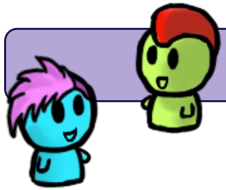
CS459/698
Privacy, Cryptography, Network and Data Security

Multi-Party Computation, PSI, PIR, Secret-Sharing

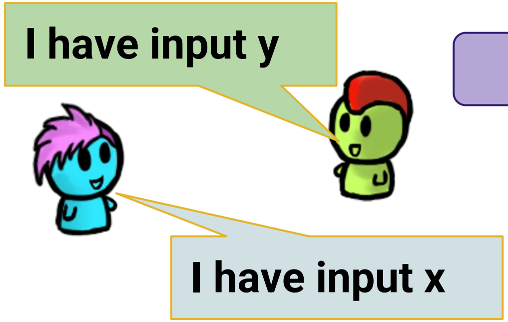
Spring 2025, Monday/Wednesday 2:30pm-3:50pm

What is Multi-Party Computation?



1) **At least two parties**

What is Multi-Party Computation?



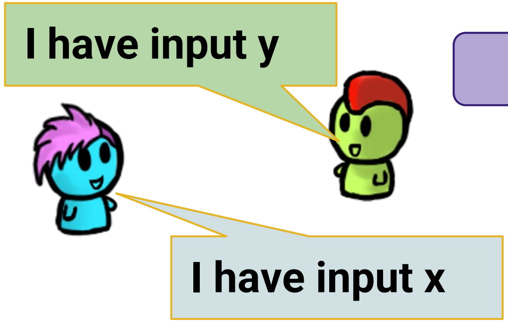
1) At least two parties

What is Multi-Party Computation?

1) At least two parties

2) Both Alice and Bob know a function f

What is Multi-Party Computation?

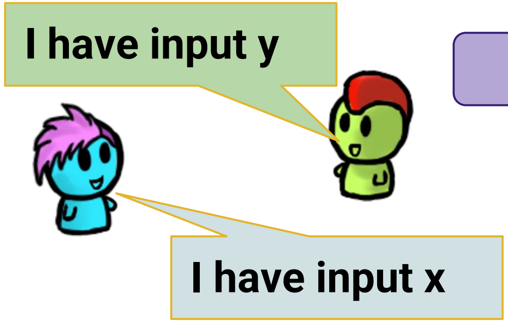


1) At least two parties

2) Both Alice and Bob know a function f

Goal: learn f(x, y) but <u>not</u> reveal anything else about x or y

What is Multi-Party Computation?



1) At least two parties

2) Both Alice and Bob know a function f

Goal: learn f(x, y) but <u>not</u> reveal anything else about x or y

Critical: Secret inputs, public outputs (to at least one party)

A Potential “Real-World” Example

I want to analyse sentence x (NLP)

A Potential “Real-World” Example

I want to analyse sentence x (NLP)

I have model parameters y…

But I want to keep my NN secret…

A Potential “Real-World” Example

I want to analyse sentence x (NLP)

I have model parameters y…

But I want to keep my NN secret…

Require: A function f over public parameters, but secret architecture

Goal: A MPC for f(x, y) such that only Alice learns the analysis of her sentence and Alice does not learn the NN

“Types” of MPC: Participant Set

Two-party

Multi-Party

MPC Server Model

- Assume n >> 3 clients with an input
  - E.g., collect statistics about emoji usage in texting

- Dedicate 2 (or 3) parties as computation nodes (servers)

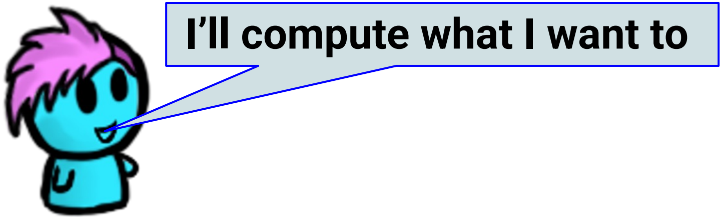
- The clients send “encrypted” versions of their inputs

- The servers perform multi-party computation
  - Decrypt input
  - Compute *f*

“Types” of MPC: Functionality

Yao’s Garbled Circuits

GMW Protocol



Generic

Generic functions:

A multi-party computation protocol that can be used for **“any” function f**

“Types” of MPC: Functionality

Yao’s Garbled Circuits

GMW Protocol

Yao Millionaire’s Problem

Private Set Intersection

Generic

Specific

Generic functions:

A multi-party computation protocol that can be used for **“any” function f**

Specific functions:

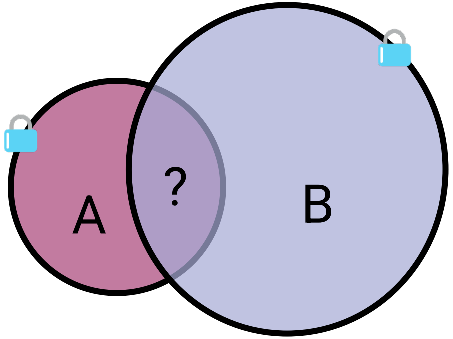
A multi-party computation protocol that can only be used for **a specific function f**

Private Set Intersection (PSI) – A *specific* MPC

- Alice has set **X** = $\{x_1, x_2, x_3, \ldots, x_n\}$
- Bob has set **Y** = $\{y_1, y_2, y_3, \ldots, y_m\}$
- They want to compute **Z = X ∩ Y** (but reveal nothing else)

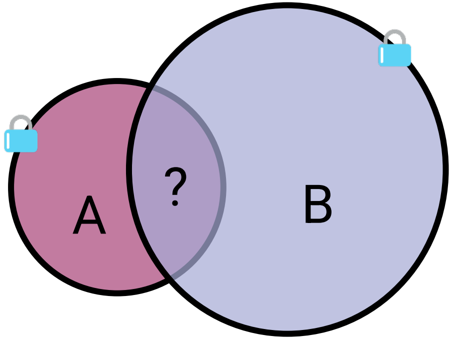
- Good real-world use case: private contact discovery
  - i.e., how many and which contacts do we have in common?

Private Set Intersections



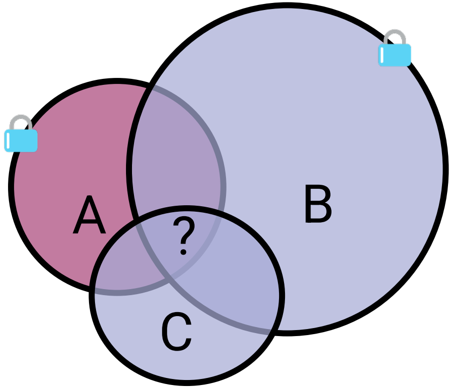
2-Party, One-Way PSI

A ⟶ B



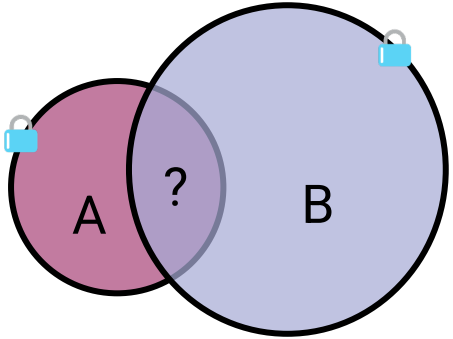
2-Party, Two-Way PSI

A ⟷ B



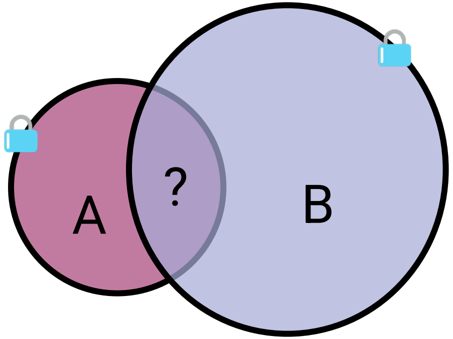
n-Party PSI

Private Set Intersections



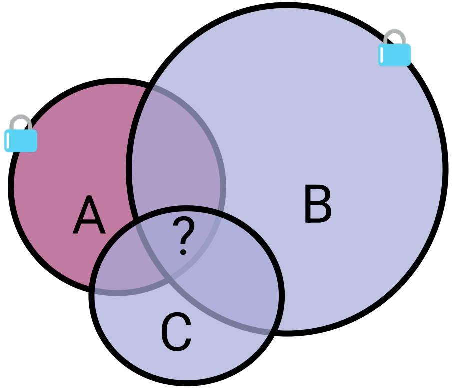
2-Party, One-Way PSI

A ⟶ B



2-Party, Two-Way PSI

A ⟷ B



n-Party PSI

Directionality | **Reducing Information Exchange** | **Multi-party** | **Varying Guarantees**

Strawman Protocol for PSI

- Alice permutes her set ***X***, Bob permutes his set ***Y***. Then:
  - For each $\boldsymbol{x} \in X$
    - For each $\boldsymbol{y} \in Y$
      - Compute ***x* =? *y***

- **Protocol for comparison (*x* =? *y*)**
  - Alice → Bob: $E_A(\boldsymbol{x})$
  - Bob: Choose random ***r*** and compute $\boldsymbol{c} = (E_A(\boldsymbol{x}) * E_A(-\boldsymbol{y}))^r$
    - Add encrypted value of ***x*** with encrypted value of ***–y*** (the negative of ***y***) and raise the result to the power of ***r***.
  - Bob → Alice: ***c*** (Bob has no idea what ***x*** is)
  - Alice: Knows whether $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

Strawman Protocol for PSI

- Alice permutes her set ***X***, Bob permutes his set ***Y***. Then:
  - For each $\boldsymbol{x} \in X$
    - For each $\boldsymbol{y} \in Y$
      - Compute ***x* =? *y***

- **Protocol for comparison (*x* =? *y*)**
  - Alice → Bob: $E_A(\boldsymbol{x})$
  - Bob: Choose random ***r*** and compute $\boldsymbol{c} = (E_A(\boldsymbol{x}) * E_A(-\boldsymbol{y}))^r$
    - Add encrypted value of ***x*** with encrypted value of ***–y*** (the negative of ***y***) and raise the result to the power of ***r***.
  - Bob → Alice: ***c*** (Bob has no idea what **x** is)
  - Alice: Knows whether $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

blinding factor

Strawman Protocol for PSI

- Alice permutes her set ***X***, Bob permutes his set ***Y***. Then:
  - For each $\boldsymbol{x} \in X$
    - For each $\boldsymbol{y} \in Y$
      - Compute ***x* =? *y***

- **Protocol for comparison (*x* =? *y*)**
  - Alice → Bob: $E_A(\boldsymbol{x})$
  - Bob: Choose random ***r*** and compute $\boldsymbol{c} = (E_A(\boldsymbol{x}) * E_A(-\boldsymbol{y}))^r$
    - Add encrypted value of ***x*** with encrypted value of ***−y*** (the negative of ***y***) and raise the result to the power of ***r***.
  - Bob → Alice: ***c*** (Bob has no idea what **x** is)
  - Alice: Knows whether $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

blinding factor

E_A and D_A are part of a homomorphic encryption scheme that supports operations on ciphertexts.
We will see more later!

Strawman Protocol for PSI

Complexity of O(xy)

More efficient solutions exist
e.g., based on precomputations

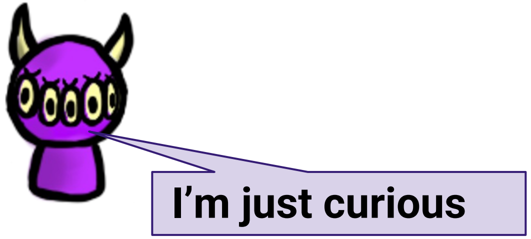
- Alice permutes her set ***X***, Bob permutes his set ***Y***. Then:
  - For each $\boldsymbol{x} \in X$
    - For each $\boldsymbol{y} \in Y$
      - Compute ***x* =? *y***

- **Protocol for comparison (*x* =? *y*)**
  - Alice → Bob: $E_A(\boldsymbol{x})$
  - Bob: Choose random ***r*** and compute $\boldsymbol{c} = (E_A(\boldsymbol{x}) * E_A(-\boldsymbol{y}))^r$
    - Add encrypted value of ***x*** with encrypted value of ***–y*** (the negative of ***y***) and raise the result to the power of ***r***.
  - Bob → Alice: ***c*** (Bob has no idea what **x** is)
  - Alice: Knows whether $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

blinding factor

E_A and D_A are part of a homomorphic encryption scheme that supports operations on ciphertexts.
We will see more later!

“Types” of MPC: Security

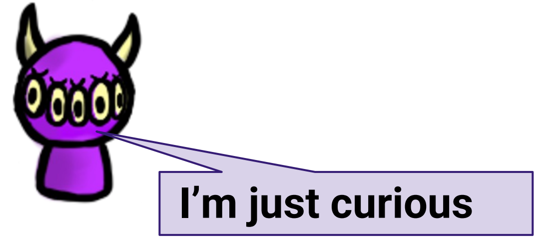


Passive

Passive security (security against **semi-honest adversaries**)

Each party **follows the protocol** but keeps a record of all messages and after the protocol is over, **tries to infer additional information** about the other parties’ inputs

“Types” of MPC: Security



Passive

Passive security (security against **semi-honest adversaries**)

Each party **follows the protocol** but keeps a record of all messages and after the protocol is over, **tries to infer additional information** about the other parties’ inputs

Active

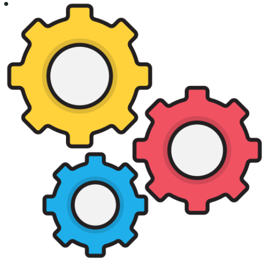
Active security (security against **malicious adversaries**)

Each party **may arbitrarily deviate from the protocol**. Either the protocol computes *f* or the protocol is aborted.

Relationship between Passive and Active Security

- Passive security is a **prerequisite** for active security
  - A protocol can be secure against passive adversaries but not active ones
  - A protocol secure against active adversaries is also secure against passive ones

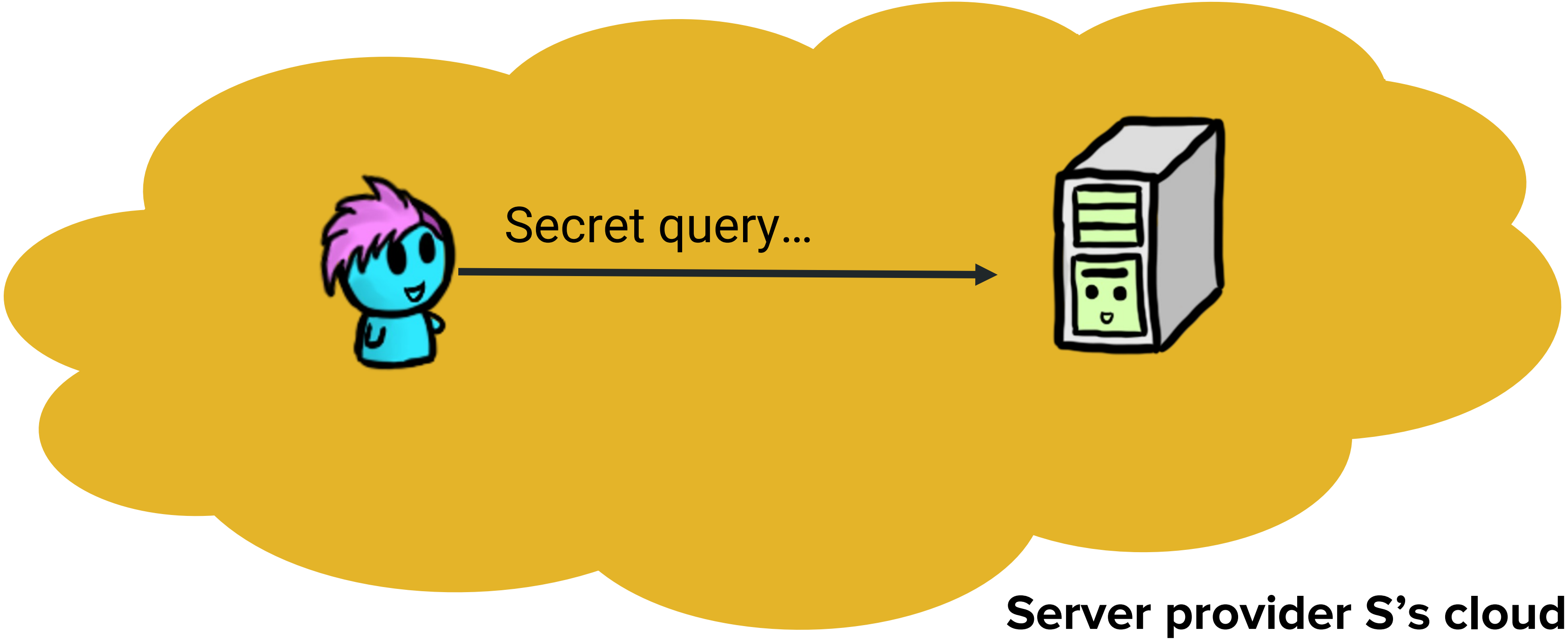
- Any protocol secure against passive adversaries can be turned into a protocol secure actives adversaries
  - E.g., by adding protocol steps proving the correct computation of each message:
    - Cryptographic commitments: can we detect a partipant deviates from the proto?
    - Validations: Are parameters within expected bounds?



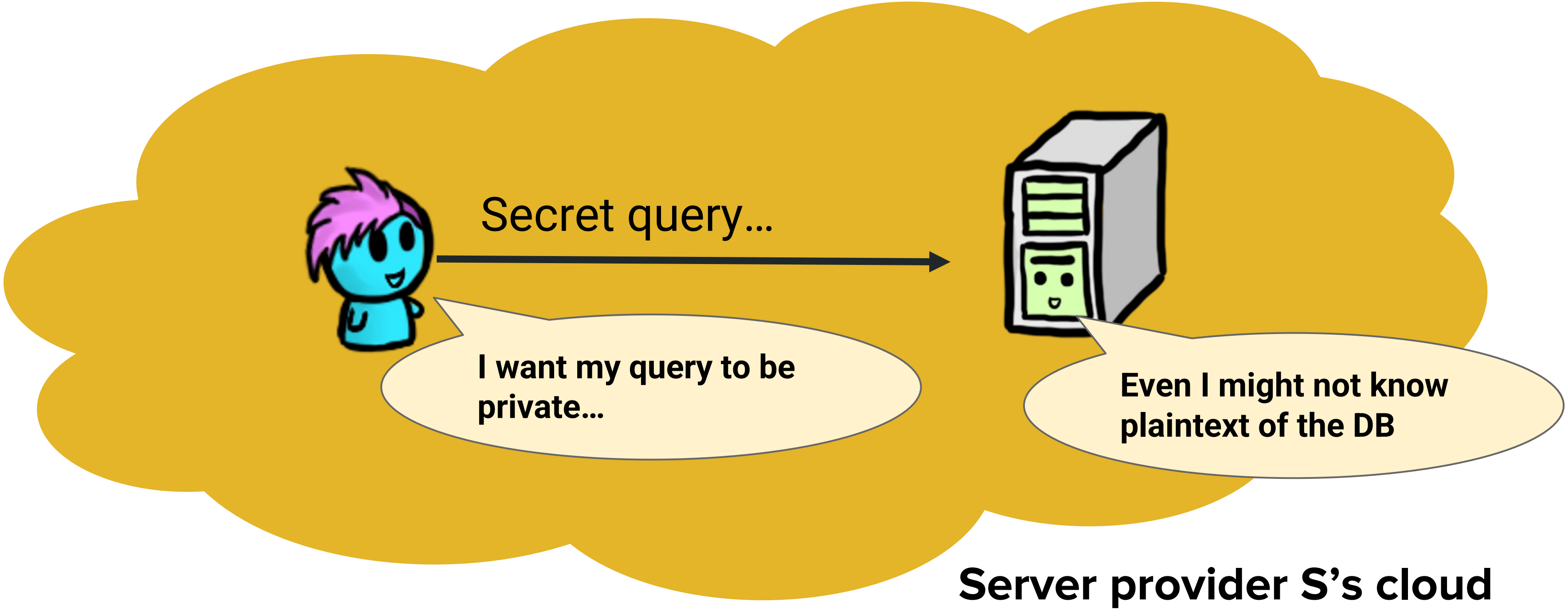
Known as Goldreich's compiler (Oded Goldreich, Knuth Prize 2017)

Private Information Retrieval (PIR)

Can we privately query a database?



Ideally…



Secret query…
I want my query to be private…
Even I might not know plaintext of the DB
Server provider S's cloud

Motivating Example (1)

- A server stores a list of “broken” passwords that appeared on the Internet

- The client wants to check whether the password they just created for an Internet site is in that database
  - **If it is**, they should not use it…
  - **If it is not**, but is revealed to the database, it should not be used either!

Motivating Example (1)

- A server stores a list of “broken” passwords that appeared on the Internet

- The client wants to check whether the password they just created for an Internet site is in that database
  - **If it is**, they should not use it…
  - **If it is not**, but is revealed to the database, it should not be used either!

- The client should query **without revealing** the password!

Motivating Example (2)

- Netflix stores movies in a database
    - 1. The Shawshank Redemption
    - 2. The Godfather
    - 3. The Dark Knight
    - 4. Lord of the Rings: The Two Towers
    - ...
- You request movies by index, say 1, 4, 2, ...
- Netflix caches your selection and gradually builds a profile on your movie preferences

Motivating Example (2)

- Netflix stores movies in a database
  - 1. The Shawshank Redemption
  - 2. The Godfather
  - 3. The Dark Knight
  - 4. Lord of the Rings: The Two Towers
  - ...
- You request movies by index, say 1, 4, 2, ...
- Netflix caches your selection and gradually builds a profile on your movie preferences
- The server should be queried **without learning** the item of interest!

PIR

Carol has index i

PIR

Carol has index i

Server has DB $d_1,\ldots,d_n$

PIR

Carol has index i

Server has DB $d_1,\ldots,d_n$

Goal 1: Correctness - Client learns d_i

PIR

Server has DB $d_1,\ldots,d_n$

Carol has index i

Goal 1: Correctness - Client learns d_i

Goal 2: Security - Server does not learn index i

Blatantly non-private protocol

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Protocol:

- User: show me **i**
- Server: here is $\mathbf{X_i}$

Analysis:

- No privacy!
- # of bits: 1 — very efficient

Trivially-private protocol

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Protocol:

- User: show me ***ALL indexes***
- Server: here is $\mathbf{\{X_1, X_2, ..., X_n\}}$

Analysis:

- Complete privacy!
- # of bits: n — very impractical

More solutions?

User asks for additional random indices

- **Drawback:** balance information leak vs communication cost

Anonymous communication:

- **Note:** this is in fact a different concern: it hides the identity of a user, not the fact that X_i is retrieved

Information-Theoretic PIR

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

An example 2-server IT-PIR protocol:

- User → Server 1: $\mathbf{Q_1} \subset R \{1, 2, ..., n\}$, $i \notin Q_1$
- Server 1 → User: $\mathbf{R_1} = \bigoplus_{k \in Q1} X_k$
- User → Server 2: $\mathbf{Q_2} = Q_1 \cup \{i\}$
- Server 2 → User: $\mathbf{R_2} = \bigoplus_{k \in Q2} X_k$
- User derives $\mathbf{X_i} = R_1 \oplus R_2$

Analysis:

- Probabilistic-based privacy $(1/|Q_2|)$
- # of bits: 1 (× 2 servers) + inexpensive computation

Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

Database: $[X_1, X_2, \mathbf{X_3}, X_4] = [0, 1, \mathbf{0}, 1]$

- User → Server 1: $\mathbf{Q_1} \subset \{1, 2, ..., N\}$, $i \notin Q_1$
- Server 1 → User: $\mathbf{R_1} = \bigoplus_{k \in Q1} X_k$
- User → Server 2: $\mathbf{Q_2} = Q_1 \cup \{i\}$
- Server 2 → User: $\mathbf{R_2} = \bigoplus_{k \in Q2} X_k$
- User derives $\mathbf{X_i} = R_1 \oplus R_2$

- User → Server 1: $\mathbf{Q_1} = X_1, X_4$
- Server 1 → User: $\mathbf{R_1} = 1$
- User → Server 2: $\mathbf{Q_2} = X_1, \mathbf{X_3}, X_4$
- Server 2 → User: $\mathbf{R_2} = 1$
- User derives $\mathbf{X_i} = 0$

Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

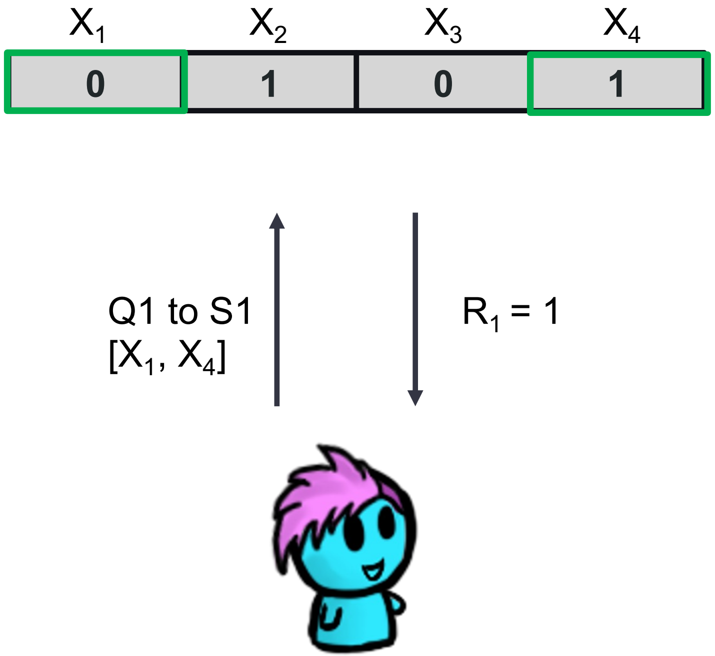
X_1	X_2	X_3	X_4
0	1	0	1

Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

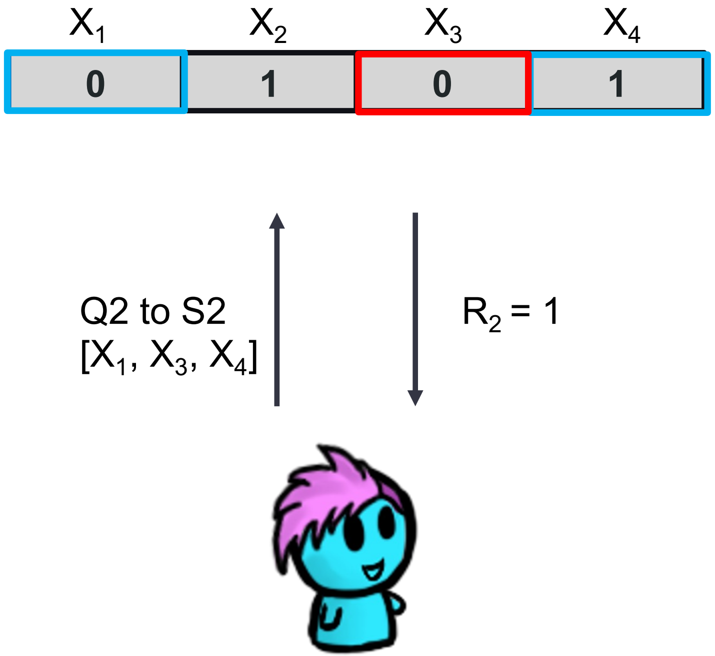


Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

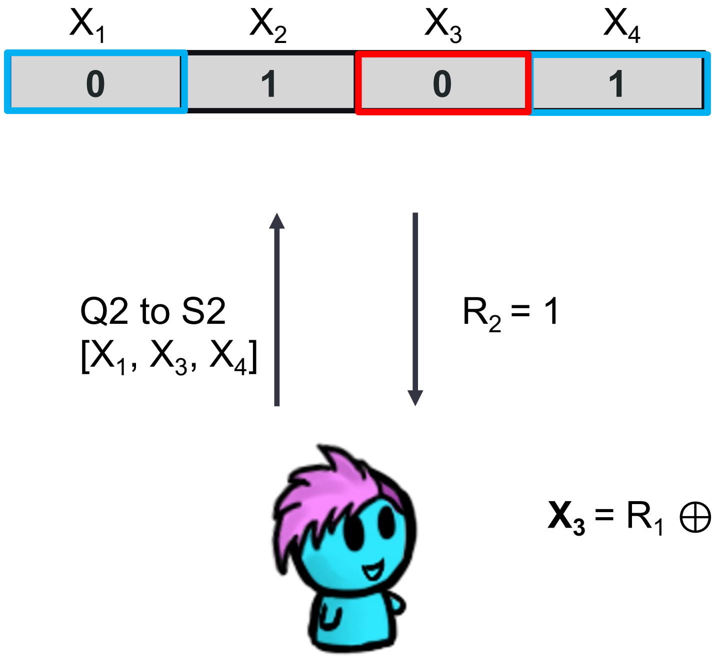


Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers



$\mathbf{X_3} = R_1 \oplus R_2 = 0$

Computational PIR

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, \ldots, X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: 1 server with limited computation power

An example CPIR protocol:

- User chooses a large random number ***m***
- User generates ***n* – 1** random quadratic residues (QR) mod ***m***: $a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$
- User generates a quadratic non-residue (QNR) mod m: $\mathbf{b_i}$
- User → Server: $a_1, a_2, \ldots, a_{i-1}, \mathbf{b_i}, a_{i+1}, \ldots, a_n$

(The server cannot distinguish between QRs and QNRs mod m, i.e., the request is just a series of random numbers: $u_1, u_2, \ldots, u_n$)

- Server → User: $\mathbf{R} = u_1^{X1} * u_2^{X2} * \ldots * u_n^{Xn}$ (The product of QRs is still a QR)
- User check: if **R** is a QR mod m, $X_i = 0$, else (**R** is a QNR mod m) $X_i = 1$

Quadratic Residues: A recap

Definition: A number ***a*** is a quadratic residue modulo ***n*** if there is an integer ***x*** such that $\mathbf{x^2 = a \bmod n}$

Quadratic Residues: A recap

Definition: A number ***a*** is a quadratic residue modulo ***n*** if there is an integer ***x*** such that $\mathbf{x^2 = a \bmod n}$

e.g., let ***n*** = 7

$0^2 = 0 \bmod 7$

$1^2 = 0 \bmod 7$

$2^2 = 4 \bmod 7$

$3^2 = 2 \bmod 7$

$4^2 = 2 \bmod 7$

$5^2 = 4 \bmod 7$

$6^2 = 1 \bmod 7$

…

Quadratic Residues: A recap

Definition: A number ***a*** is a quadratic residue modulo ***n*** if there is an integer ***x*** such that $\mathbf{x^2 = a \bmod n}$

e.g., let ***n*** = 7

$0^2 = 0 \bmod 7$

$1^2 = 0 \bmod 7$

$2^2 = 4 \bmod 7$

$3^2 = 2 \bmod 7$

$4^2 = 2 \bmod 7$

$5^2 = 4 \bmod 7$

$6^2 = 1 \bmod 7$

… (and so on)

0, 1, 2, 4 are Quadratic Residues mod **7**

Quadratic Residues: A recap

Definition: A number ***a*** is a quadratic residue modulo ***n*** if there is an integer ***x*** such that $\mathbf{x^2 = a \bmod n}$

e.g., let ***n*** = 7

$1^2 = 1 \bmod 7$

$2^2 = 4 \bmod 7$

$3^2 = 2 \bmod 7$

$4^2 = 2 \bmod 7$

$5^2 = 4 \bmod 7$

$6^2 = 1 \bmod 7$

… (and so on)

1, 2, 4 are Quadratic <u>Residues</u> mod **7**

3, 5, 6 are Quadratic <u>Non-Residues</u> mod **7**

Quadratic Residues: A recap

If we know the factorization of n, we can reduce the problem to checking residues modulo each prime factor…
Does this remind you of something?

Definition: A number ***a*** is a quadratic residue modulo ***n*** if there is an integer ***x*** such that $\mathbf{x^2 = a \bmod n}$

e.g., let ***n*** = 7

$1^2 = 1 \bmod 7$

$2^2 = 4 \bmod 7$

$3^2 = 2 \bmod 7$

$4^2 = 2 \bmod 7$

$5^2 = 4 \bmod 7$

$6^2 = 1 \bmod 7$

… (and so on)

→ **1, 2, 4** are Quadratic Residues mod **7**

→ **3, 5, 6** are Quadratic Non-Residues mod **7**

Computational PIR

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: 1 server with limited computation power

An example CPIR protocol:

- User chooses a large random number ***m***
- User generates ***n* − 1** random quadratic residues (QR) mod ***m***: $a_1, a_2, ..., a_{i-1}, a_{i+1}, ..., a_n$
- User generates a quadratic non-residue (QNR) mod m: $\mathbf{b_i}$
- User → Server: $a_1, a_2, ..., a_{i-1}, \mathbf{b_i}, a_{i+1}, ..., a_n$

(The server cannot distinguish between QRs and QNRs mod m, i.e., the request is just a series of random numbers: $u_1, u_2, ..., u_n$)

- Server → User: $\mathbf{R} = u_1^{X1} * u_2^{X2} * ... * u_n^{Xn}$ (The product of QRs is still a QR)
- User check: if **R** is a QR mod m, $X_i = 0$, else (**R** is a QNR mod m) $X_i = 1$

Computational PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: 1 server with limited computation power

Database: $[X_1, X_2, X_3, X_4] = [0, 1, 0, 1]$

- User chooses random number ***7***
- User generates ***n* − 1** random quadratic residues (QR) mod ***7***: $a_1, a_2, a_4 = 1, 2, 4$
- User generates a quadratic non-residue (QNR) mod m: b_3 = **3**
- User → Server: a_1, a_2, b_3, a_4 **1, 2, 3, 4**

(The server cannot distinguish between QRs and QNRs mod m)

- Server → User: $\mathbf{R} = \underline{1^{X1} * 2^{X2} * 3^{X3} * 4^{X4}} = \underline{0^0 * 2^1 * 3^0 * 4^1} = \underline{1 * 2 * 1 * 4} = 8$ (The product of QRs is still a QR)
- User check: **8 = 1 mod 7**. Thus, 8 is a quadratic residue modulo 7, since 1 is a QR mod 7
  Hence, X_3 = 0

Comparison of CPIR and IT-PIR

CPIR

- Possible with a single server
- Server needs to perform intensive computations
- To break it, the server needs to solve a hard problem

IT-PIR

- Only possible with >1 server
- Server may need lightweight computations only
- To break it, the server needs to collude with other servers

(Additive) Secret Sharing

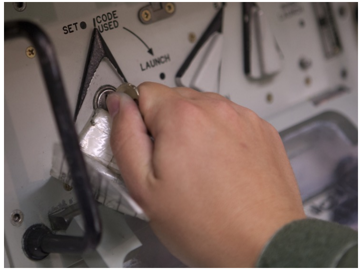
Sharing a secret across multiple parties

Could we share a piece of information between several parties, so that the individual parties learn nothing about it, but can work together to recover this information?

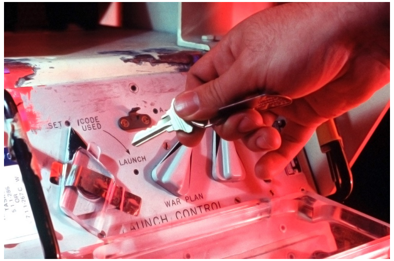
https://www.isec.tugraz.at/wp-content/uploads/teaching/mfc/secret_sharing.pdf
See the reading on “Secret Sharing” by Daniel Kales

Sharing a secret across multiple parties

Could we share a piece of information between several parties, so that the individual parties learn nothing about it, but can work together to recover this information?



e.g., nuclear launch codes



Secret Sharing Schemes

- **Generate ()**
  - Set up and return public parameters ***pp***

- **Share (*pp*, *x*, *n*, *t*)**
  - Share secret ***x*** between ***n*** players, returning a set of ***n*** shares $\{x_1, x_2, \ldots, x_n\}$. Later, ***t*** of these shares can be used to reconstruct the secret.

- **Reconstruct (*pp*, $\{x_1, x_2, \ldots, x_n\}$)**

- Recover secret **x** from **t** shares $x_1 \ldots x_t$

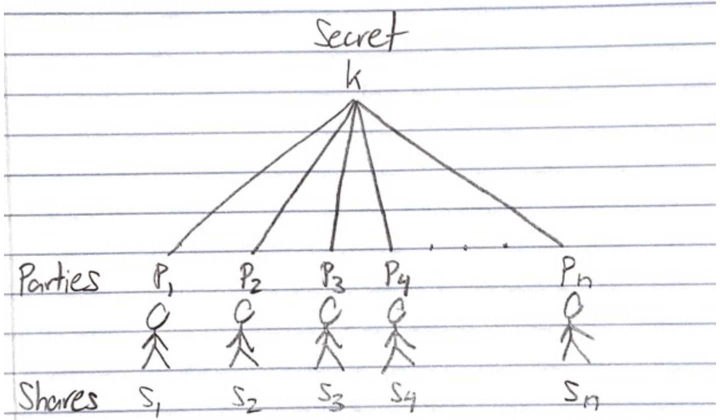


Image from: https://medium.com/data-science/how-to-share-a-secret-shamirs-secret-sharing-9a18a109a860

Secret Sharing Schemes

- **Generate ()**

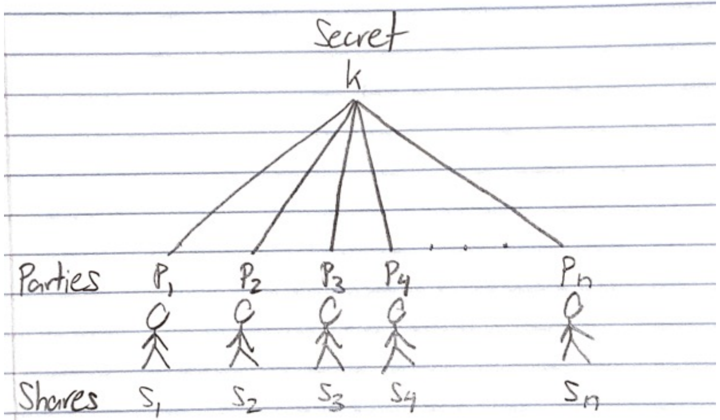
  - Set up and return public parameters ***pp***

- **Share (*pp*, *x*, *n*, *t*)**

  - Share secret ***x*** between ***n*** players, returning a set of ***n*** shares $\{x_1, x_2, ..., x_n\}$. Later, ***t*** of these shares can be used to reconstruct the secret.

- **Reconstruct (*pp*, $\{x_1, x_2, ..., x_n\}$)**

  - Recover secret **x** from **t** shares $x_1 ... x_t$



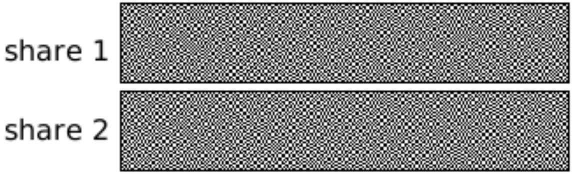


Image from: https://medium.com/data-science/how-to-share-a-secret-shamirs-secret-sharing-9a18a109a860

Secret Sharing Schemes

- **Generate ()**

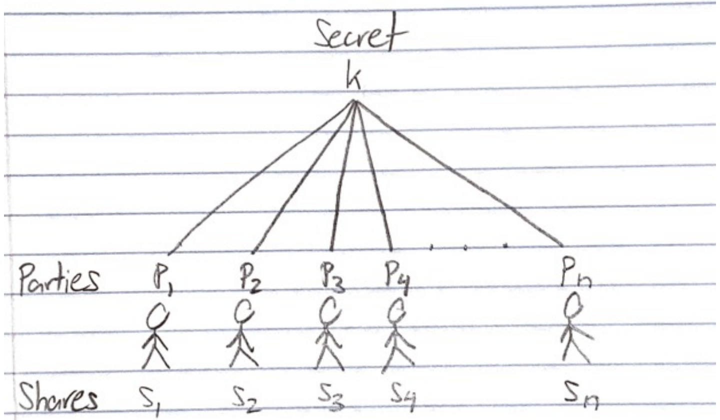
  - Set up and return public parameters ***pp***

- **Share (*pp*, *x*, *n*, *t*)**

  - Share secret ***x*** between ***n*** players, returning a set of ***n*** shares {$\mathbf{x_1}$, $\mathbf{x_2}$, …, $\mathbf{x_n}$}. Later, ***t*** of these shares can be used to reconstruct the secret.

- **Reconstruct (*pp*, {x_1, x_2, …, x_n})**

- Recover secret **x** from **t** shares $x_1 \ldots x_t$



What about these **t** shares?

Image from: https://medium.com/data-science/how-to-share-a-secret-shamirs-secret-sharing-9a18a109a860

Access thresholds

- We generally split a secret ***x*** into ***n*** shares, but for some use-cases we might want to retrieve ***x*** by combining only ***t*** shares (where ***t*** <= ***n***) .

  - These are called ***t*-out-of-*n*** secret sharing schemes!

- Today, we're going to cover a simpler ***n*-out-of-*n*** secret sharing scheme

  - That is, the specific case where ***t* = *n***

Access thresholds

- We generally split a secret ***x*** into ***n*** shares, but for some use-cases we might want to retrieve ***x*** by combining only ***t*** shares (where ***t*** <= ***n***) .

  - These are called ***t*-out-of-*n*** secret sharing schemes!

- Today, we're going to cover a simpler ***n*-out-of-*n*** secret sharing scheme

  - That is, the specific case where ***t* = *n***

Q: What about the ***t* = 1** case?

Access thresholds

- We generally split a secret ***x*** into ***n*** shares, but for some use-cases we might want to retrieve ***x*** by combining only ***t*** shares (where ***t*** <= ***n***) .

  - These are called ***t*-out-of-*n*** secret sharing schemes!

- Today, we're going to cover a simpler ***n*-out-of-*n*** secret sharing scheme

  - That is, the specific case where ***t* = *n***

Q: What about the ***t* = 1** case?

A: Well, just share ***x***…

Additive Secret Sharing ($t = n$)

- **Idea:** Split a secret $\boldsymbol{x}$ into $\boldsymbol{n}$ shares so that the sum of all shares equals $\boldsymbol{x}$
- Consider a **2-out-of-2** additive secret sharing scheme for a secret $\boldsymbol{x}$ in $\mathbb{Z}_p$
  - Choose s_1 uniformly at random
  - Set $\boldsymbol{s_2} = \boldsymbol{x} - s_1 \pmod{\boldsymbol{p}}$
  - To reconstruct $\boldsymbol{x}$, compute $\boldsymbol{s_1} + \boldsymbol{s_2} \pmod{\boldsymbol{p}}$

Additive Secret Sharing $(t = n)$

- **Idea:** Split a secret $\boldsymbol{x}$ into $\boldsymbol{n}$ shares so that the sum of all shares equals $\boldsymbol{x}$

- Consider a **2-out-of-2** additive secret sharing scheme for a secret $\boldsymbol{x}$ in $\mathbb{Z}_p$
  - Choose s_1 uniformly at random
  - Set $\boldsymbol{s_2} = \boldsymbol{x} - s_1 \pmod{\boldsymbol{p}}$
  - To reconstruct $\boldsymbol{x}$, compute $\boldsymbol{s_1} + \boldsymbol{s_2} \pmod{\boldsymbol{p}}$

Q: What if we want more shares?

Additive Secret Sharing ($t = n$)

- **Idea:** Split a secret $\boldsymbol{x}$ into $\boldsymbol{n}$ shares so that the sum of all shares equals $\boldsymbol{x}$

- Consider a **2-out-of-2** additive secret sharing scheme for a secret $\boldsymbol{x}$ in $\mathbb{Z}_p$
  - Choose s_1 uniformly at random
  - Set $\boldsymbol{s_2} = \boldsymbol{x} - s_1 \pmod{\boldsymbol{p}}$
  - To reconstruct $\boldsymbol{x}$, compute $\boldsymbol{s_1} + \boldsymbol{s_2} \pmod{\boldsymbol{p}}$

Q: What if we want more shares?

A: Easy to extend for **n-out-of-n**!

Additive Secret Sharing ($t = n$)

n-out-of-*n*

Example:

- Let $\boldsymbol{x}$ = 2025, to be shared amongst 3 parties over $\mathbb{Z}_{10^5}$

1. Randomly pick $\boldsymbol{s_1}$ = 15254 and $\boldsymbol{s_2}$ = 96214

2. $\boldsymbol{s_3} = \boldsymbol{x} - \boldsymbol{s_1} - \boldsymbol{s_2} \pmod{\boldsymbol{p}} = (2025 - 15254 - 96214) \bmod 10^5 = 90557$

3. To reconstruct $\boldsymbol{x}$, compute $\boldsymbol{s_1} + \boldsymbol{s_2} + \boldsymbol{s_3} \pmod{\boldsymbol{p}} = 15254 + 96214 + 90557 \bmod 10^5 = 2025$

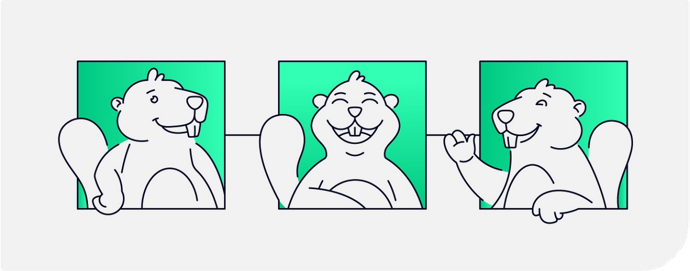
Additive Secret Sharing ($t = n$)

n-out-of-*n*

Example:

- Let x = 2025, to be shared amongst 3 parties over $\mathbb{Z}_{10^5}$

1. Randomly pick s_1 = 15254 and s_2 = 96214
2. $s_3 = x - s_1 - s_2 \pmod{p}$ = (2025 − 15254 − 96214) mod 10^5 = 90557
3. To reconstruct x, compute $s_1 + s_2 + s_3 \pmod{p}$ = 15254 + 96214 + 90557 mod 10^5 = 2025



Q: What if we want multiplicative secret sharing?

A: Trickier pre-processing steps… Make sure to check the reading (and the link below) on Beaver triples!

https://medium.com/partisia-blockchain/beavers-trick-e275e79839cc

Quick announcement (again ☺)

- **Student Course Perceptions** (https://perceptions.uwaterloo.ca/)
    - Close on Wed, July 30th
    - Did you like it? Did you hate it? Let me know!