online phase (input-dependent)

1. Each party shares its input with every other party.
2. For each addition gate with inputs $[x], [y]$, parties compute shares of $[x + y]$
3. For each multiplication gate with inputs $[x], [y]$, parties compute shares of $[x \cdot y]$
4. Parties publish shares of output.

Outline

1. MPC introduction
2. MPC for arithmetic circuits
3. **Logistics**

Logistics

Project report due 11/11

- Look out for signups for optional project feedback meetings

References

Boneh, Dan and Shoup, Victor. “A Graduate Course in Applied Cryptography.” Chapter 23. https://toc.cryptobook.us/book.pdf

Damgård, Ivan, Valerio Pastro, Nigel Smart, and Sarah Zakarias. "Multiparty computation from somewhat homomorphic encryption." In *Annual cryptology conference*, pp. 643-662. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

Lindell, Yehuda. "How to simulate it–a tutorial on the simulation proof technique." *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich* (2017): 277-346.

https://securecomputation.org/docs/pragmaticmpc.pdf

Stanford CS 355

MIT 6.5610