

Cryptography

Lecture 25

Arkady Yerukhimovich

December 2, 2024

- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review
- 3 Secure Multi-Party Computation (MPC)
- 4 A Simple MPC Protocol
- 5 MPC Based on Secret Sharing
- 6 Defining MPC Security

Lecture 24 Review

- Signatures from private-key primitives
- One-time signatures
- Going from one-time to standard signatures

Outline

- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review**
- 3 Secure Multi-Party Computation (MPC)
- 4 A Simple MPC Protocol
- 5 MPC Based on Secret Sharing
- 6 Defining MPC Security

El Gamal Encryption: $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ with $\mathcal{M} = G$

- $\text{Gen}(1^n)$: $(G, q, g) \leftarrow \text{Gen}(1^n)$, $x \leftarrow \mathbb{Z}_q$, $h = g^x$,
 $pk = (G, q, g, h)$ and $sk = x$
- $\text{Enc}_{pk}(m)$: $y \leftarrow \mathbb{Z}_q$, compute $c = (g^y, h^y \cdot m)$
- $\text{Dec}_{sk}(c)$: Compute $\hat{m} = c_2 / c_1^x$

Public-Key Protocols

El Gamal Encryption: $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ with $\mathcal{M} = G$

- $\text{Gen}(1^n)$: $(G, q, g) \leftarrow \text{Gen}(1^n)$, $x \leftarrow \mathbb{Z}_q$, $h = g^x$, $pk = (G, q, g, h)$ and $sk = x$
- $\text{Enc}_{pk}(m)$: $y \leftarrow \mathbb{Z}_q$, compute $c = (g^y, h^y \cdot m)$
- $\text{Dec}_{sk}(c)$: Compute $\hat{m} = c_2 / c_1^x$

Plain RSA Encryption

- $\text{Gen}(1^n)$: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = (N, d)$
- $\text{Enc}_{pk}(m)$: For $m \in \mathbb{Z}_N^*$, compute $c = [m^e \bmod N]$
- $\text{Dec}_{sk}(c)$: Compute $m = [c^d \bmod N]$

Public-Key Protocols

El Gamal Encryption: $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ with $\mathcal{M} = G$

- $\text{Gen}(1^n)$: $(G, q, g) \leftarrow \text{Gen}(1^n)$, $x \leftarrow \mathbb{Z}_q$, $h = g^x$, $pk = (G, q, g, h)$ and $sk = x$
- $\text{Enc}_{pk}(m)$: $y \leftarrow \mathbb{Z}_q$, compute $c = (g^y, h^y \cdot m)$
- $\text{Dec}_{sk}(c)$: Compute $\hat{m} = c_2 / c_1^x$

Plain RSA Encryption

- $\text{Gen}(1^n)$: $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = (N, e)$, $sk = (N, d)$
- $\text{Enc}_{pk}(m)$: For $m \in \mathbb{Z}_N^*$, compute $c = [m^e \bmod N]$
- $\text{Dec}_{sk}(c)$: Compute $m = [c^d \bmod N]$

Plain RSA Signature

- Gen : $(N, e, d) \leftarrow \text{GenRSA}(1^n)$, $pk = N, e$, $sk = d$
- $\text{Sign}_{sk}(m \in \mathbb{Z}_N^*)$: $\sigma = [m^d \bmod N]$
- $\text{Verify}_{pk}(m \in \mathbb{Z}_N^*, \sigma \in \mathbb{Z}_N^*)$: Output 1 if and only if $m = [\sigma^e \bmod N]$

Outline

- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review
- 3 Secure Multi-Party Computation (MPC)**
- 4 A Simple MPC Protocol
- 5 MPC Based on Secret Sharing
- 6 Defining MPC Security

The Need for Collaboration

Yao's Millionaire Problem



\$1.2M



\$1.6M

Who is the richest?

The Need for Collaboration

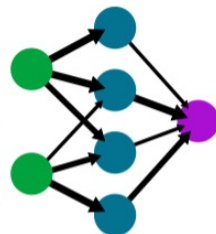


Name	Age	Smokes	Cancer
Jane Doe	25	Y	Y
John Doe	45	N	Y

Name	Age	Smokes	Cancer
Jane Smith	35	Y	N
John Smith	85	Y	Y

Name	Age	Smokes	Cancer
Joe Smith	50	N	N
Jim Doe	60	Y	Y

Neural Network



Will Get Cancer?

The Security Challenge

Question

How do parties who *do not trust each other* work together to compute joint functions on their private data?

- Without revealing their data or intermediate results to each other
- Without relying on a mutually-trusted third party

The Security Challenge

Question

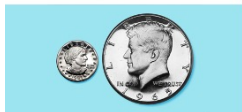
How do parties who *do not trust each other* work together to compute joint functions on their private data?

- Without revealing their data or intermediate results to each other
- Without relying on a mutually-trusted third party

Secure Multi-Party Computation (MPC or SMC) gives a solution:

- Originally developed in the 1980's
- Originally believed to be purely of theoretical interest
- But, since 2004 there have been many implementations showing real-world value
- Protocol and engineering improvements have yielded 6+ orders of magnitude speed up

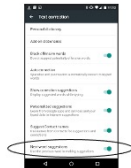
MPC in the Real World



Gender Pay Equity



Cryptographic Key Control



Federated Keyboard Prediction

Outline

- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review
- 3 Secure Multi-Party Computation (MPC)
- 4 A Simple MPC Protocol**
- 5 MPC Based on Secret Sharing
- 6 Defining MPC Security

Additive One-Time Pad

Given an Integer $x \in \mathbb{Z}_N$, consider the following scheme:

- Choose $r \leftarrow \mathbb{Z}_N$ at random
- Compute $y = x + r \bmod N$
- Can recover x from y if know r

Additive One-Time Pad

Given an Integer $x \in \mathbb{Z}_N$, consider the following scheme:

- Choose $r \leftarrow \mathbb{Z}_N$ at random
- Compute $y = x + r \bmod N$
- Can recover x from y if know r

Hiding

Given y , the value of x is perfectly hidden:

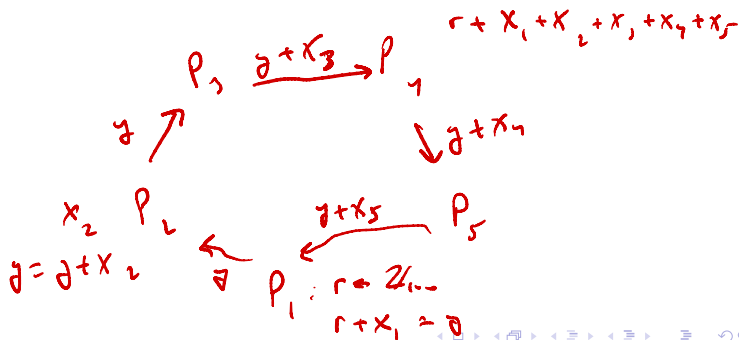
- Every value of $x \in \mathbb{Z}_N$ is equally likely

An MPC Protocol

Secure Summation

5 parties $\{P_1, \dots, P_5\}$ each with private input $x_i \in \mathbb{Z}_{100}$ want to compute

$$y = \sum_{i=1}^5 x_i$$



Outline

- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review
- 3 Secure Multi-Party Computation (MPC)
- 4 A Simple MPC Protocol
- 5 MPC Based on Secret Sharing**
- 6 Defining MPC Security

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

- P_1 chooses random x_1, x_2, \dots, x_n s.t. $\sum_{i=1}^n x_i = x \bmod N$ and sends x_i to P_i

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

- P_1 chooses random x_1, x_2, \dots, x_n s.t. $\sum_{i=1}^n x_i = x \bmod N$ and sends x_i to P_i
- P_j 's share of x is denoted as $[x]_j$

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

- P_1 chooses random x_1, x_2, \dots, x_n s.t. $\sum_{i=1}^n x_i = x \bmod N$ and sends x_i to P_i
- P_j 's share of x is denoted as $[x]_j$

Reconstruct:

- All parties send their shares to P_1 , who computes $x = \sum_{i=1}^n [x]_i$

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

- P_1 chooses random x_1, x_2, \dots, x_n s.t. $\sum_{i=1}^n x_i = x \bmod N$ and sends x_i to P_i
- P_j 's share of x is denoted as $[x]_j$

Reconstruct:

- All parties send their shares to P_1 , who computes $x = \sum_{i=1}^n [x]_i$
- Can send shares to all parties to open publicly

Additive Secret Sharing

Goal

We wish to distribute a secret value $x \in \mathbb{Z}_N$ among n parties s.t.:

- All n parties together can recover x
- Any set of $\leq n - 1$ parties has no information about x

Share: Suppose P_1 wants to secret-share $x \in \mathbb{Z}_N$

- P_1 chooses random x_1, x_2, \dots, x_n s.t. $\sum_{i=1}^n x_i = x \bmod N$ and sends x_i to P_i
- P_j 's share of x is denoted as $[x]_j$

Reconstruct:

- All parties send their shares to P_1 , who computes $x = \sum_{i=1}^n [x]_i$
- Can send shares to all parties to open publicly

Security

Easy to see that any set of $n - 1$ parties has no info about x

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$
- Correctness: $\sum_{i=1}^n ([x]_i + [y]_i) = \sum [x]_i + \sum [y]_i = x + y$

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$
- Correctness: $\sum_{i=1}^n ([x]_i + [y]_i) = \sum [x]_i + \sum [y]_i = x + y$
- Cost: 0 communication, 1 local addition

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$
- Correctness: $\sum_{i=1}^n ([x]_i + [y]_i) = \sum [x]_i + \sum [y]_i = x + y$
- Cost: 0 communication, 1 local addition

Multiplication

- Party P_i computes $[xy]_i = [x]_i \cdot [y]_i$

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$
- Correctness: $\sum_{i=1}^n ([x]_i + [y]_i) = \sum [x]_i + \sum [y]_i = x + y$
- Cost: 0 communication, 1 local addition

Multiplication

- Party P_i computes $[xy]_i = [x]_i \cdot [y]_i$
- Correctness: $xy = (\sum [x]_i)(\sum [y]_i) =$
 $([x]_1 + [x]_2 + \dots)([y]_1 + [y]_2 + \dots) =$
 $[x]_1[y]_1 + [x]_2[y]_2 + [x]_1[y]_2 + [x]_2[y]_1 + \dots \neq \sum x_i y_i$

Computing on Shared Values

Suppose parties hold two secret shared values:

- $[x] = ([x]_1, [x]_2, \dots, [x]_n)$
- $[y] = ([y]_1, [y]_2, \dots, [y]_n)$
- Each party P_i holds $[x]_i, [y]_i$

Addition

- Party P_i computes $[x + y]_i = [x]_i + [y]_i$
- Correctness: $\sum_{i=1}^n ([x]_i + [y]_i) = \sum [x]_i + \sum [y]_i = x + y$
- Cost: 0 communication, 1 local addition

Multiplication

- Party P_i computes $[xy]_i = [x]_i \cdot [y]_i$
- Correctness: $xy = (\sum [x]_i)(\sum [y]_i) = ([x]_1 + [x]_2 + \dots)([y]_1 + [y]_2 + \dots) = [x]_1[y]_1 + [x]_2[y]_2 + [x]_1[y]_2 + [x]_2[y]_1 + \dots \neq \sum x_i y_i$
- Problem: We need to compute the cross-terms (e.g., $[x]_1[y]_2$)

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$

$$[x] \rightarrow [2x]$$

$$x_1 + x_2 + x_3 = x \quad 2x_1 + 2x_2 + 2x_3 = 2x$$

$$[x] \rightarrow [x + c] \quad \text{for public } c$$
$$[x]_i \rightarrow [x]_i + c$$

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$
- Correctness:

$$(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c] =$$

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$
- Correctness:

$$\begin{aligned} & (x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c] = \\ & \cancel{x}[y] + \cancel{a}[y] + \cancel{y}[x] + \cancel{b}[x] - (\cancel{x}y + \cancel{a}y + \cancel{b}x + \cancel{a}b) + \cancel{[ab]} \end{aligned}$$

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$
- Correctness:

$$\begin{aligned}(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c] &= \\ x[y] + a[y] + y[x] + b[x] - (xy + ay + bx + ab) + [ab] &= [xy]\end{aligned}$$

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$
- Correctness:

$$\begin{aligned}(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c] &= \\ x[y] + a[y] + y[x] + b[x] - (xy + ay + bx + ab) + [ab] &= [xy]\end{aligned}$$

- Cost: 2 openings, and three local multiplications

Multiplication Triples

Suppose parties additionally hold shares of random *multiplication triples*:

- Random values $[a]$, $[b]$, $[c]$ such that $c = a \cdot b$

Multiplication: Given $[x]$, $[y]$ compute $[xy]$

- Parties compute $[x + a]$ and $[y + b]$ and open these values to everyone
- Each party computes $(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c]$
- Correctness:

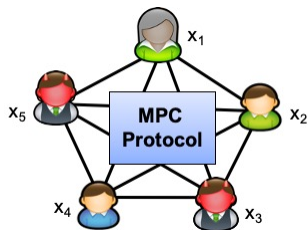
$$\begin{aligned}(x + a)[y] + (y + b)[x] - (x + a)(y + b) + [c] &= \\ x[y] + a[y] + y[x] + b[x] - (xy + ay + bx + ab) + [ab] &= [xy]\end{aligned}$$

- Cost: 2 openings, and three local multiplications
- Security: Since a and b are random $(x + a)$, $(y + b)$ reveal nothing about x , y

Outline

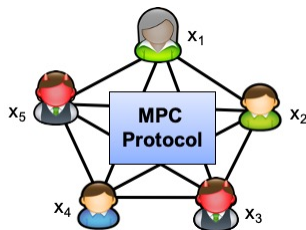
- 1 Lecture 24 Review
- 2 Public-Key Crypto Protocol Review
- 3 Secure Multi-Party Computation (MPC)
- 4 A Simple MPC Protocol
- 5 MPC Based on Secret Sharing
- 6 Defining MPC Security**

The MPC Protocol



$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

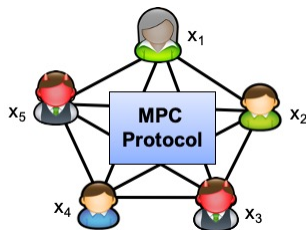
The MPC Protocol



$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

- n parties run an interactive protocol to compute function f

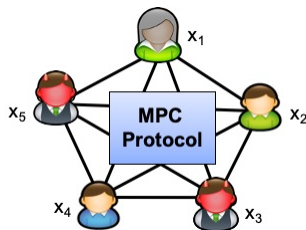
The MPC Protocol



$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

- n parties run an interactive protocol to compute function f
- Each party P_i gives input x_i and receives output y_i

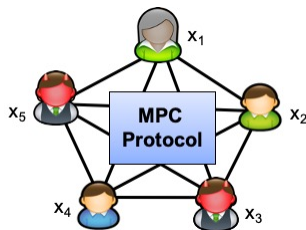
The MPC Protocol



$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

- n parties run an interactive protocol to compute function f
- Each party P_i gives input x_i and receives output y_i
- We assume private, authenticated channels between each pair of parties
- Often, also assume a secure broadcast channel

The MPC Protocol



$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

- n parties run an interactive protocol to compute function f
- Each party P_i gives input x_i and receives output y_i
- We assume private, authenticated channels between each pair of parties
- Often, also assume a secure broadcast channel
- Adversary (\mathcal{A}) controls some of the parties

Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else

Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else
- Correctness: The output should be a correct evaluation of f on provided inputs

Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else
- Correctness: The output should be a correct evaluation of f on provided inputs
- Fairness: If one player learns the output, all players learn the output

Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else
- Correctness: The output should be a correct evaluation of f on provided inputs
- Fairness: If one player learns the output, all players learn the output
- Independence of inputs: Each player must choose his input independently from other players' inputs

Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else
- Correctness: The output should be a correct evaluation of f on provided inputs
- Fairness: If one player learns the output, all players learn the output
- Independence of inputs: Each player must choose his input independently from other players' inputs
- Many more ...

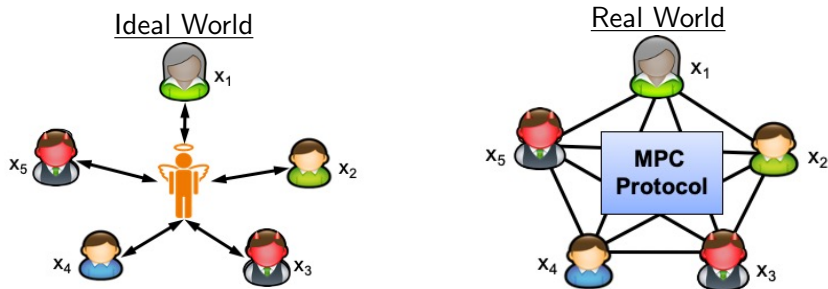
Desired Security Properties

- Privacy: Player P_i learns his input x_i , output y_i , and nothing else
- Correctness: The output should be a correct evaluation of f on provided inputs
- Fairness: If one player learns the output, all players learn the output
- Independence of inputs: Each player must choose his input independently from other players' inputs
- Many more ...

Defining Security

We could give a security definition for each of these, but instead take a different approach.

Real-Ideal Paradigm



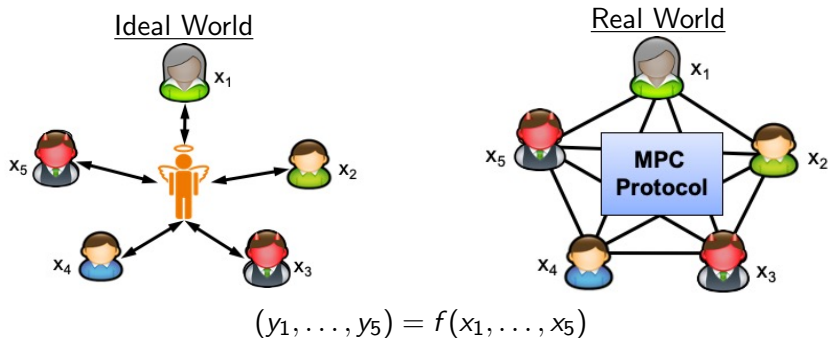
$$(y_1, \dots, y_5) = f(x_1, \dots, x_5)$$

Security Definition

MPC protocol emulates ideal-world execution

- Whatever security holds in ideal world holds in the real world
- Formally: Can build an ideal-world adversary (Simulator) that produces same view as real-world adversary (\mathcal{A})

Real-Ideal Paradigm



Security Definition

MPC protocol emulates ideal-world execution

- Whatever security holds in ideal world holds in the real world
- Formally: Can build an ideal-world adversary (Simulator) that produces same view as real-world adversary (\mathcal{A})

Simulating the Sharing-based MPC protocol

- Assume that \mathcal{A} corrupts $t < n$ parties

Simulating the Sharing-based MPC protocol

- Assume that \mathcal{A} corrupts $t < n$ parties
- Then, all \mathcal{A} even sees are random shares sent to parties he corrupts
- Since he never sees all n , these are completely random
- Simulator, can just send random values to \mathcal{A} to simulate his view (distributed the same as real protocol)

Limitations of Security Definition

Correctness of inputs

- MPC allows adversary to choose his own inputs
- No notion of “valid” input
- If you want to restrict inputs to have some properties need additional tools

Limitations of Security Definition

Correctness of inputs

- MPC allows adversary to choose his own inputs
- No notion of “valid” input
- If you want to restrict inputs to have some properties need additional tools

Output may leak private information

- MPC guarantees that adversaries learn nothing more than the output
- But, MPC does not consider what the output reveals
- In particular, output may reveal parties' inputs: e.g. $C(x,y) = x+y$
- Deciding what functions are safe to compute is orthogonal to MPC – this is studied by differential privacy

Limitations of Security Definition

Correctness of inputs

- MPC allows adversary to choose his own inputs
- No notion of “valid” input
- If you want to restrict inputs to have some properties need additional tools

Output may leak private information

- MPC guarantees that adversaries learn nothing more than the output
- But, MPC does not consider what the output reveals
- In particular, output may reveal parties' inputs: e.g. $C(x,y) = x+y$
- Deciding what functions are safe to compute is orthogonal to MPC – this is studied by differential privacy

Why does this not break security?

Limitations of Security Definition

Correctness of inputs

- MPC allows adversary to choose his own inputs
- No notion of “valid” input
- If you want to restrict inputs to have some properties need additional tools

Output may leak private information

- MPC guarantees that adversaries learn nothing more than the output
- But, MPC does not consider what the output reveals
- In particular, output may reveal parties' inputs: e.g. $C(x,y) = x+y$
- Deciding what functions are safe to compute is orthogonal to MPC – this is studied by differential privacy

Why does this not break security?

Both of these limitations also happen in the ideal world

Adversary Types

Adversary type:

- Semi-honest (honest but curious) – follows protocol, but tries to learn more
- Malicious – may deviate from protocol arbitrarily
- Covert – Malicious, but does not want to be caught

Adversary Types

Adversary type:

- Semi-honest (honest but curious) – follows protocol, but tries to learn more
- Malicious – may deviate from protocol arbitrarily
- Covert – Malicious, but does not want to be caught

Adversary Types

Adversary type:

- Semi-honest (honest but curious) – follows protocol, but tries to learn more
- Malicious – may deviate from protocol arbitrarily
- Covert – Malicious, but does not want to be caught

Adversary threshold:

- $t < n/2$ – honest majority
- $t < n$ – dishonest majority (particularly important for 2PC)
- $t < n/3$ – allows highly optimized protocols