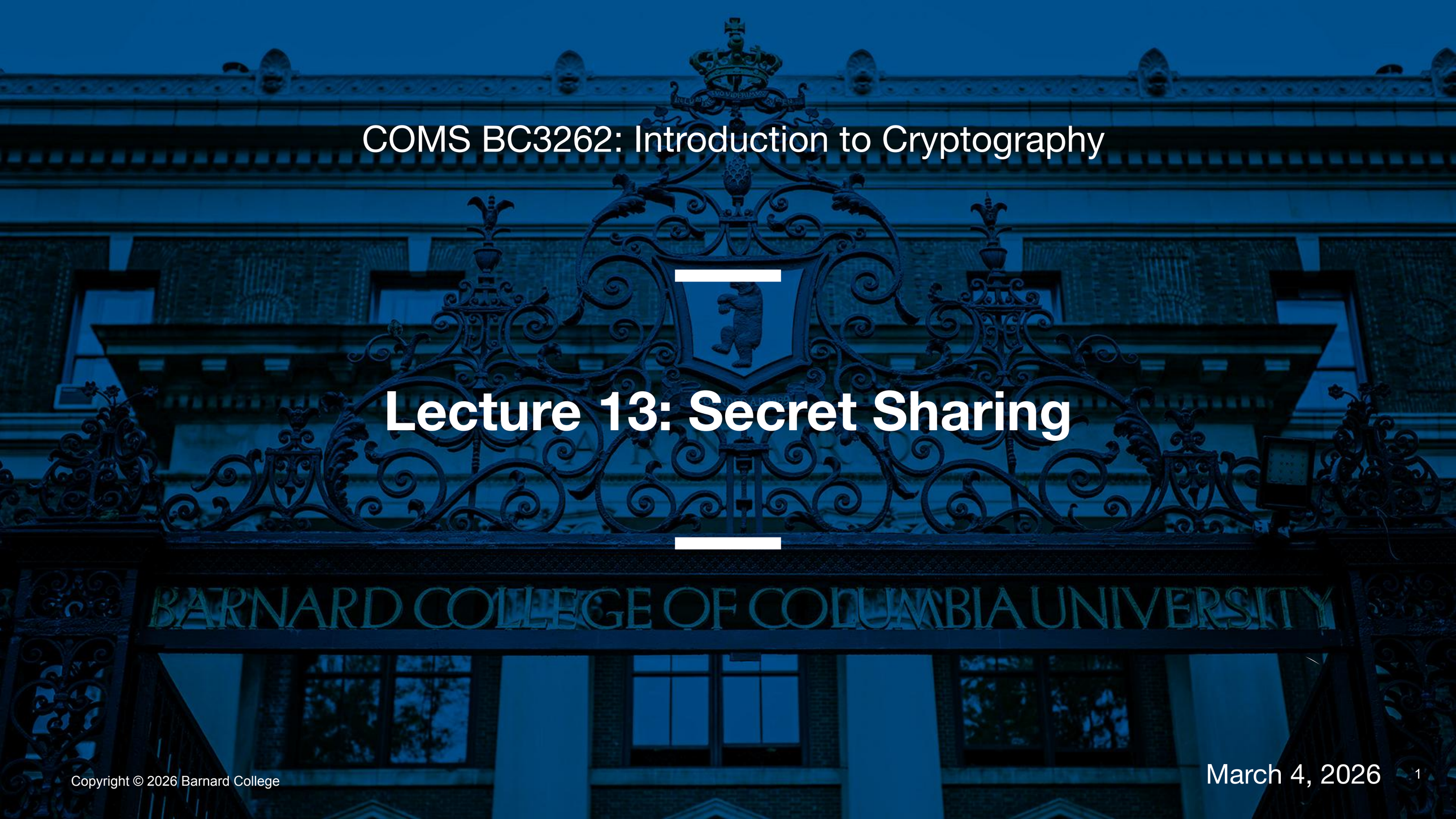


COMS BC3262: Introduction to Cryptography

Lecture 13: Secret Sharing

March 4, 2026

Office Hours

Office hours:

- **Eysa**: Mondays 3-5, Milstein 512
- **Mark**: Tuesdays 6:30-8:30, Milstein 503

Problem Sets

- PS 3 is due tomorrow
- We’ll go over answers on Monday
  - Unfortunately this means no late assignments will be accepted past Sunday 11:59pm

3262 students as soon as midterm season ends (finally getting to rest)

Midterm

- **In-class written midterm** next week on **Wednesday, March 11**
- Monday (March 9) will be a review session
  - Come with questions!
  - Lecture may end early if we run out of questions
- You may bring a single letter-sized reference sheet (double-sided)
  - You will be expected to submit your reference sheet along with your exam
- Exam is closed note, no technology, no collaboration
- *Exam will not cover any material introduced this week*

Today’s Lecture

- Secret Sharing
  - Shamir’s Secret Sharing Scheme
- Multiparty Computation
  - BGW

Secret Sharing

Protecting Secrets

- We’ve seen how to communicate privately using encryption and ensure authenticity with message authentication codes

Protecting Secrets

- We’ve seen how to communicate privately using encryption and ensure authenticity with message authentication codes

- Security crucially relies on the secret key remaining hidden

Protecting Secrets

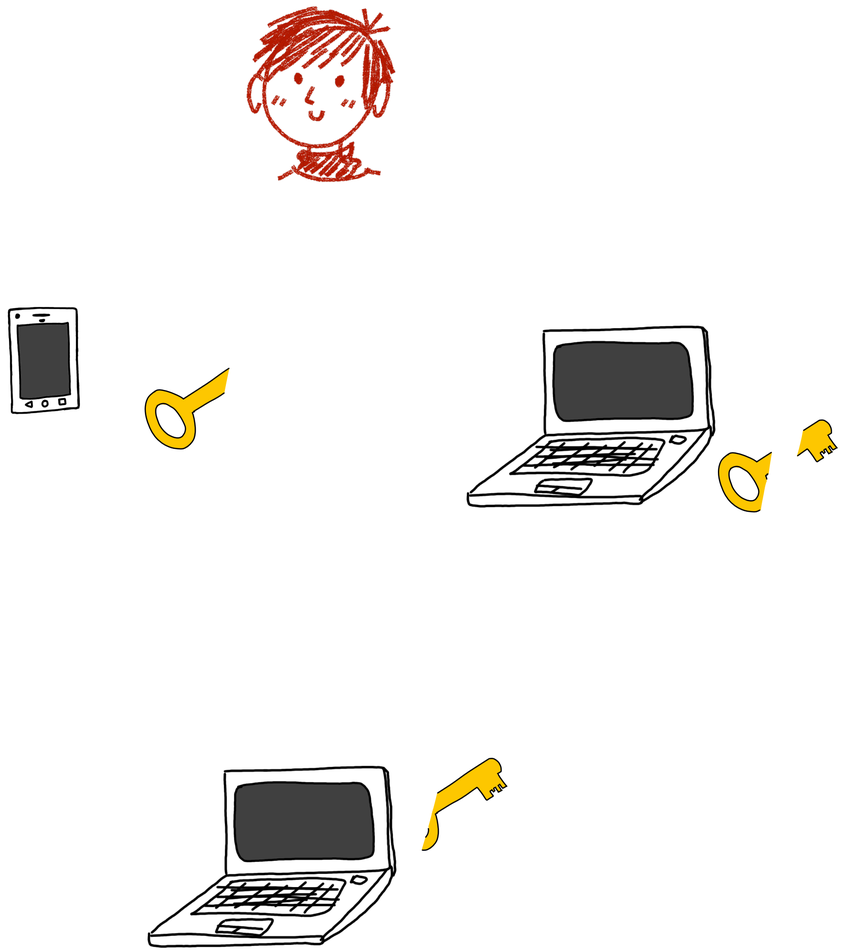
- We've seen how to communicate privately using encryption and ensure authenticity with message authentication codes
- Security crucially relies on the secret key remaining hidden
  - What if an adversary corrupts the device you store your secret key?

General Secret Sharing

The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device

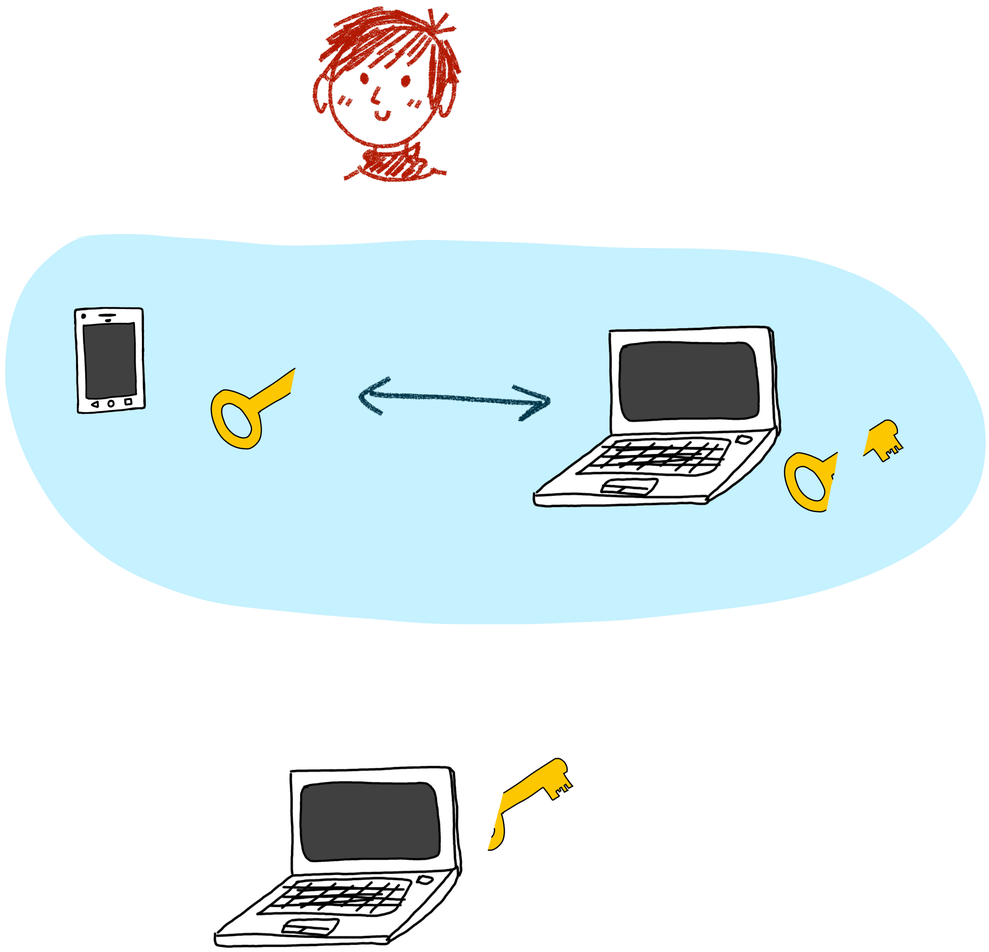
General Secret Sharing



The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device

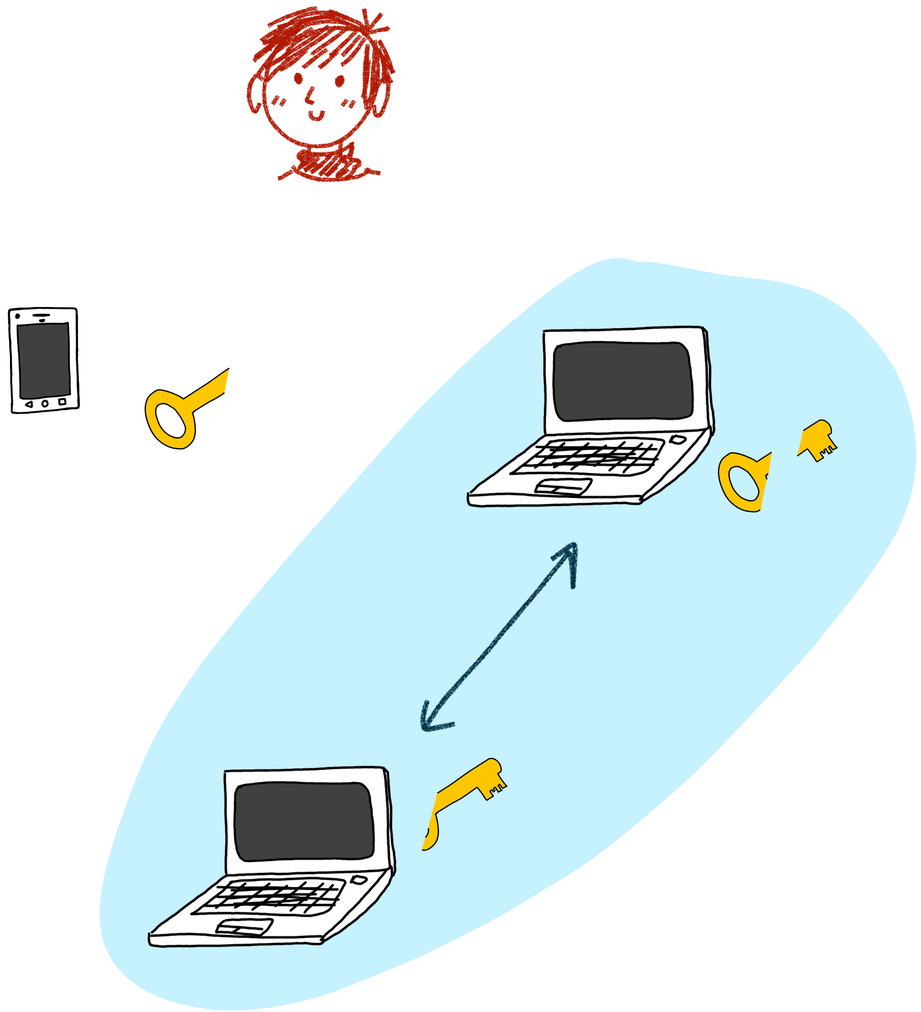
General Secret Sharing



The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device
- Authorized subsets of parties can reconstruct the secret using their shares

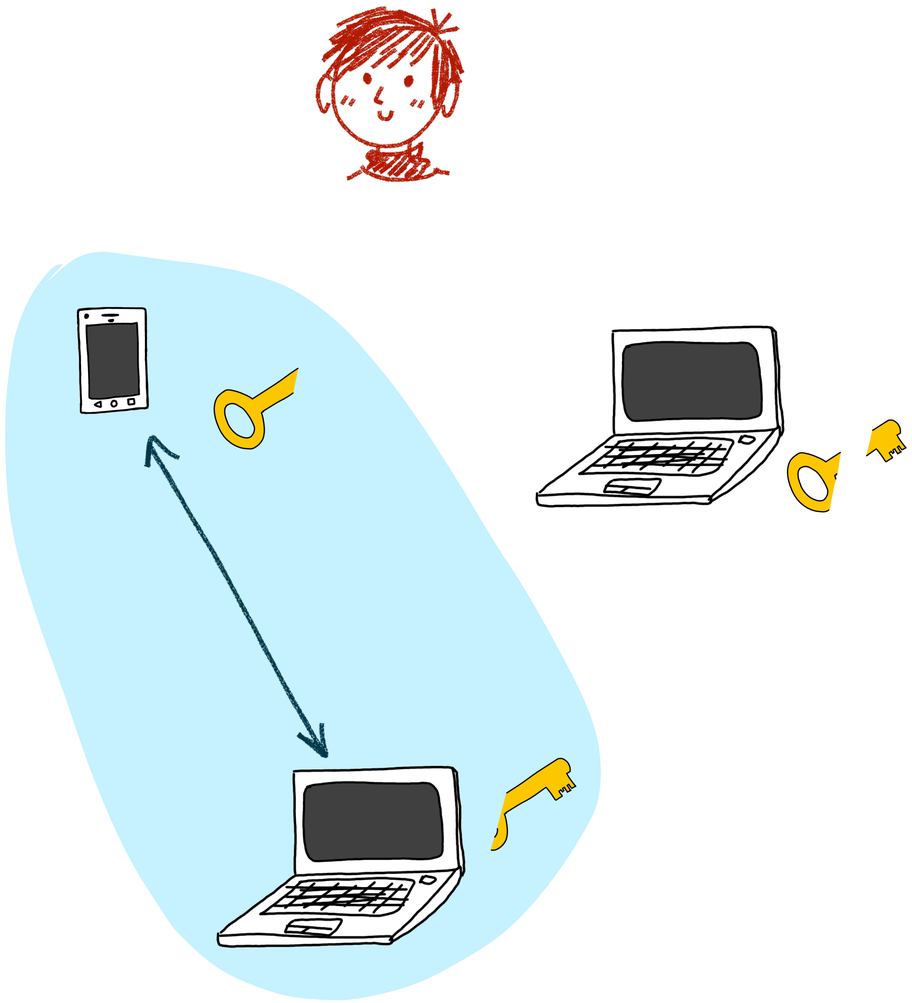
General Secret Sharing



The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device
- Authorized subsets of parties can reconstruct the secret using their shares

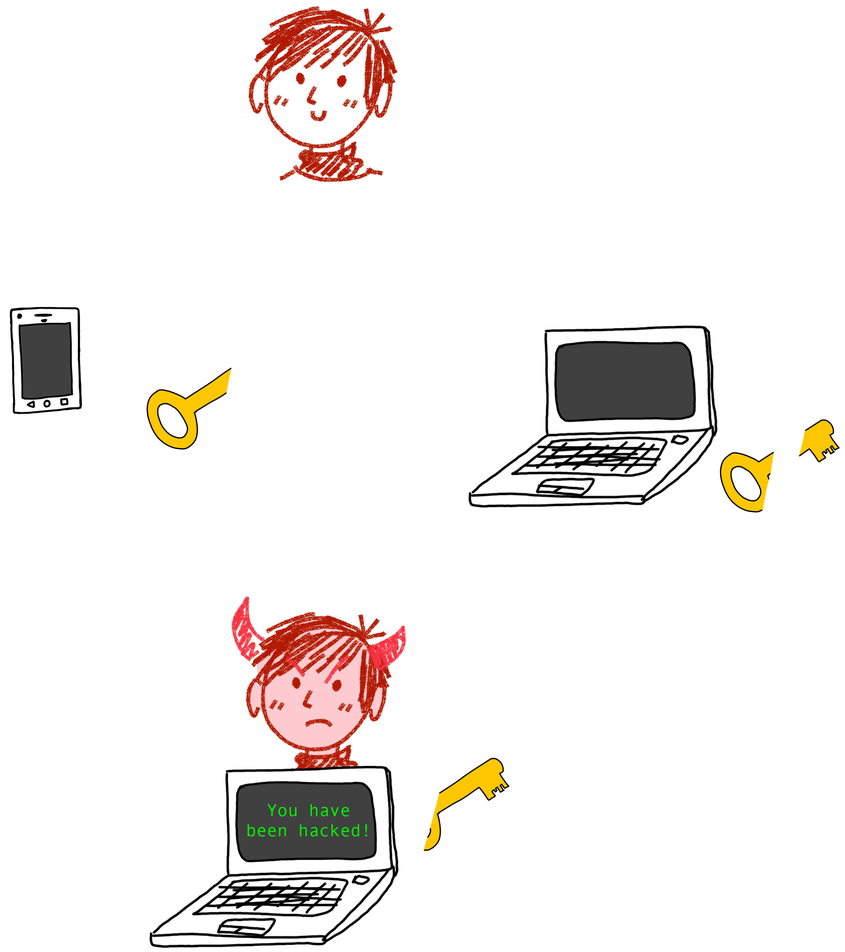
General Secret Sharing



The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device
- Authorized subsets of parties can reconstruct the secret using their shares

General Secret Sharing



The setting:

- A dealer holds a secret value
- The dealer can “split” the secret value into shares that is sends to each device
- Authorized subsets of parties can reconstruct the secret using their shares
- While unauthorized subsets cannot learn any new information about the secret from their share

Secret Sharing

A secret sharing scheme for an access structure $\Gamma \subseteq \{P_1, \ldots, P_n\}$ (over parties $P_1, \ldots, P_n$) with secret space $\mathscr{M}$ consists of two algorithms (Share, Recon):

Secret Sharing

A secret sharing scheme for an access structure $\Gamma \subseteq \{P_1, \ldots, P_n\}$ (over parties $P_1, \ldots, P_n$) with secret space $\mathscr{M}$ consists of two algorithms (Share, Recon):

Access structure could be something like
"Alice + Bob is allowed, Alice + Carol is allowed, but Bob + Carol is not allowed"
or "any set of at least 3 people"

Secret Sharing

A secret sharing scheme for an access structure $\Gamma \subseteq \{P_1, \ldots, P_n\}$ (over parties $P_1, \ldots, P_n$) with secret space $\mathscr{M}$ consists of two algorithms (Share, Recon):

- **Sharing algorithm** Share takes a share s and outputs n shares $(s_1, \ldots, s_n)$
- **Reconstruction algorithm** Recon takes as input shares $\{s_i\}_{i \in B}$. If $B \in \Gamma$, output s. Otherwise, output $\perp$

Secret Sharing

A secret sharing scheme for an access structure $\Gamma \subseteq \{P_1, \ldots, P_n\}$ (over parties $P_1, \ldots, P_n$) with secret space $\mathscr{M}$ consists of two algorithms (Share, Recon):

- **Sharing algorithm** Share takes a share s and outputs n shares $(s_1, \ldots, s_n)$
- **Reconstruction algorithm** Recon takes as input shares $\{s_i\}_{i \in B}$. If $B \in \Gamma$, output s. Otherwise, output $\perp$

Correctness: For every $s \in \mathscr{M}$, if $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$, then for every $B \in \Gamma$ it holds that $s = \mathsf{Recon}(\{s_i\}_{i \in B})$

Secrecy: For every $s, s' \in \mathscr{M}$ and every $B \notin \Gamma$,

$$\left\{ \{s_i\}_{i \in B} \mid (s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s) \right\} \approx_s \left\{ \{s_i\}_{i \in B} \mid (s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s') \right\}$$

That is, the distributions of $\{s_i\}_{i \in B}$, for every possible secret is statistically close.

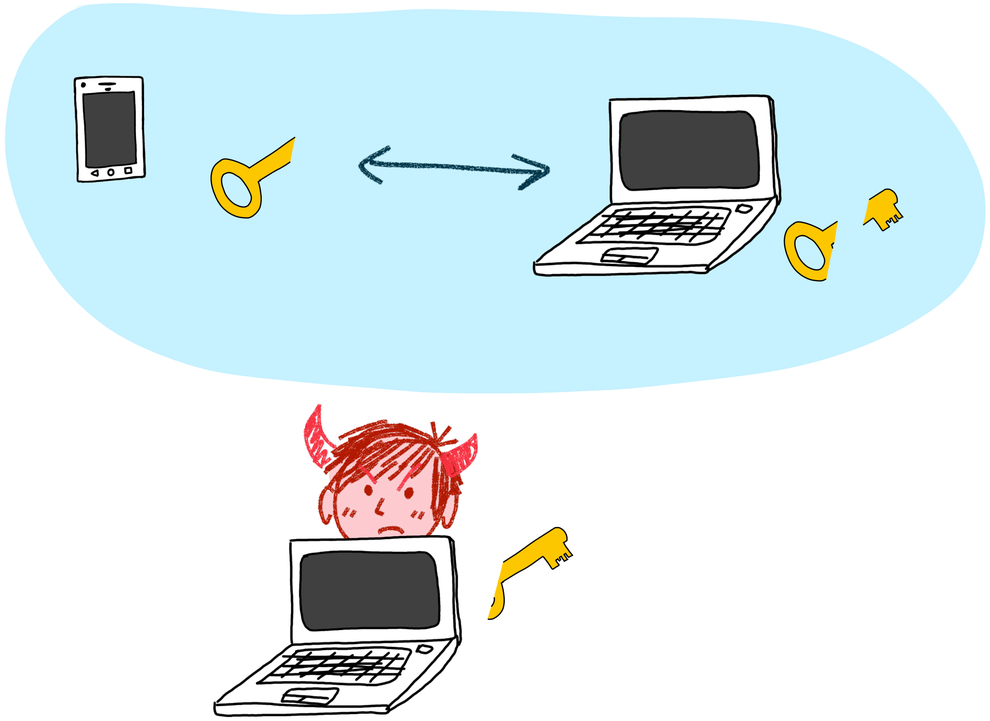
Threshold Secret Sharing

- Threshold secret sharing is a special case of general secret sharing

- Denote $(t+1)$-out-of-n secret sharing, for $t < n$

- Access structure consists of all sets of size at least $t+1$

- Forbidden structure consists of all sets of size at most t



n-out-of-n Additive Secret Sharing

Suppose we want a n-out-of-n secret sharing of $s \in \mathbb{Z}_q$ (where q is prime)

- Share(s):
  - Choose $s_1, \ldots, s_{n-1}$ uniformly at random from $\mathbb{Z}_q$
  - Compute $s_n = s - (s_1 + \ldots + s_{n-1}) \bmod q$
- Recon($s_1, \ldots, s_n$) $= s_1 + \ldots + s_n \bmod q$

Notice, without all n shares, you don't learn anything about s!

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and Alice wants to share the value 5

$$s_1 \leftarrow \mathbb{Z}_{13}$$
$$s_2 \leftarrow \mathbb{Z}_{13}$$
$$s_3 \leftarrow \mathbb{Z}_{13}$$
$$s_4 \leftarrow \mathbb{Z}_{13}$$
$$s_5 = 5 - s_1 - s_2 - s_3 - s_4$$

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and Alice wants to share the value 5

$s_1 \leftarrow \mathbb{Z}_{13}$

$s_2 \leftarrow \mathbb{Z}_{13}$

$s_3 \leftarrow \mathbb{Z}_{13}$

$s_4 \leftarrow \mathbb{Z}_{13}$

$s_5 = 5 - s_1 - s_2 - s_3 - s_4$

$s_1 = 4$

$s_2 = 8$

$s_5 = 7$

$s_3 = 10$

$s_4 = 2$

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and Alice wants to share the value 5

$s_1 \leftarrow \mathbb{Z}_{13}$

$s_2 \leftarrow \mathbb{Z}_{13}$

$s_3 \leftarrow \mathbb{Z}_{13}$

$s_4 \leftarrow \mathbb{Z}_{13}$

$s_5 = 5 - s_1 - s_2 - s_3 - s_4$

$s_1 = 4$

$s_2 = 8$

$s_5 = 7$

$s_3 = 10$

$s_4 = 2$

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and Alice wants to share the value 5

$s_1 \leftarrow \mathbb{Z}_{13}$

$s_2 \leftarrow \mathbb{Z}_{13}$

$s_3 \leftarrow \mathbb{Z}_{13}$

$s_4 \leftarrow \mathbb{Z}_{13}$

$s_5 = 5 - s_1 - s_2 - s_3 - s_4$

$s_1 = 4$

$s_2 = 8$

$s_5 = 7$

$s_3 = 10$

$s_4 = 2$

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and
Alice wants to share the value 5

$s_1 \leftarrow \mathbb{Z}_{13}$

$s_2 \leftarrow \mathbb{Z}_{13}$

$s_3 \leftarrow \mathbb{Z}_{13}$

$s_4 \leftarrow \mathbb{Z}_{13}$

$s_5 = 5 - s_1 - s_2 - s_3 - s_4$

$s_2 = 8$

$s_1 = 4$

$s_5 = 7$

$s_3 = 10$

$s_4 = 2$

Additive Secret Sharing Example

Suppose we're working over $\mathbb{Z}_{13}$ and Alice wants to share the value 5

$s_1 \leftarrow \mathbb{Z}_{13}$

$s_2 \leftarrow \mathbb{Z}_{13}$

$s_3 \leftarrow \mathbb{Z}_{13}$

$s_4 \leftarrow \mathbb{Z}_{13}$

$s_5 = 5 - s_1 - s_2 - s_3 - s_4$

$s_1 = 4$

$s_2 = 8$

$s_3 = 10$

$s_5 = 7$

$s_4 = 2$

n-out-of-n Additive Secret Sharing

We can also do this over bits using XOR. To share a value $s \in \{0,1\}^m$:

- Share(s):
  - Choose $s_1, \ldots, s_{n-1}$ uniformly at random from $\{0,1\}^m$
  - Compute $s_n = s \oplus s_1 \oplus \ldots \oplus s_{n-1}$
- Recon($s_1, \ldots, s_n$) $= s_1 \oplus \ldots \oplus s_n$

Do you notice something when $n = 2$?

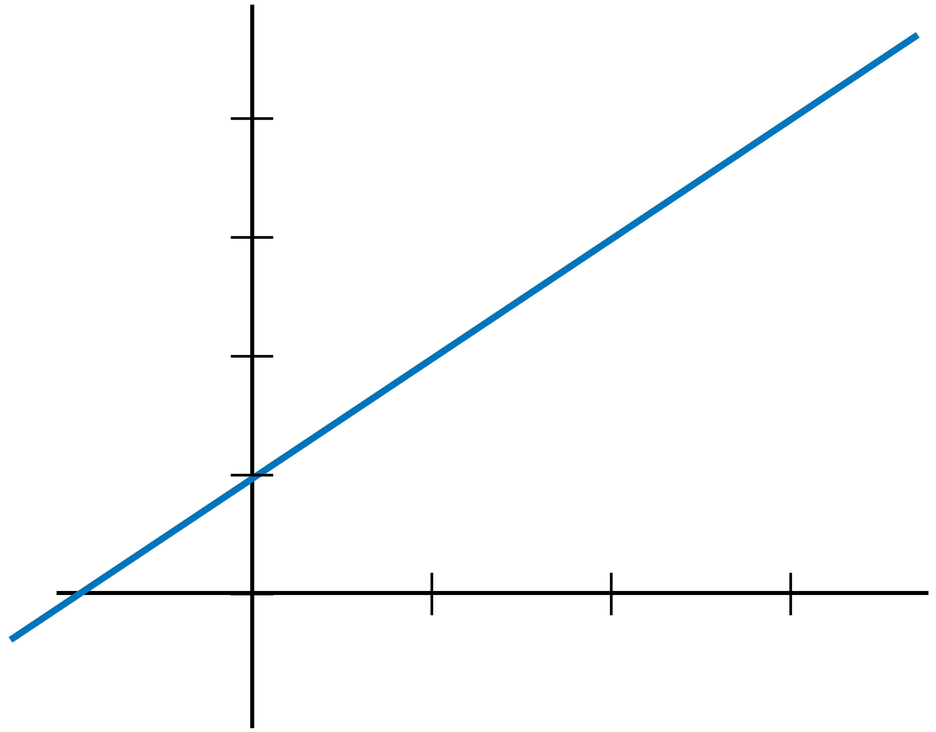
$(t+1)$-out-of-n Secret Sharing

- What if we want to only require a smaller subset of shares to reconstruct?
- Naive solution:
  - Dealer shares s using a $(t+1)$-out-of-$(t+1)$ secret sharing scheme for every subset of $t+1$ parties
  - This works because every set of t parties can't reconstruct, but every set of $t+1$ *can* reconstruct
  - How many subsets are there? $\binom{n}{t+1}$
- Can we do better?

Shamir’s Secret Sharing

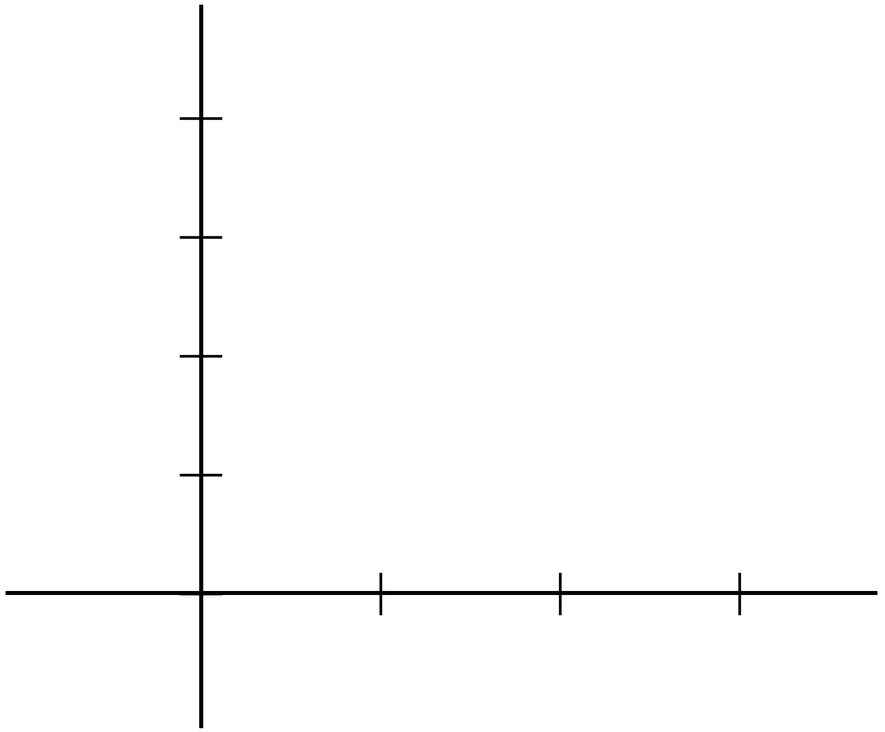
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$



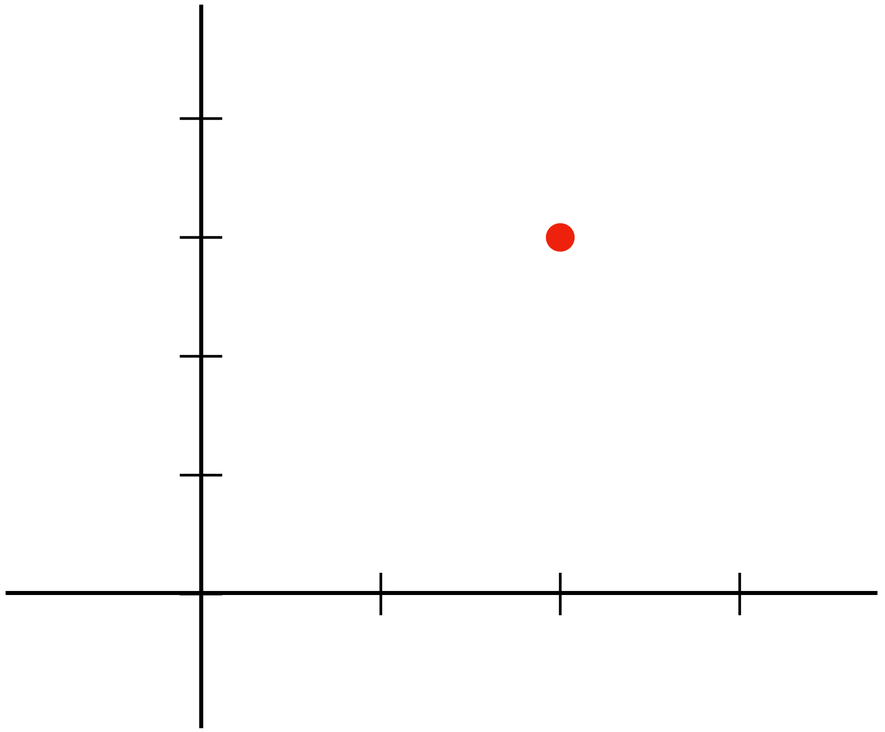
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?



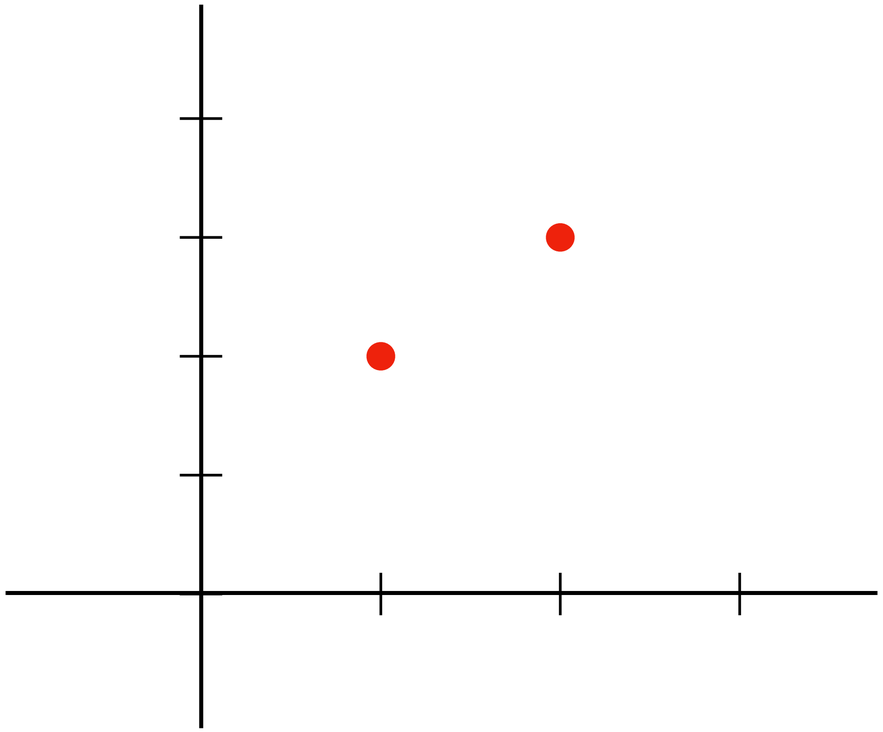
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?



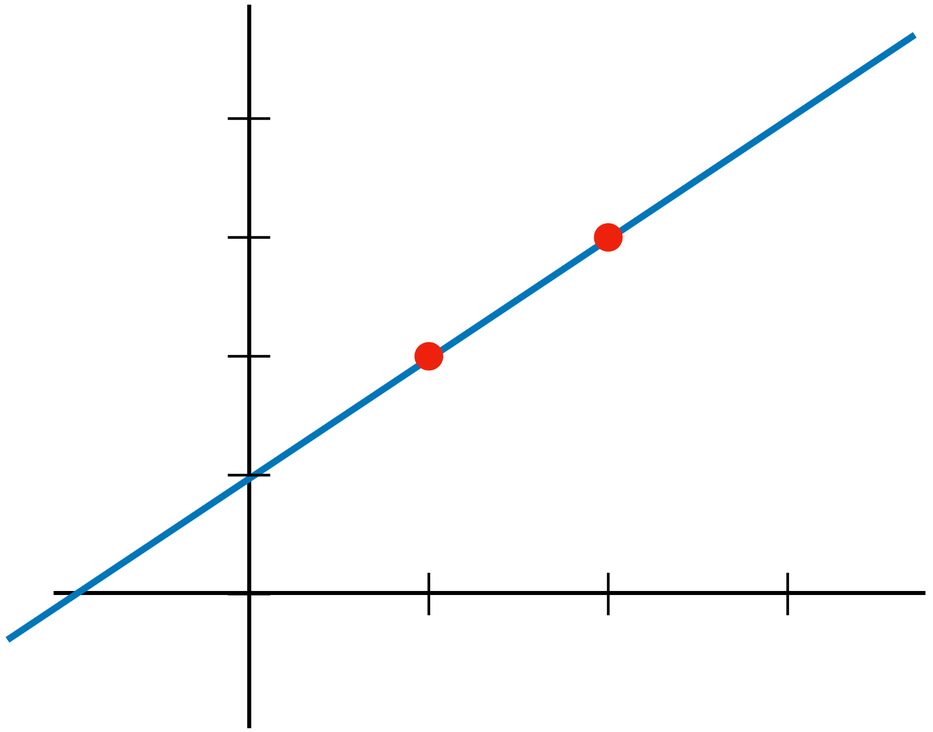
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?



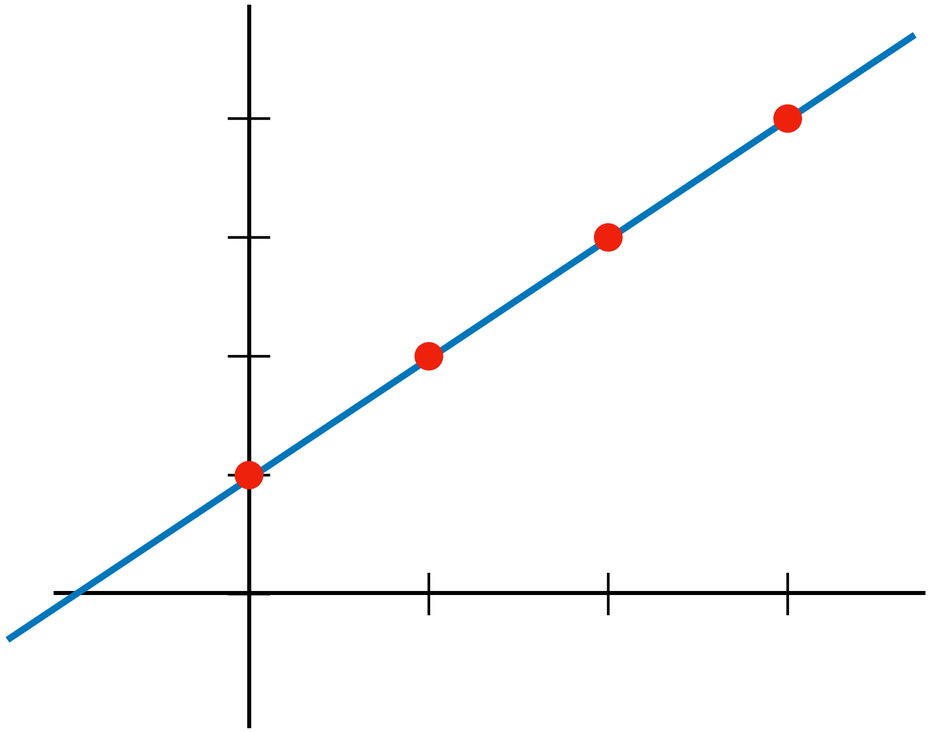
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?



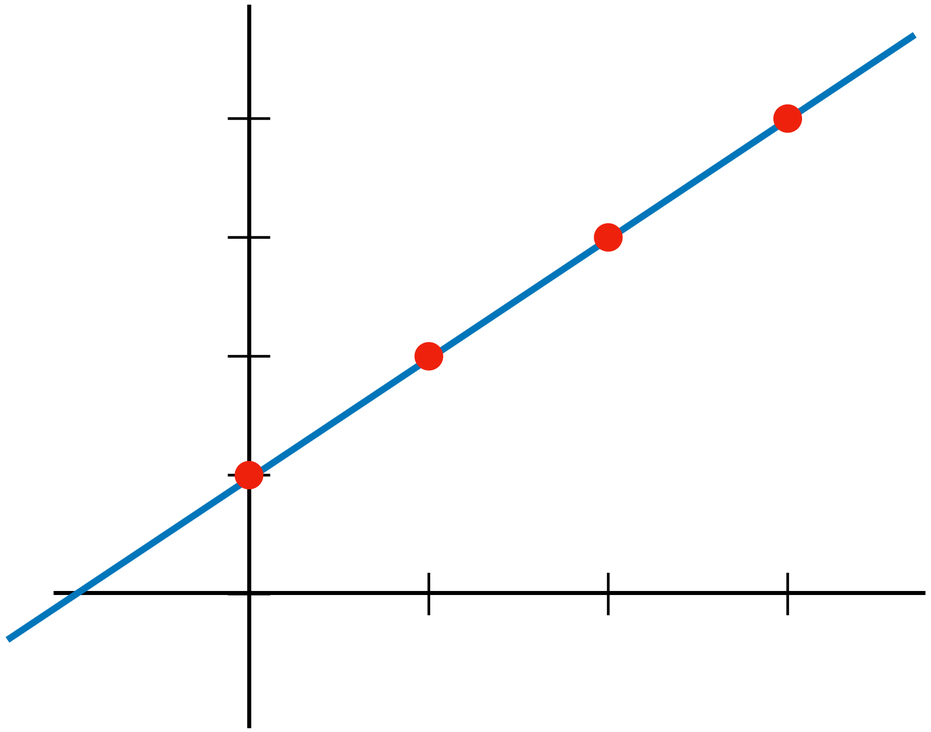
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?



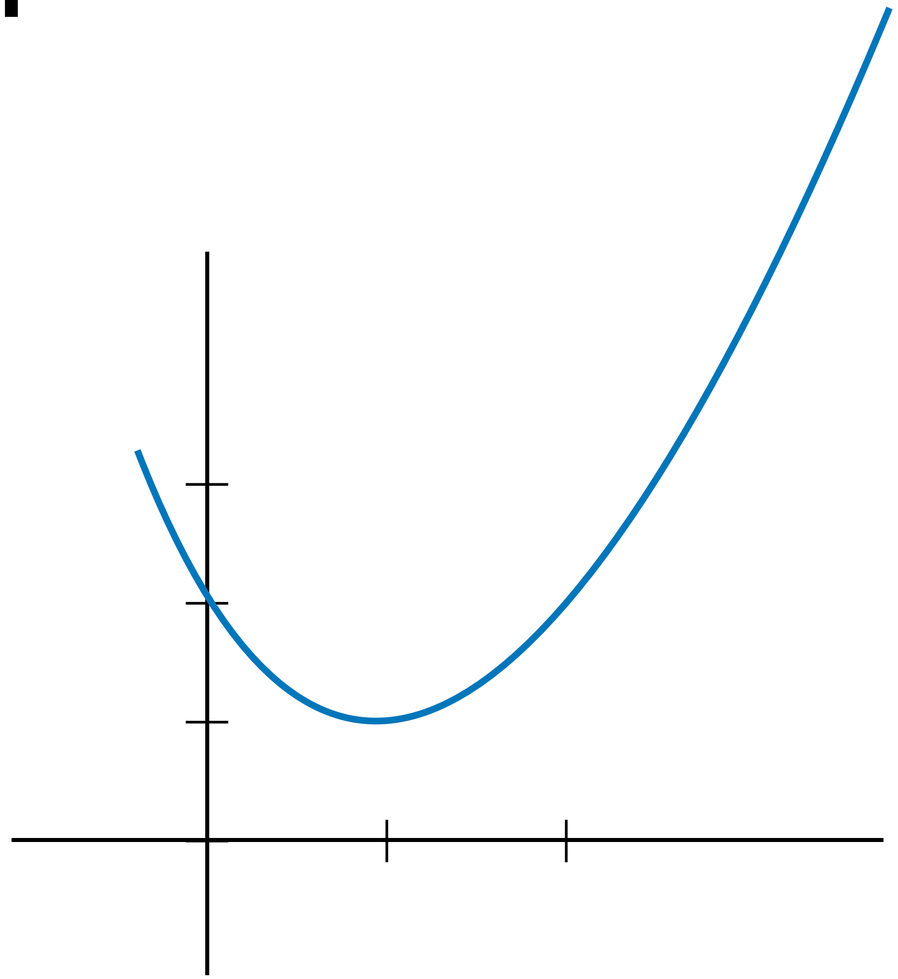
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?
  - What about $y = a_2 x^2 + a_1 x + a_0$?



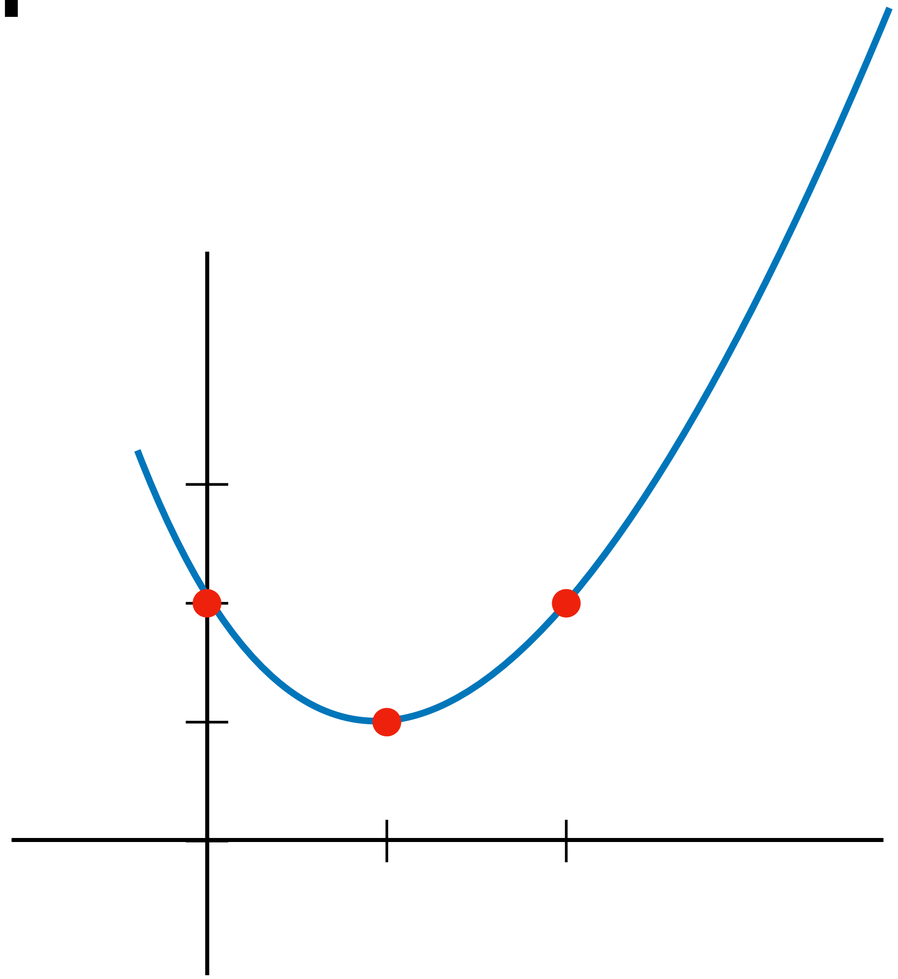
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?
  - What about $y = a_2 x^2 + a_1 x + a_0$?



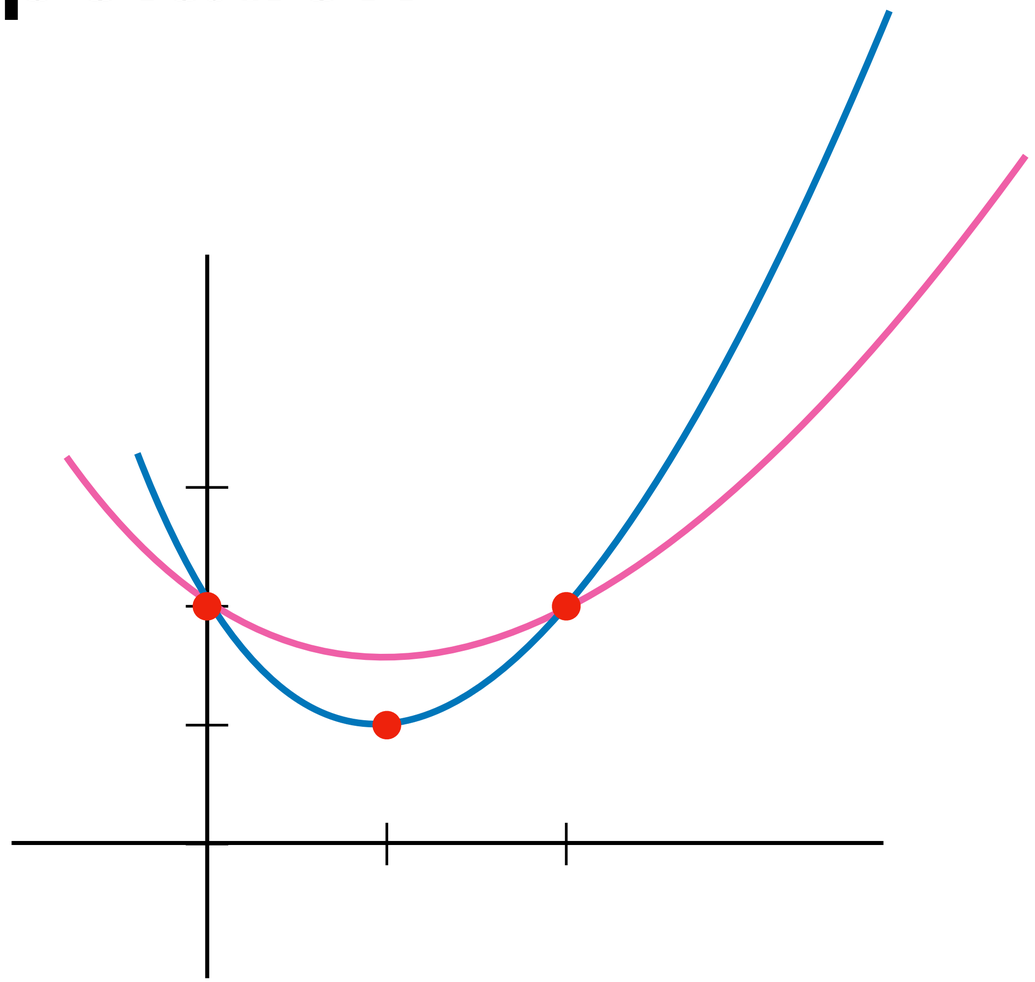
Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?
  - What about $y = a_2 x^2 + a_1 x + a_0$?



Polynomial Interpolation

- What is the equation for a line?
  - $y = a_1 x + a_0$
- Let's suppose I hid the line.
  - How many points would you need to reconstruct it?
  - What about $y = a_2 x^2 + a_1 x + a_0$?



Notice that any $n + 1$ points fully determine a degree n polynomial!

Some Helpful Number Theory: Fields

Definition: A **field** $(\mathbb{F}, +, \cdot)$ consists of a set $\mathbb{F}$ and an addition $(+)$ and multiplication $(\cdot)$ operations

- Operations $+, \cdot$ are **associative** and **commutative**
- $(\mathbb{F}, +)$ is a group with identity 0
- $(\mathbb{F}^*, \cdot)$ is a group with identity 1 where $\mathbb{F}^* = \mathbb{F} \backslash \{0\}$
- **Distributive**: $x \cdot (y + z) = x \cdot y + x \cdot z$

Example of infinite fields: Rationals $\mathbb{Q}$, reals $\mathbb{R}$, complex $\mathbb{C}$,

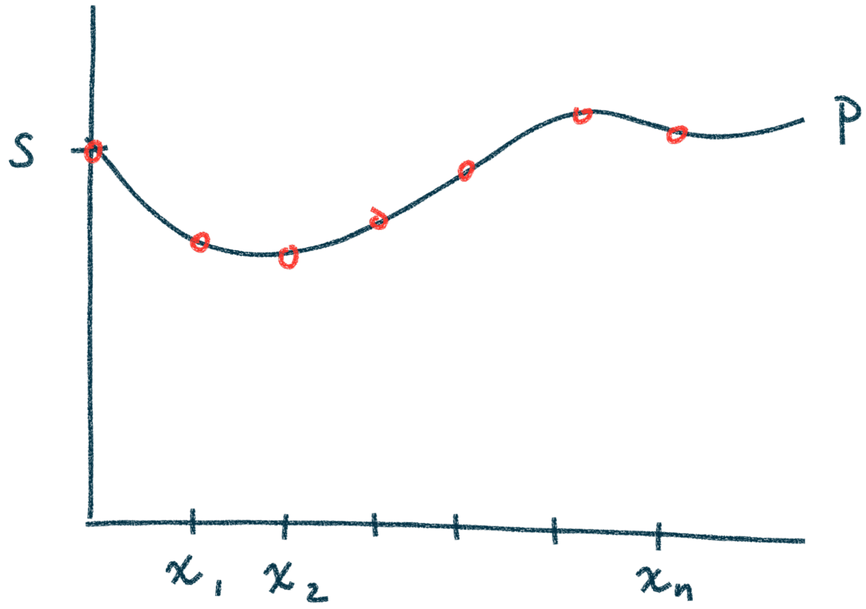
Example finite fields: $\mathbb{Z}_q$ where p is p prime

Polynomial Interpolation

Theorem:

Let $\{(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})\} \subseteq \mathbb{F}^2$ be a set of points whose x_i values are distinct and $\mathbb{F}$ is a finite field.

Then there is a unique degree-d polynomial $F(X)$ with real coefficients that satisfies $y_i = F(x_i)$ for all i



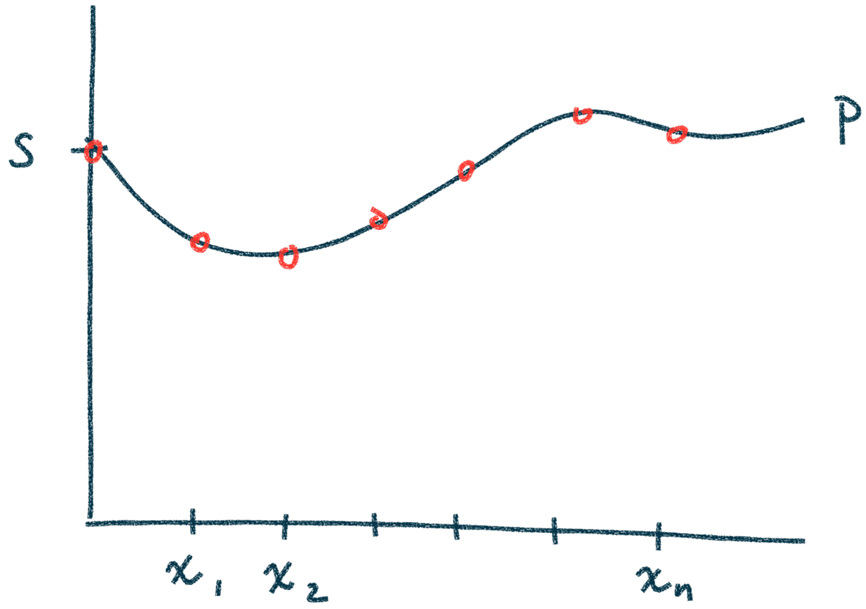
Polynomial Interpolation

Lemma (Lagrange interpolation):

Given $n+1$ points $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ in which each x_i is distinct, the unique degree-d polynomial that interpolates these points is

$$F(x) = \sum_{i=1}^{d+1} a_i \lambda_i(x)$$

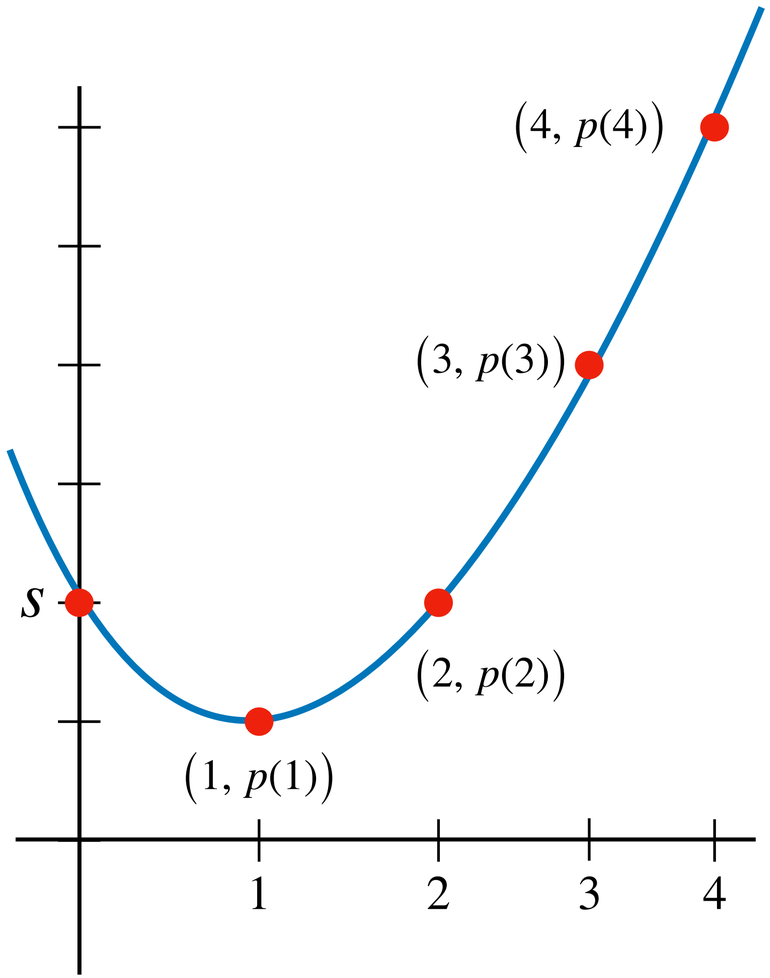
where $\lambda_i(x) = \prod_{j=1; j \neq i}^{d+1} \frac{x - x_j}{x_i - x_j}$



Shamir's Secret Sharing

A $(t+1)$-out-of-n secret sharing of a value $s \in \mathbb{F}$:

- Share(s):
  - Choose t random coefficients $a_1, \ldots, a_t \in \mathbb{F}$ and set $a_0 = s$
  - Define the polynomial as $p(x) = \sum_{j=0}^{t} a_j x^j$
  - Set each share as $s_i = p(i)$
- Recon $\left(\left\{ (i, s_i) \right\}_B \right)$:
  - Use Lagrange interpolation to reconstruct $p(x)$ passing through $t+1$ points
  - Output $s = p(0)$

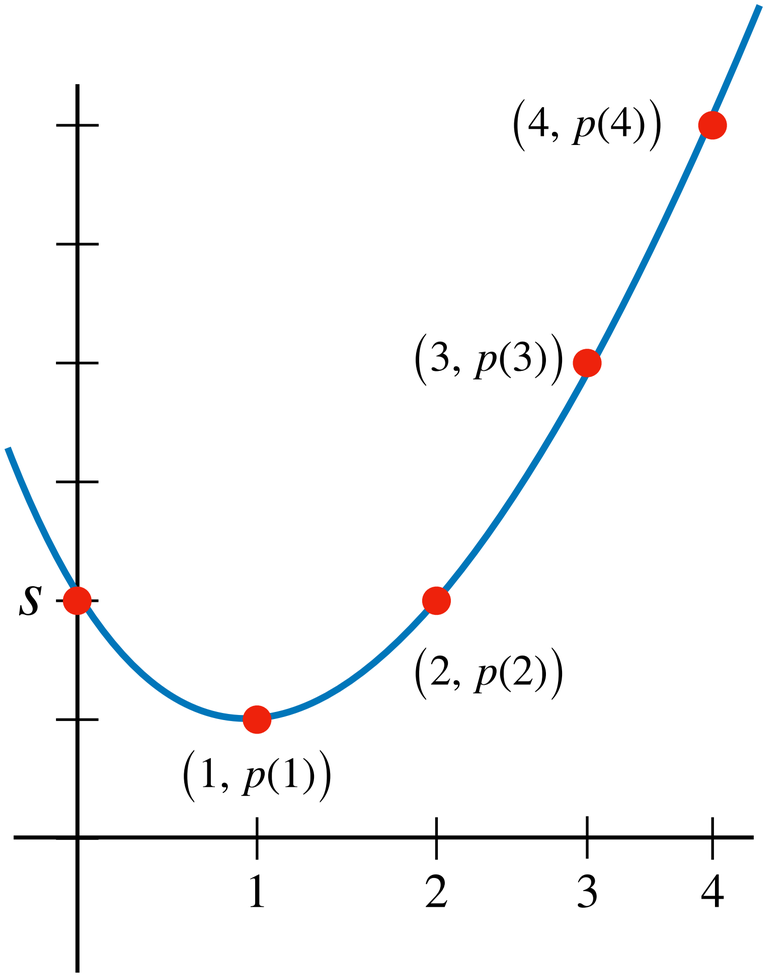


Shamir's Secret Sharing

A $(t+1)$-out-of-n secret sharing of a value $s \in \mathbb{F}$:

- Share(s):
  - Choose t random coefficients $a_1, \ldots, a_t \in \mathbb{F}$ and set $a_0 = s$
  - Define the polynomial as $p(x) = \sum_{j=0}^{t} a_j x^j$
  - Set each share as $s_i = p(i)$
- Recon $\left(\{ (i, s_i) \}_B \right)$:
  - Use Lagrange interpolation to reconstruct $p(x)$ passing through $t+1$ points
  - Output $s = p(0)$

What happens if I don't have enough points?

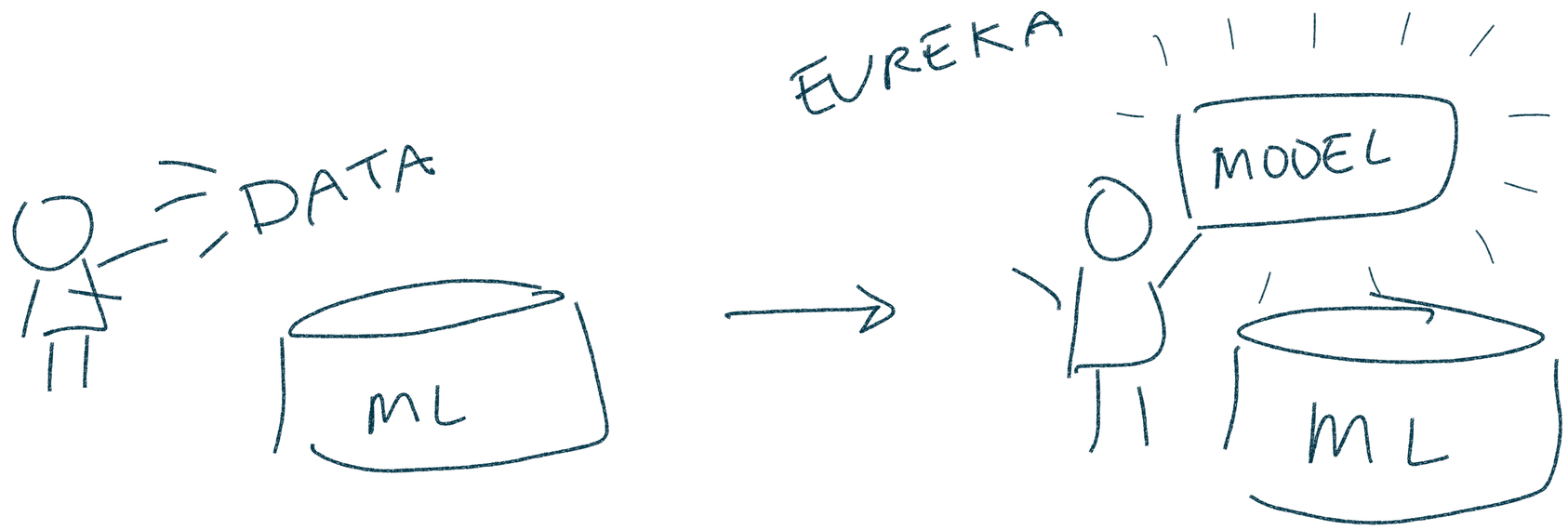


Multiparty Computation

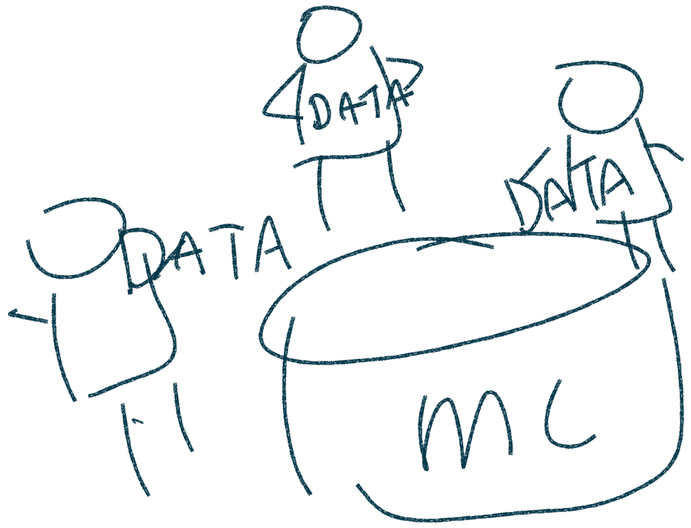
A brief detour into the land of Machine Learning

A brief detour into the land of Machine Learning

A brief detour into the land of Machine Learning

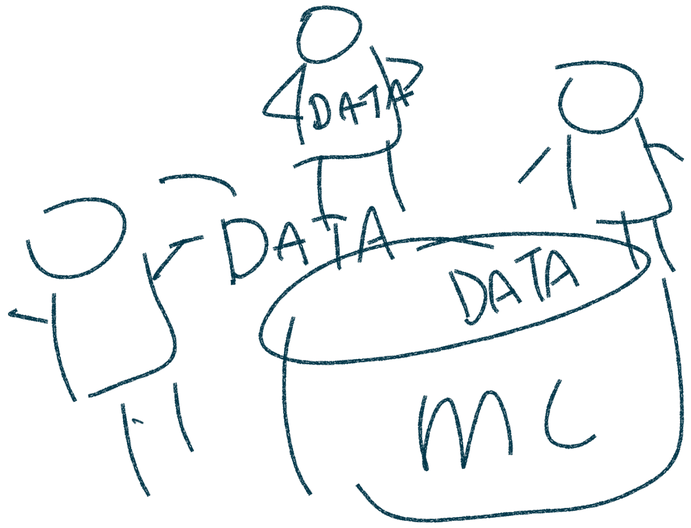


A brief detour into the land of Machine Learning with Friends



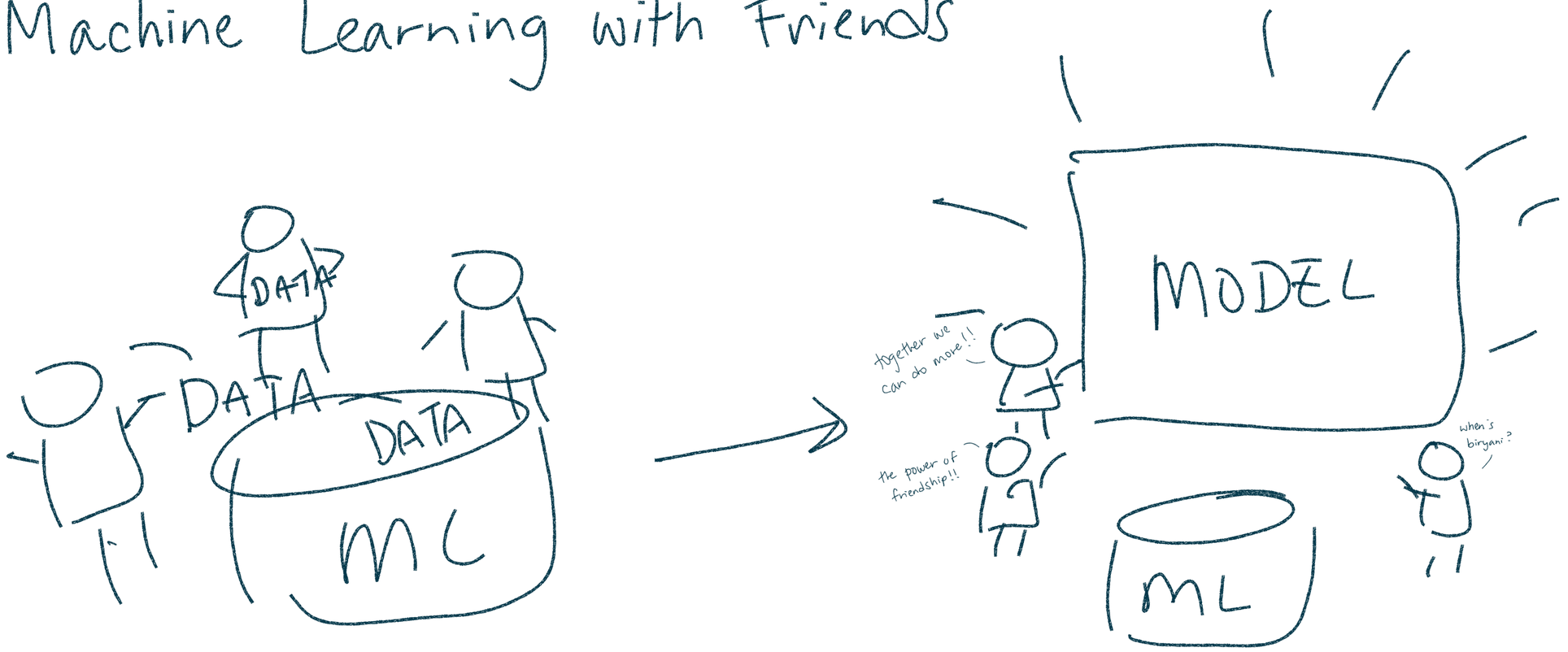
* Disclaimer: Friendship not req to compute ML models together

A brief detour into the land of Machine Learning with Friends



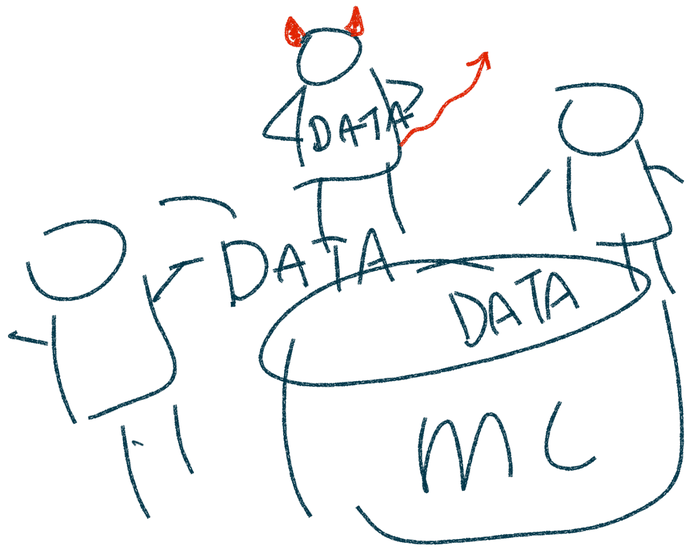
* Disclaimer: Friendship not req to compute ML models together

A brief detour into the land of Machine Learning with Friends

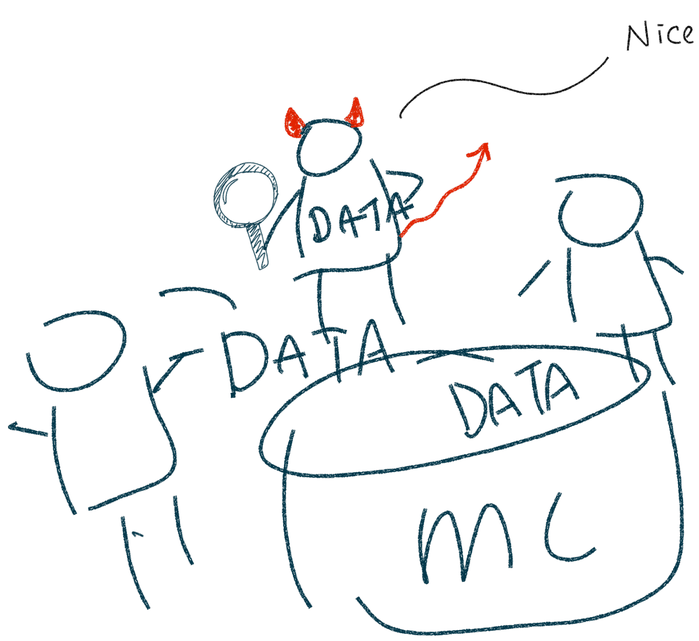


* Disclaimer: Friendship not req to compute ML models together

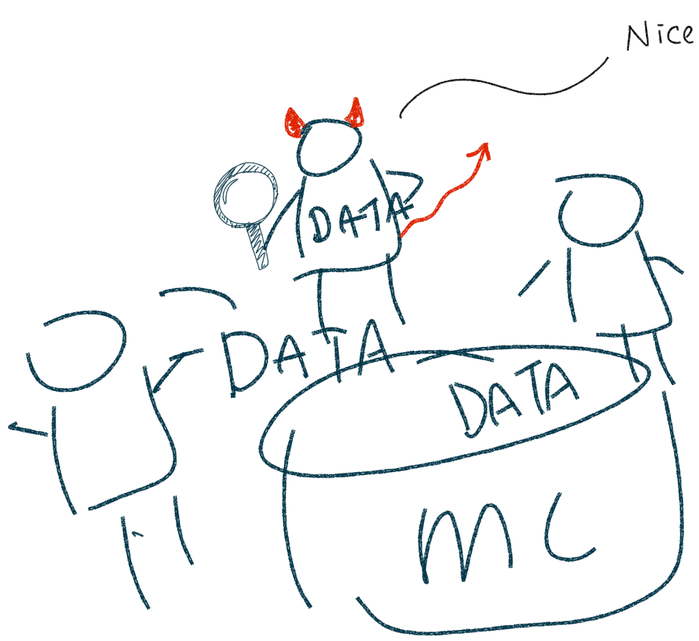
A brief detour into the land of Machine Learning and Privacy



A brief detour into the land of Machine Learning and Privacy



A brief detour into the land of Machine Learning and Privacy



Is it possible for parties to jointly compute functions without revealing their inputs to other (potentially bad) parties?

Multiparty Computation (MPC)

- Each of n parties holds a private input x_i

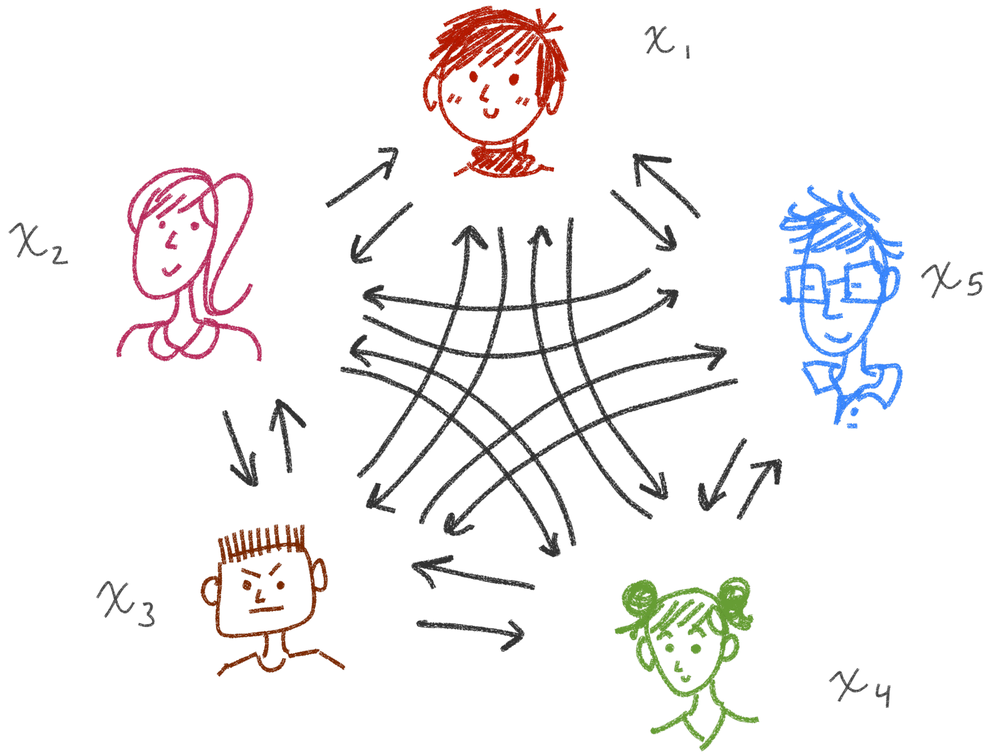
Multiparty Computation (MPC)

- Each of n parties holds a private input x_i

- Can communicate via private point-to-point channels

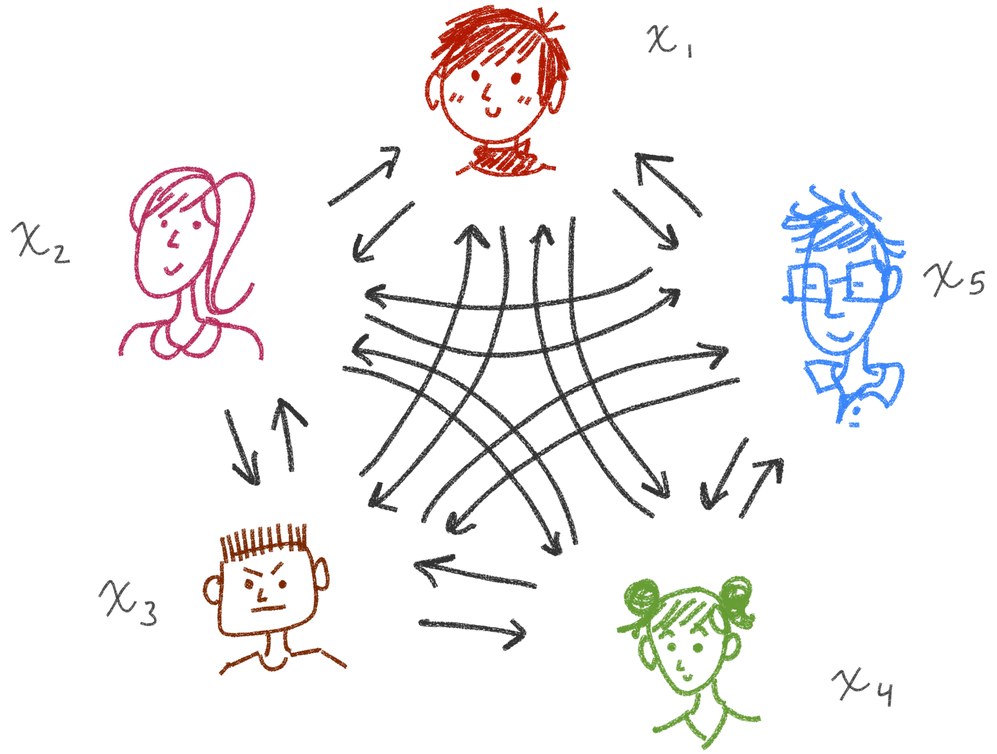
Multiparty Computation (MPC)

- Each of n parties holds a private input x_i
- Can communicate via private point-to-point channels
- They want to jointly compute some function $f(x_1, \ldots, x_n)$
- They don't want to reveal their inputs to each other



Multiparty Computation (MPC)

- Goal: design a protocol Π that computes f
- Properties:
  - **Correctness**: all parties learn $f(x_1, \ldots, x_n)$
  - **Security** (informal): no subset $\leq t$ parties "learns anything" about other parties inputs beyond what can be learned from the output



Formalizing Security

For any set $S \subseteq \{1,\ldots,n\}$ and any $x_1, \ldots, x_n$ let $View_{\Pi}(A, x_1, \ldots, x_n)$ be the view of the parties $i \in A$ during the execution of Π with the given inputs. The view consists of:

- The inputs $x_S = \{x_i \ : \ i \in A\}$
- The randomness of the parties $i \in A$
- All protocol messages receives by parties $i \in A$

We want the random variable $View_{\Pi}(A, x_1, \ldots, x_n)$ to only depend on x_A and $f(x_1, \ldots, x_n)$

Why define security like this?

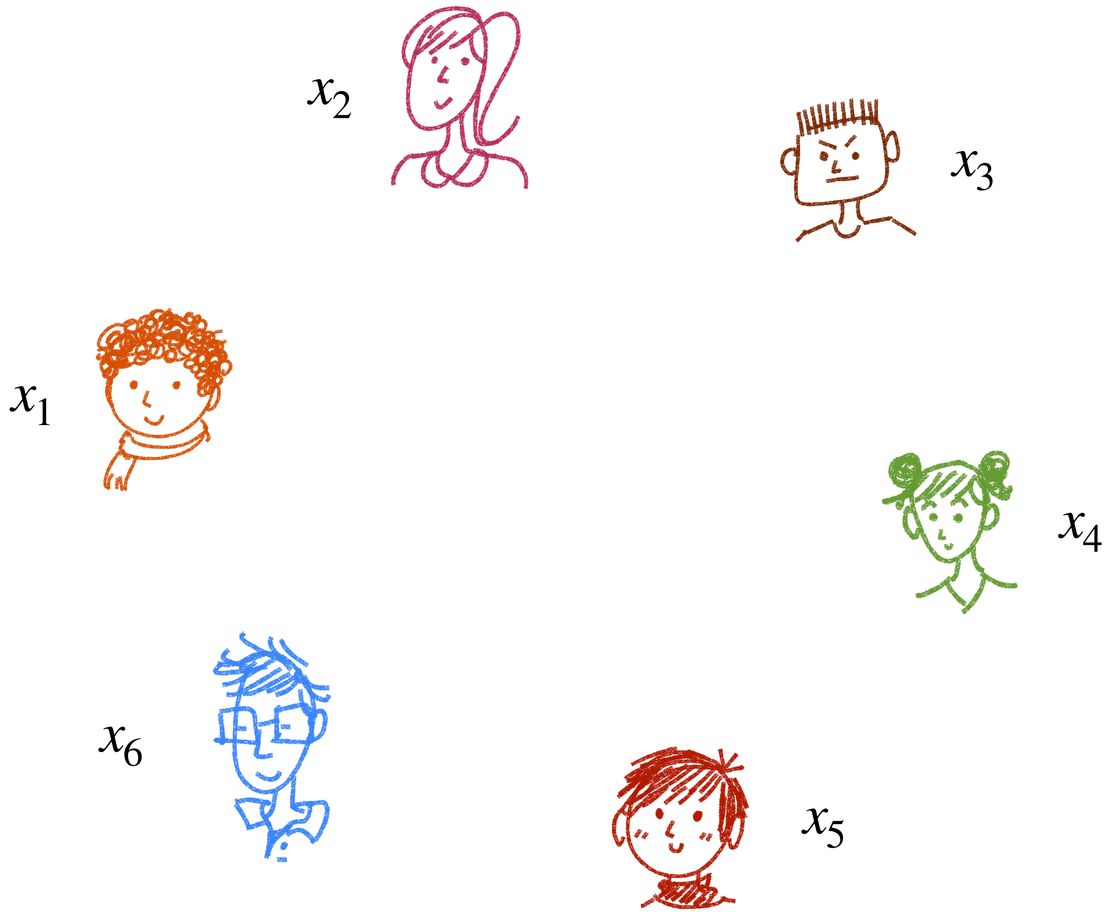
- Why do we say the adversary's view can only depend on its inputs and the output?
  - Why not say it can *only* depend on the adversarial inputs?
  - Hint: Can you think of any functions that may reveal part of their inputs?

A toy example: Computing Sum

Suppose parties have $x_1, \ldots, x_n$

How could we compute

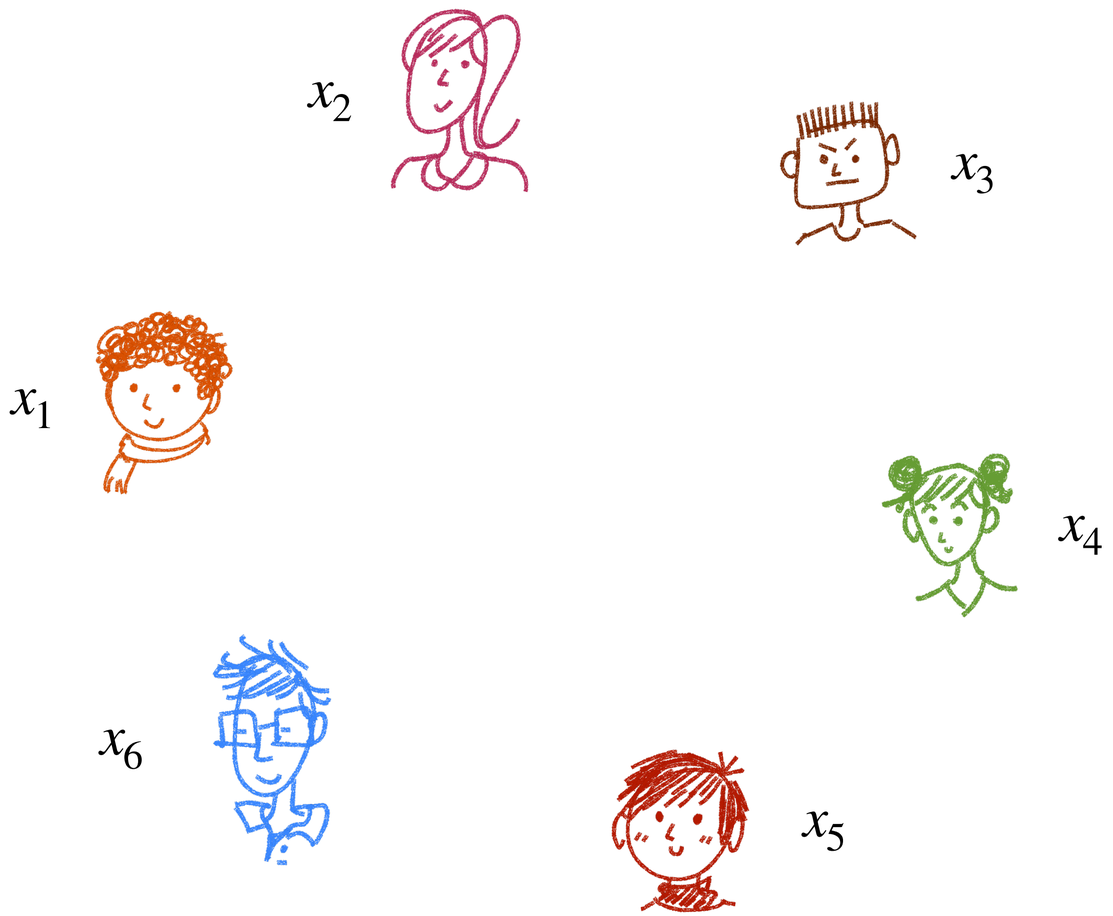
$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i ?$$



Limitations

What if parties don't follow the protocol honestly?

- Semihonest security vs Malicious security



Computing on Polynomials

Suppose we had $p_1(x) = a_0 + a_1 x$ and $p_2(x) = b_0 + b_1 x$

- What is the degree of $c \cdot p_1(x)$, where c is a scalar?
- What is the degree of $p_1(x) + p_2(x)$?
- What is the degree of $p_1(x) \cdot p_2(x)$

Computing on Polynomials

Suppose we had $p_1(x) = \sum_{i=0}^{t} a_i x^i$ and $p_2(x) = \sum_{i=0}^{t} b_i x^i$

- What is the degree of $c \cdot p_1(x)$, where c is a scalar?
- What is the degree of $p_1(x) + p_2(x)$?
- What is the degree of $p_1(x) \cdot p_2(x)$

Computing on Polynomials

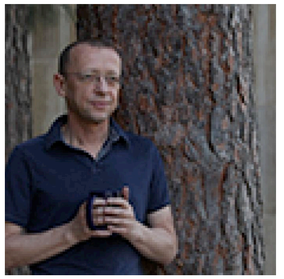
Suppose we had $p_1(x) = \sum_{i=0}^{t} a_i x^i$ and $p_2(x) = \sum_{i=0}^{t} b_i x^i$

- What is the degree of $c \cdot p_1(x)$, where c is a scalar?
- What is the degree of $p_1(x) + p_2(x)$?
- What is the degree of $p_1(x) \cdot p_2(x)$

How many times can I do this?

Ben-Or Goldwasser Wigderson (BGW)

- Arithmetic computations over a field
- Perfect secrecy
  - Tolerate computationally unbounded adversaries
  - Zero error probability
- Two results for computing any efficiently computable function f:
  - Perfect, semi-honest security against $t < n/2$ corruptions
  - Perfect, malicious security against $t < n/3$ corruptions
- Number of rounds is proportional to the multiplicative depth



Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation

(Extended Abstract)

Michael Ben-Or*
Hebrew University

Shafi Goldwasser†
MIT

Avi Wigderson‡
Hebrew University

Abstract

Every function of n inputs can be efficiently computed by a complete network of n processors in such a way that:

1. If no faults occur, no set of size $t < n/2$ of players gets any additional information (other than the function value),
2. Even if Byzantine faults are allowed, no set of size $t < n/3$ can either disrupt the computation or get additional information.

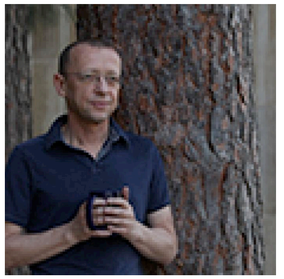
Furthermore, the above bounds on t are tight!

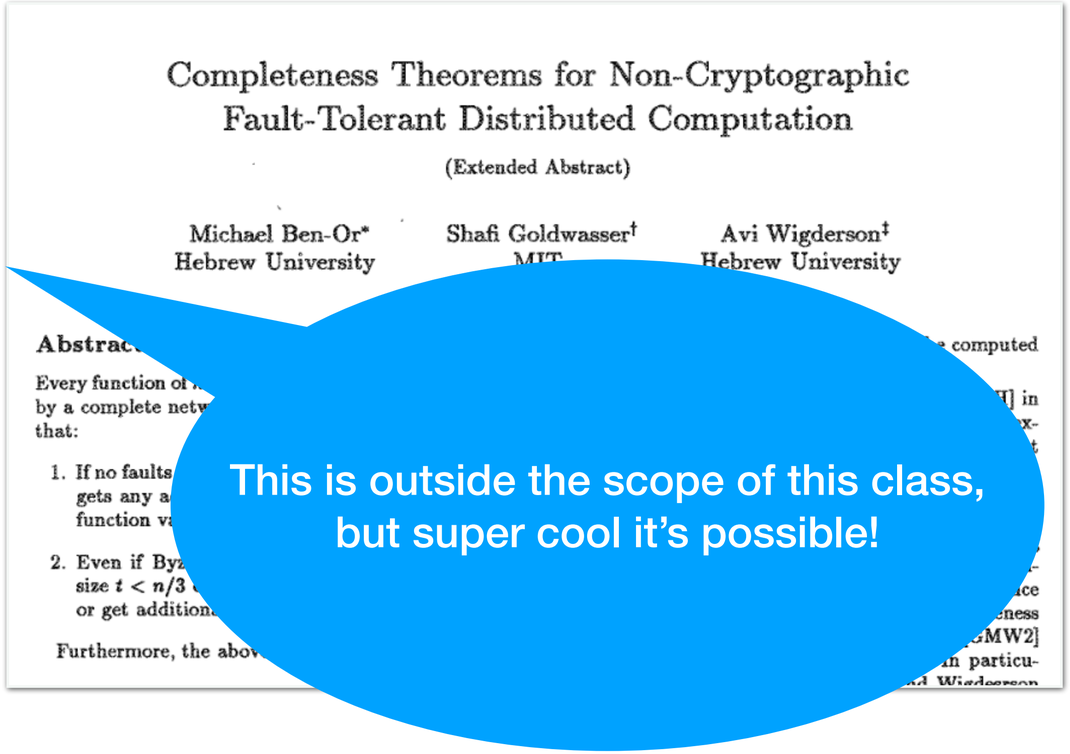
of certain (one-way) functions, that can be computed but not inverted by the player.

This simple assumption was postulated in [DH] in order to achieve the basic task of secure message exchange between two of the processors, but turned out to be universal! In subsequent years ingenious protocols based on the same assumption were given for increasingly harder tasks such as contract signing, secret exchange, joint coin flipping, voting and playing Poker. These results culminated, through the definition of zero-knowledge proofs [GMR], their existence for NP-complete problems [GMW1] in completeness theorems for two-party [Y1] and multi-party [GMW2] cryptographic distributed computation. In particular the results of Goldreich, Micali and Wigderson

Ben-Or Goldwasser Wigderson (BGW)

- Arithmetic computations over a field
- Perfect secrecy
  - Tolerate computationally unbounded adversaries
  - Zero error probability
- Two results for computing any efficiently computable function f:
  - Perfect, semi-honest security against $t < n/2$ corruptions
  - Perfect, malicious security against $t < n/3$ corruptions
- Number of rounds is proportional to the multiplicative depth





Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation

(Extended Abstract)

Michael Ben-Or* Hebrew University

Shafi Goldwasser† MIT

Avi Wigderson‡ Hebrew University

This is outside the scope of this class, but super cool it's possible!

Next Time

- Today: Secret Sharing
- Monday: Midterm Review
- Wednesday: Midterm
- After Spring Recess:
  - More Number Theory
  - Discrete Log Assumption
  - Public Key Crypto!