CS459/698 Privacy, Cryptography, Network and Data Security

Multi-Party Computation, PSI, PIR

Fall 2024, Tuesday/Thursday 02:30pm-03:50pm

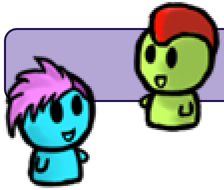
Distributed trust

The main way to use distributed trust to achieve privacy in computation is by using MPC (multiparty computation)

→ Sometime called SMC (secure multiparty computation)

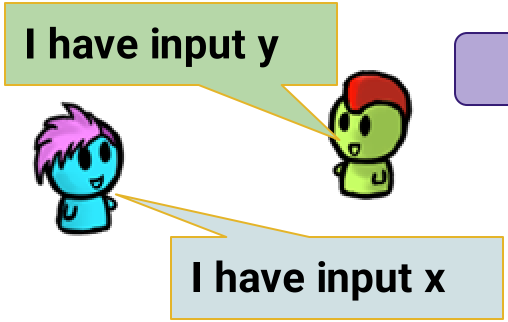
→ Let us do the computation without seeing the input data (prevent data breaches)

What is Multi-Party Computation?



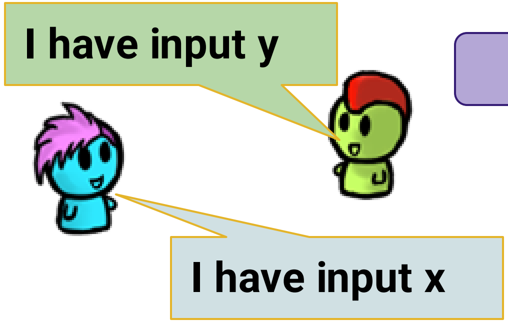
1) At least two parties

What is Multi-Party Computation?



1) At least two parties

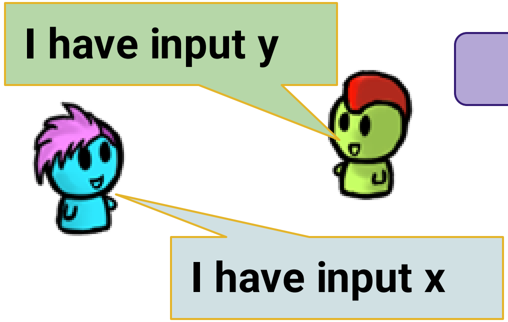
What is Multi-Party Computation?



1) At least two parties

2) Both Alice and Bob know a function f

What is Multi-Party Computation?



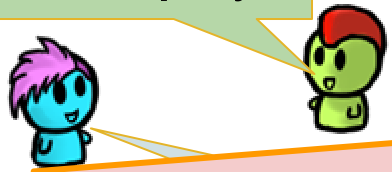
1) At least two parties

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 Comp ation?

I have input y



I have

(1) At l

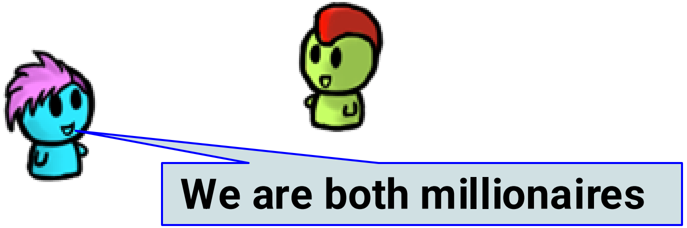
a function f

NOTE: None of the parties should learn anything about the other parties other than what is implied by the output.

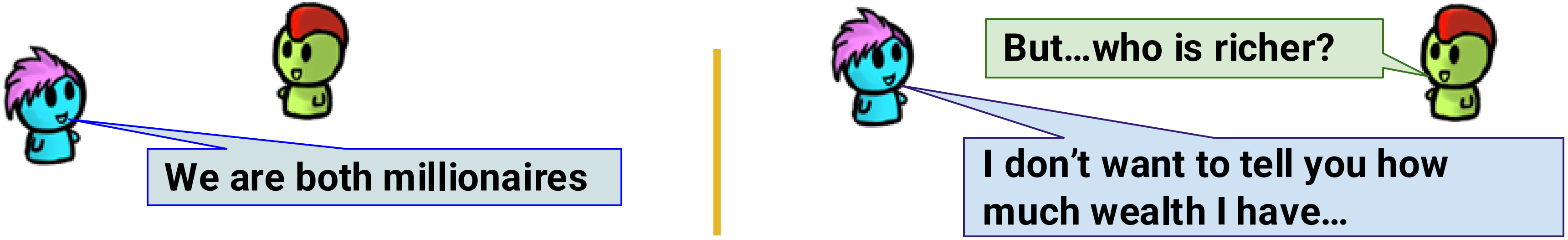
Goa rn f(but n veal anything else about x or y

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

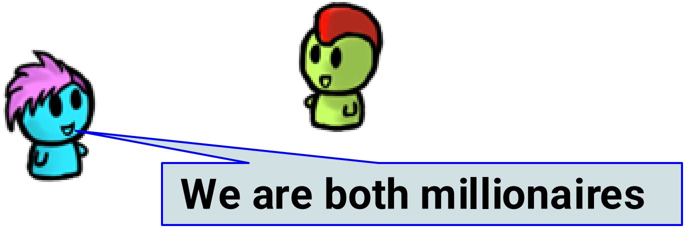
Toy Example, “The Millionaire's Problem”

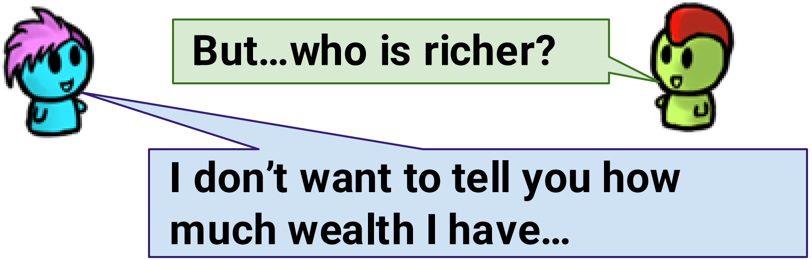


Toy Example, “The Millionaire's Problem”



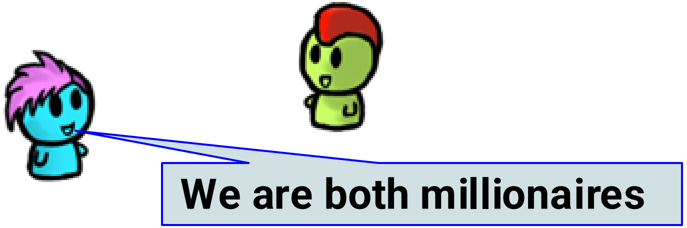
Toy Example, “The Millionaire's Problem”

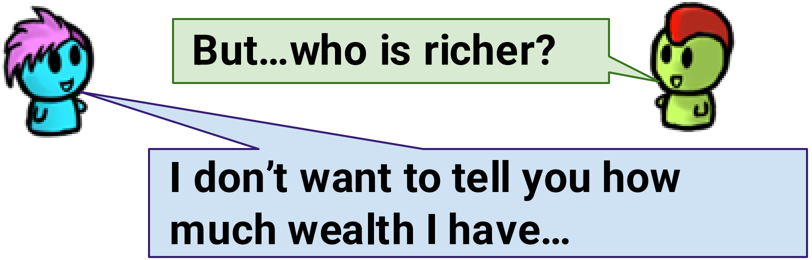




Q: how can Bob and Alice determine who is richer?

Toy Example, “The Millionaire's Problem”





Q: how can Bob and Alice determine who is richer?

A: A multi-party computation to compute f: x < y

Fun Facts:

- Andrew C. Yao, Protocols for Secure Computations Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science, 1982
- “Yao’s millionaires’ problem” (Andrew C. Yao, Turing Award 2000)

Solution

$ i millions

E_a = *Enc w/ PubA*

D_a = *Dec w/ PrivA*

$ j millions

Assume:

$1 < i, j < 10$

1. Bob picks a random N-bit integer $\boldsymbol{x}$, and computes $\boldsymbol{k} = E_a(\boldsymbol{x})$

2. Bob sends Alice the number $\boldsymbol{k - j + 1}$

3. Alice computes $y_u = D_a(k - j + u)$ for $u = [1, 2, \ldots, 10]$.

4. Alice generates random prime $\boldsymbol{p}$ of N/2-bits, and computes $z_u = y_u \pmod{p}$
   - if all z_u differ by at least 2 mod p, stop;
   - otherwise, generate another $\boldsymbol{p}$ and repeat until all z_u differ by at least 2 mod p

5. Alice sends the prime $\boldsymbol{p}$ and the following 10 numbers to Bob:
   - $z_1, z_2, \ldots, z_i$ followed by $z_{i+1} + 1, z_{i+2} + 1, \ldots, z_{10} + 1$

6. Bob looks at z_j, and decides that $i \geq j$ if $z_j = x \bmod p$, and $i < j$ otherwise. Tells Alice.

Solution Rundown

$ *i* millions

E_a = Enc w/ PubA

D_a = Dec w/ PrivA

$ *j* millions

Assume:

$1 < \boldsymbol{i}, \boldsymbol{j} < 10$

Let's use RSA as our crypto scheme!

Alice holds:

Pub_A = (e, N) = (79, 3337)

$Priv_A$ = (d) = 1019

RSA operations:

Encryption: $y = x^e \bmod N$

Decryption: $x = y^d \bmod n$

Solution Rundown

💲 *i* millions

***E*$_a$ = Enc w/ PubA**

***D*$_a$ = Dec w/ PrivA**

💲 *j* millions

Assume:

$1 < i, j < 10$

For this example, assume Alice has 5 millions ($i = 5$) and Bob has 6 millions ($j = 6$)

Step 1:

- Bob picks a random N-bit integer $x = 1234$
- Bob computes $k = E_a(x) = 1234^{79} \bmod 3337 = 901$

Step 2:

- Bob sends Alice $\underline{k - j + 1} = 901 - 6 + 1 = 896$

Solution Rundown

💲 ***i*** millions

E_a = Enc w/ PubA

D_a = Dec w/ PrivA

💲 ***j*** millions

Assume:

*1 < **i**, **j** < 10*

Step 3:

- Alice generates $Y_1 \ldots Y_{10}$, obtained by decrypting $\underline{k - j + 1}$ to $\underline{k - j + 10}$
  - This is because of our bound that tells us Alice and Bob have a number of millions between 1 and 10
  - i.e., ***u*** = [1 … 10]
- Alice can do this even without knowing ***k*** or ***j***
- ***So, what does she get?***

Solution Rundown

$ *i* millions

E_a = *Enc w/ PubA*

D_a = *Dec w/ PrivA*

$ *j* millions

Assume:

$1 < i, j < 10$

u	k - j + u	RSA decryption	y_u	
1	896	896^1019 mod 3337	1059	→ The original value Bob sent
2	897	897^1019 mod 3337	1156	
3	898	898^1019 mod 3337	2502	
4	899	.	2918	
5	900	.	385	
6	901	.	1234 (as it should be)	→ Bob's random number
7	902	.	296	
8	903	.	1596	
9	904	.	2804	
10	905	905^1019 mod 3337	1311	

Solution Rundown

$ ***i*** millions

E_a = Enc w/PubA

D_a = Dec w/ PrivA

$ ***j*** millions

Assume:

*1 < **i, j** < 10*

Step 4:

- Next, Alice generates prime number ***p*** of N/2 bits
- In this example, let's pick ***p*** = 107

- *Then, Alice generates* $\boldsymbol{Z_1 \ldots Z_{10}}$*, obtained by computing* $\boldsymbol{Y_1 \ldots Y_{10}}$ ***mod p***
- Keep in mind that ***p*** must be such that all $\mathbf{Z_u}$ differ by at least 2 units
    - This will later allow Bob to reliably determine whether i < j

Solution Rundown

$ *i* millions

E_a = **Enc w/PubA**

D_a = **Dec w/ PrivA**

$ *j* millions

Assume:

$1 < \boldsymbol{i}, \boldsymbol{j} < 10$

Step 4:

- Next, Alice generates prime number ***p*** of N/2 bits
- In this example, let's pick ***p*** = 107

- *Then, Alice generates* $\boldsymbol{Z_1 \ldots Z_{10}}$*, obtained by computing* $\boldsymbol{Y_1 \ldots Y_{10}}$ ***mod p***
- Keep in mind that ***p*** must be such that all $\mathbf{Z_u}$ differ by at least 2 units
  - This will later allow Bob to reliably determine whether i < j

- ***So, what does she get?***

Solution Rundown

$ *i* millions

E_a = Enc w/PubA

D_a = Dec w/ PrivA

$ *j* millions

Assume:

1 < i, j < 10

u	*k - j + u*	*RSA decryption*	y_u	$Z_u = (Y_u \bmod 107)$
1	896	896^1019 mod 3337	1059	96
2	897	897^1019 mod 3337	1156	86
3	898	898^1019 mod 3337	2502	41
4	899	.	2918	29
5	900	.	385	64
6	901	.	1234	57
7	902	.	296	82
8	903	.	1596	98
9	904	.	2804	22
10	905	905^1019 mod 3337	1311	27

Solution Rundown

$ *i* millions

E_a = Enc w/PubA

D_a = Dec w/ PrivA

$ *j* millions

Assume: $1 < i, j < 10$

u	k - j + u	RSA decryption	y_u	$Z_u = (Y_u \bmod 107)$
1	896	896^1019 mod 3337	1059	96
2	897	897^1019 mod 3337	1156	86
3	898	898^1019 mod 3337	2502	41
4	899	.	2918	29
5	900	.	385	64
6	901	.	1234	57
7	902	.	296	82
8	903	.	1596	98
9	904	.	2804	22
10	905	905^1019 mod 3337	1311	27

All Z_u differ by at least 2

Solution Rundown

$ i millions

E_a = *Enc w/PubA*

D_a = *Dec w/ PrivA*

$ j millions

Assume:

$1 < i, j < 10$

Step 5:

- Now, Alice sends ***p*** and 10 numbers to Bob
  - The first few numbers are Z_1, Z_2, Z_3 ... up to the value of Z_i, where ***i*** is Alice's wealth in millions

p	Z1	Z2	Z3	Z4	Z5	Z6+1	Z7+1	Z8+1	Z9+1	Z10+1
107	96	86	41	29	64	58	83	99	23	28

$ ***i*** millions

E_a = Enc w/PubA

D_a = Dec w/ PrivA

$ ***j*** millions

Assume:

*1 < **i, j** < 10*

Solution Rundown

Step 6:

- Bob now looks at the ***j***th number, where ***j*** is his wealth in millions

Bob looks at this value

p	Z1	Z2	Z3	Z4	Z5	Z6+1	Z7+1	Z8+1	Z9+1	Z10+1
107	96	86	41	29	64	58	83	99	23	28

- He then computes ***x*** mod ***p*** = ***1234*** mod ***107*** = ***57***
- Lastly, if the ***j***th number is equal to **57**, then Alice is equally wealthy (or more) than Bob (***i*** >= ***j***). Else, Bob is wealthier than Alice (***i*** < ***j***).

Solution Rundown

$ *i* millions

E_a = *Enc w/PubA*

D_a = *Dec w/ PrivA*

$ *j* millions

Assume:

$1 < i, j < 10$

Step 6:

- Bob now looks at the ***j***th number, where ***j*** is his wealth in millions

Bob looks at this value

p	Z1	Z2	Z3	Z4	Z5	Z6+1	Z7+1	Z8+1	Z9+1	Z10+1
107	96	86	41	29	64	$\boxed{58}$	83	99	23	28

- He then computes ***x*** mod ***p*** = ***1234*** mod ***107*** = ***57***
- Lastly, if the ***j***th number is equal to **57**, then Alice is equally wealthy (or more) than Bob (***i*** >= ***j***). Else, Bob is wealthier than Alice (***i*** < ***j***).

- **Step 7:** Bob tells Alice the result

Why does the Solution Work?

The intuition:

- Alice adds 1 to numbers in the series greater than her wealth (***i*** = 5);
- Bob checks to see if the one in his position in the series (***j*** = 6) has had one added to it: if it has, then he knows he must be wealthier than Alice.

I’m wealthier!

Why does the Solution Work?

The intuition:

- Alice adds 1 to numbers in the series greater than her wealth (***i*** = 5);
- Bob checks to see if the one in his position in the series (***j*** = 6) has had one added to it: if it has, then he knows he must be wealthier than Alice.

- All this has been done <u>without</u> either of them transmitting their wealth

Any issues?

Q: Can anyone identify a reason it would fail?

Any issues?

Q: Can anyone identify a reason it would fail?

Short A: Other than lies…no.

Any issues?

Q: Can anyone identify a reason it would fail?

Short A: Other than lies…no.

Long A: This technique is not cheat-proof (Bob could lie in step 7). Yao shows that such techniques can be constructed so that cheating can be limited, usually by employing extra steps.

How Scalable is this Solution?

In the real-world:

- You would need (lots of) processing power!
- **Q:** Any idea why?

How Scalable is this Solution?

In the real-world:

- You would need (lots of) processing power!
- If you wanted to cover the range 1 to 100,000,000 at a unit resolution, then Alice will be sending Bob a table of 100,000,000 numbers!
- This table would be on the order of a GB. You could handle it, but processing and storage implications are non-trivial.

New advances on MPC attempt to tackle these issues in clever ways…

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*

MPC Server Model

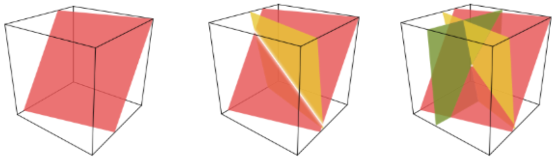
$A_1\ B_1\ C_1\ D_1\ E_1$

A few servers help many clients compute over their private inputs

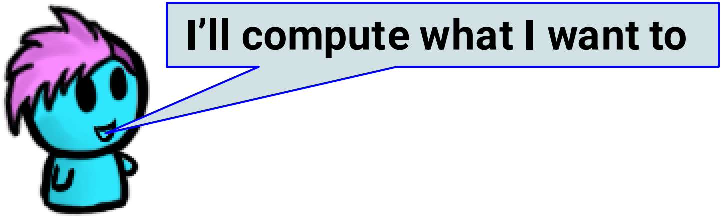
$A_2\ B_2\ C_2\ D_2\ E_2$

$A_3\ B_3\ C_3\ D_3\ E_3$

Secret Sharing



“Types” of MPC: Functionality

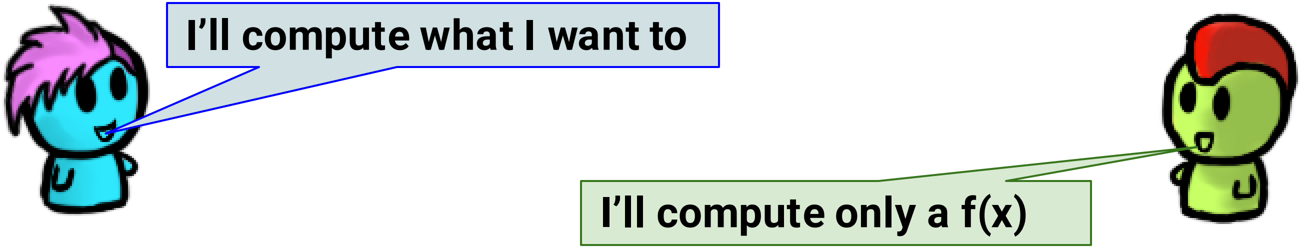


Generic

Generic functions:

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

“Types” of MPC: Functionality



Generic

Specific

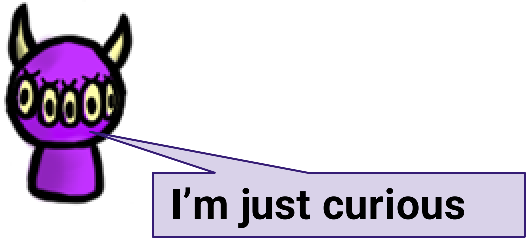
<u>Generic functions:</u>

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

<u>Specific functions</u>:

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

“Types” of MPC: Security

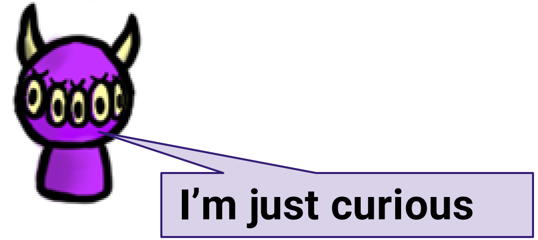


Passive

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

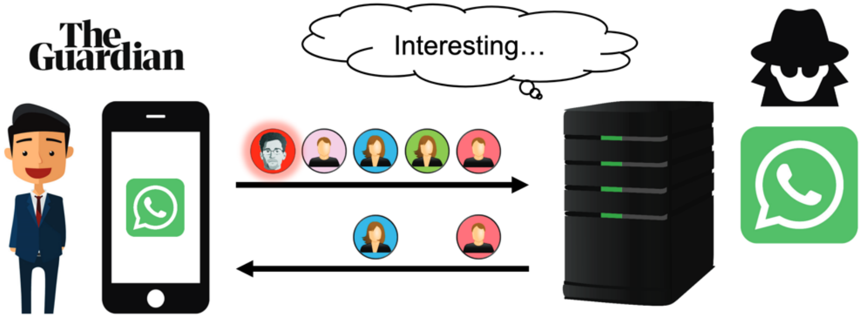
- Any protocol secure against passive adversaries can be turned into a protocol secure actives adversaries
  - E.g., by adding protocol steps proving the correct computation of each message:
    - Cryptographic commitments: can we detect if a participant deviates from the protocol ?
    - Validations: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

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

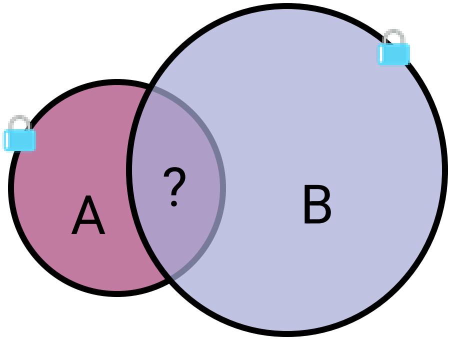
An MPC Application for a specific function: Private Set Intersection (PSI)

Private Set Intersection (PSI)

- 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?

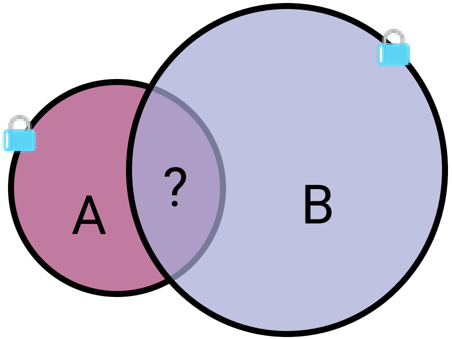


Private Set Intersections



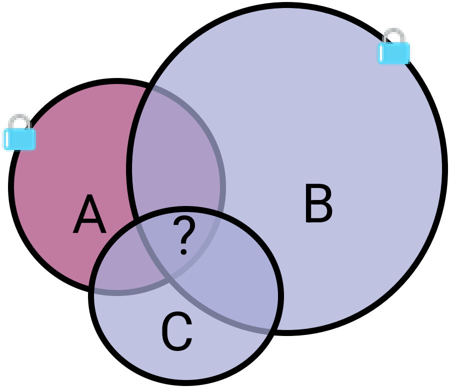
2-Party, One-Way PSI

$A \longrightarrow B$



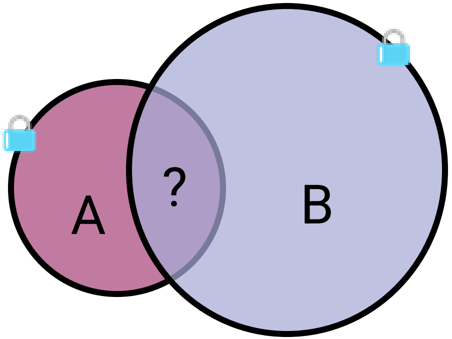
2-Party, Two-Way PSI

$A \longleftrightarrow B$



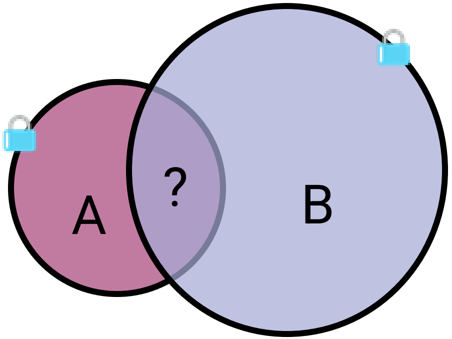
n-Party PSI

Private Set Intersections



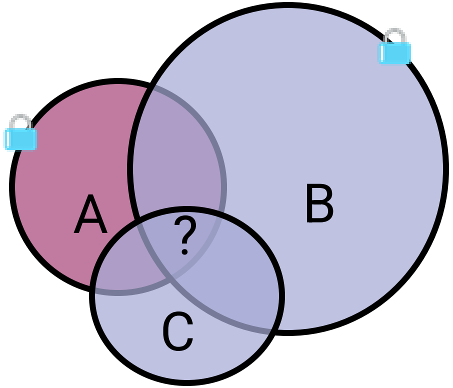
2-Party, One-Way PSI

A $\longrightarrow$ B



2-Party, Two-Way PSI

A $\longleftrightarrow$ B



n-Party PSI

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

Strawman Protocol for PSI

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

- **Protocol for comparison ($\boldsymbol{x} =? \boldsymbol{y}$)**
  - Alice → Bob: $E_A(\boldsymbol{x})$
  - Bob: Choose random $\boldsymbol{r}$ and compute $\boldsymbol{c} = (E_A(\boldsymbol{x}) * E_A(-\boldsymbol{y}))^r$
    - Add encrypted value of $\boldsymbol{x}$ with encrypted value of $-\boldsymbol{y}$ (the negative of $\boldsymbol{y}$) and raise the result to the power of $\boldsymbol{r}$.
  - Bob → Alice: $\boldsymbol{c}$
  - Alice: Output $\boldsymbol{x} = \boldsymbol{y}$, 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***
  - Alice: Output $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

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

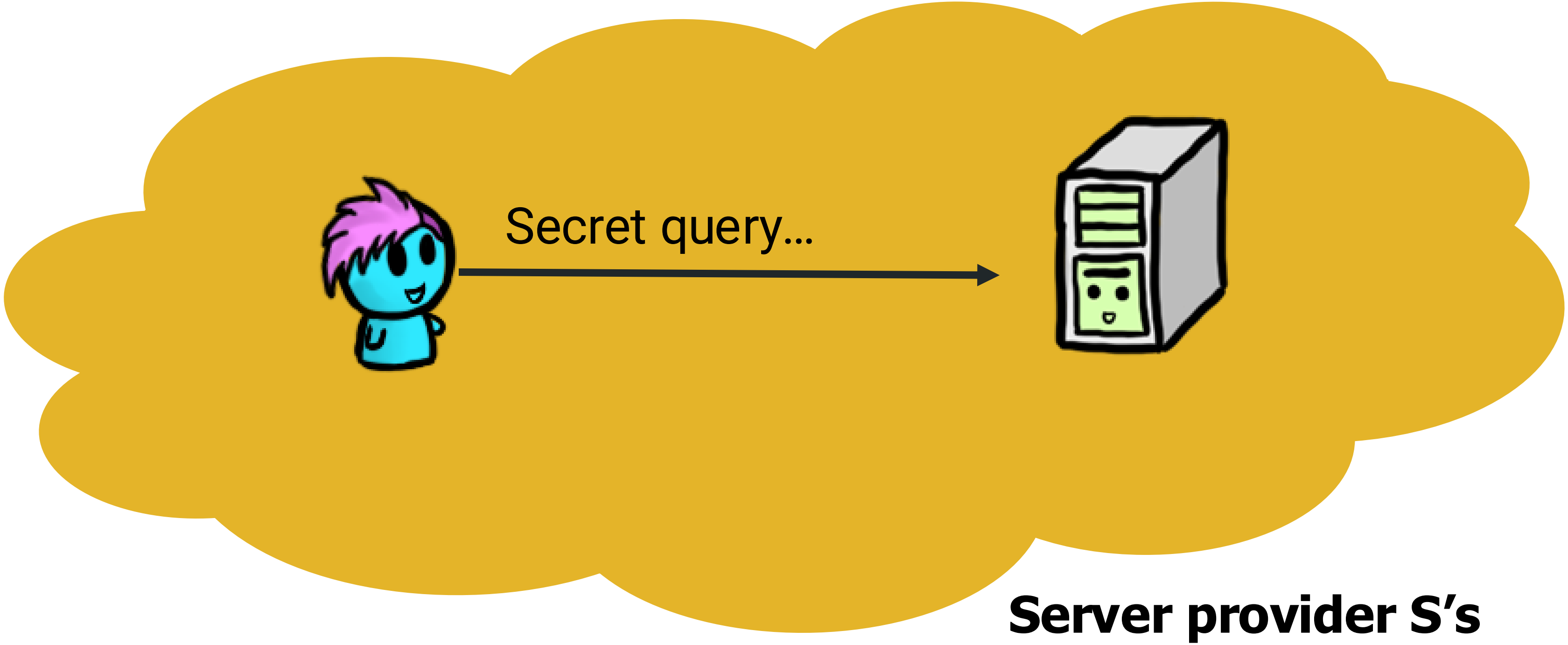
- Alice permutes her set ***X***, Bob permutes his set ***Y***. Then:
  - For each $\boldsymbol{x} \in X$
    - For each $\boldsymbol{y} \in Y$
      - Compute ***x* =? *y***

- **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***
  - Alice: Output $\boldsymbol{x} = \boldsymbol{y}$, if $D_A(\boldsymbol{c}) = 0$, else $\boldsymbol{x} \neq \boldsymbol{y}$

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

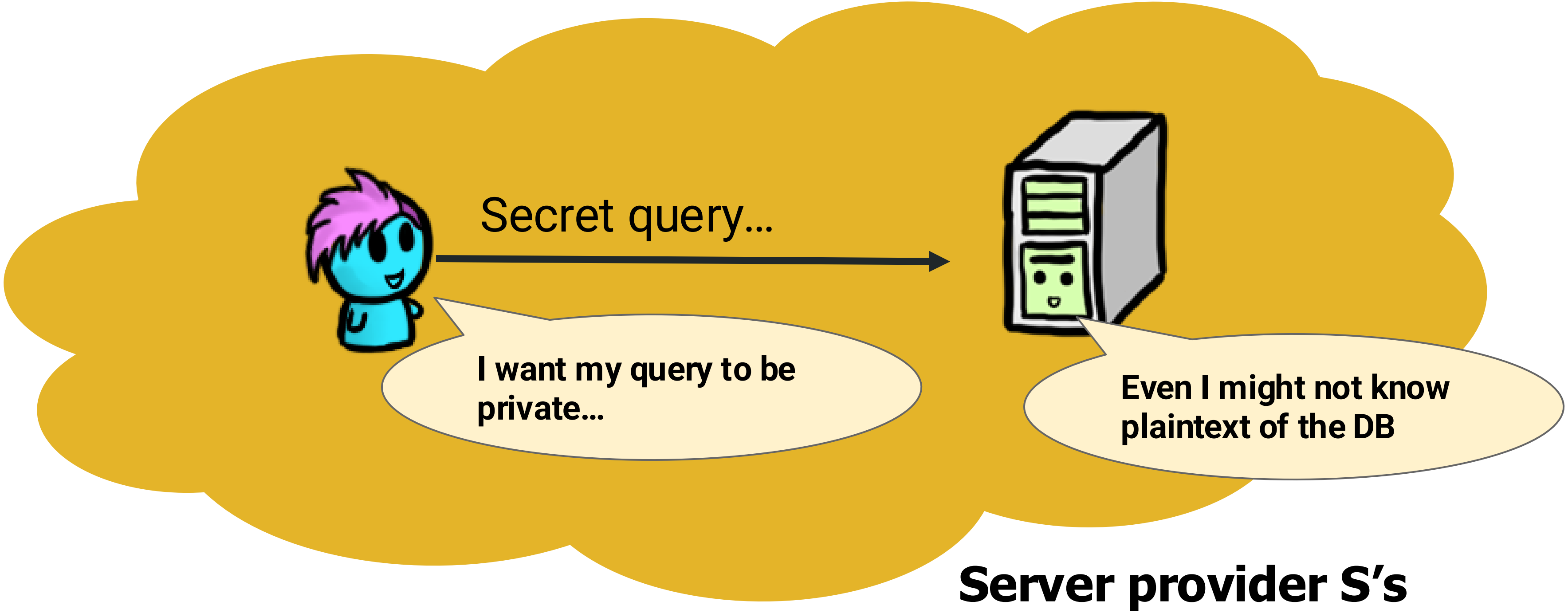
Private Information Retrieval (PIR)

Can we privately query a database?



Server provider S's cloud

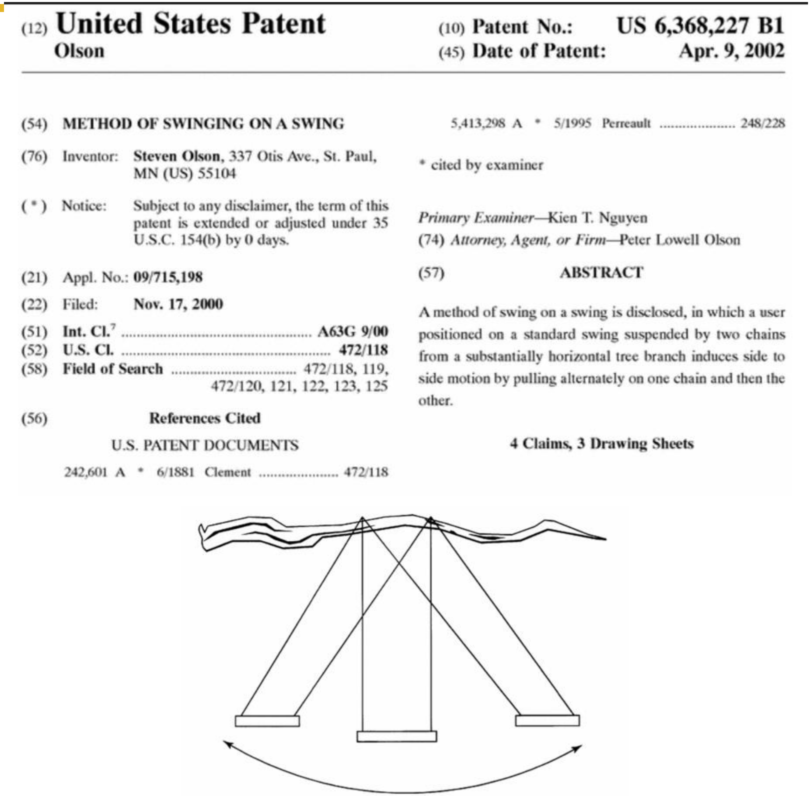
Ideally…



Server provider S’s cloud

Motivating Example (0)

- You want to look something up in an online database
  - For example, a database of patents
- You want to keep private the information being retrieved
  - For example, the patent number (6368227) you're looking up



(12) **United States Patent**
Olson

(10) **Patent No.:** **US 6,368,227 B1**
(45) **Date of Patent:** **Apr. 9, 2002**

(54) **METHOD OF SWINGING ON A SWING**

(76) Inventor: **Steven Olson**, 337 Otis Ave., St. Paul, MN (US) 55104

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/715,198**

(22) Filed: **Nov. 17, 2000**

(51) **Int. Cl.**[7] **A63G 9/00**
(52) **U.S. Cl.** **472/118**
(58) **Field of Search** 472/118, 119, 472/120, 121, 122, 123, 125

(56) **References Cited**

U.S. PATENT DOCUMENTS

242,601 A * 6/1881 Clement 472/118
5,413,298 A * 5/1995 Perreault 248/228

* cited by examiner

Primary Examiner—Kien T. Nguyen
(74) *Attorney, Agent, or Firm*—Peter Lowell Olson

(57) **ABSTRACT**

A method of swing on a swing is disclosed, in which a user positioned on a standard swing suspended by two chains from a substantially horizontal tree branch induces side to side motion by pulling alternately on one chain and then the other.

4 Claims, 3 Drawing Sheets

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 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 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. 12 Angry Men
  - ...
- 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)

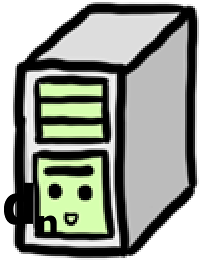
- Netflix stores movies in a database
  - 1. The Shawshank Redemption
  - 2. The Godfather
  - 3. The Dark Knight
  - 4. 12 Angry Men
  - ...
- You request movies by index, say 1, 4, 2, ...
- Netflix caches your selection and gradually builds a profile on your movie preferences
- The server should be queried **without learning** the item of interest!

PIR

Carol has index i

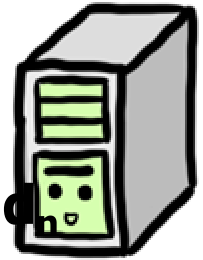
PIR

Carol has index i



Server has DB $d_1,\ldots,d_n$

PIR



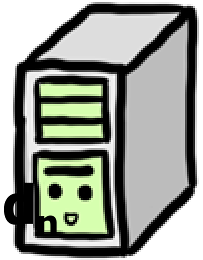
Server has DB $d_1,\ldots,d_n$

Carol has index i

Goal 1: Correctness - Client learns d_i

PIR

Carol has index i



Server has DB $d_1, \ldots, d_n$

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 $\{X_1, X_2, \ldots, X_n\}$
- 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 $\{X_1, X_2, \ldots, X_n\}$
- 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: $\mathbf{Q_1} \subset R \{1, 2, \ldots, n\}, \quad i \notin Q_1$
- 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 $\{X_1, X_2, \ldots, X_n\}$
- User: wishes to retrieve X_i AND keep i private

Assumption: multiple (≥ 2) non-cooperating servers

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

- User → Server 1: $\mathbf{Q_1} \subset \{1, 2, \ldots, N\}$, $i \notin Q_1$
- 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: $\mathbf{Q_2} = X_1, X_3, X_4$
- Server 2 → User: $\mathbf{R_2} = 1$
- User derives $\mathbf{X_3} = 0$

Information-Theoretic PIR (Example)

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: multiple (≥ 2) non-cooperating servers

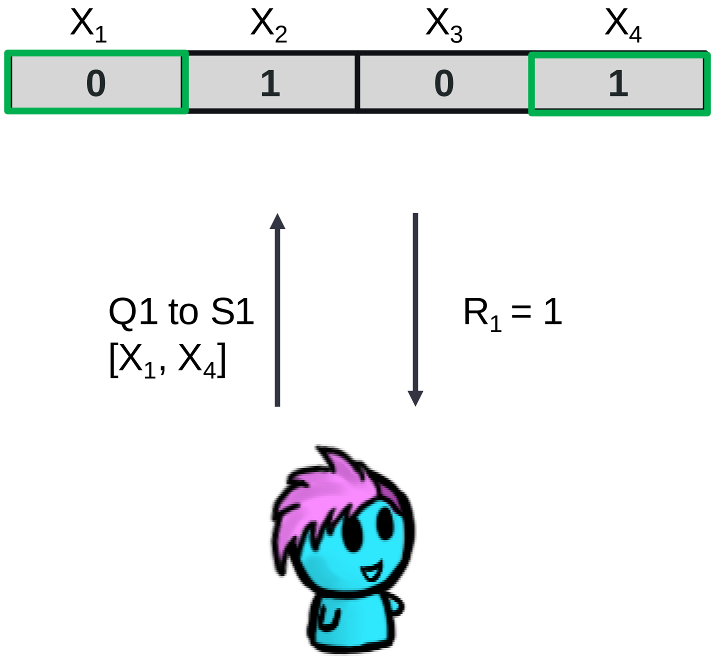
X_1	X_2	X_3	X_4
0	1	0	1

Information-Theoretic PIR (Example)

Formal model:

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

Assumption: multiple (≥ 2) non-cooperating servers

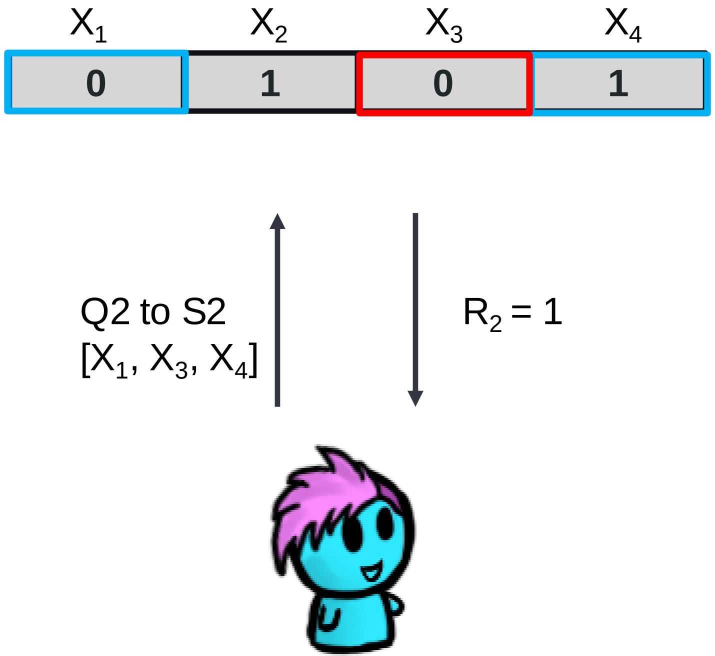


Information-Theoretic PIR (Example)

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, \ldots, X_n\}$
- 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 $\{X_1, X_2, \ldots, X_n\}$
- 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**

Q2 to S2
$[X_1, X_3, X_4]$

$R_2 = 1$

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

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 $\mathbf{x^2 = a \bmod n}$

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 $n = 7$

$0^2 = 0 \bmod 7$

$1^2 = 0 \bmod 7$

$2^2 = 4 \bmod 7$

$3^2 = 2 \bmod 7$

$4^2 = 2 \bmod 7$

$5^2 = 4 \bmod 7$

$6^2 = 1 \bmod 7$

… (and so on)

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

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 $\{X_1, X_2, \ldots, X_n\}$
- 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**: $a_1, a_2, a_4 = 0, 2, 4$
- User generates a quadratic non-residue (QNR) mod m: b_3 = **3**
- User → Server: a_1, a_2, b_3, a_4 **0, 2, 3, 4**

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

- Server → User: $\mathbf{R} = \underline{0^{X1} * 2^{X2} * 3^{X3} * 4^{X4}} = \underline{0^0 * 2^1 * 3^0 * 4^1} = \underline{1 * 2 * 1 * 4} = 8$ (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

Quick announcements

- Student Course Perceptions (https://perceptions.uwaterloo.ca/)
    - Open on Wednesday, Nov 20th
    - Close on Tuesday, Dec 3rd
    - Did you like it? Did you hate it? Let me know!