



# Security and Privacy

Secure Multiparty Computation & Privacy

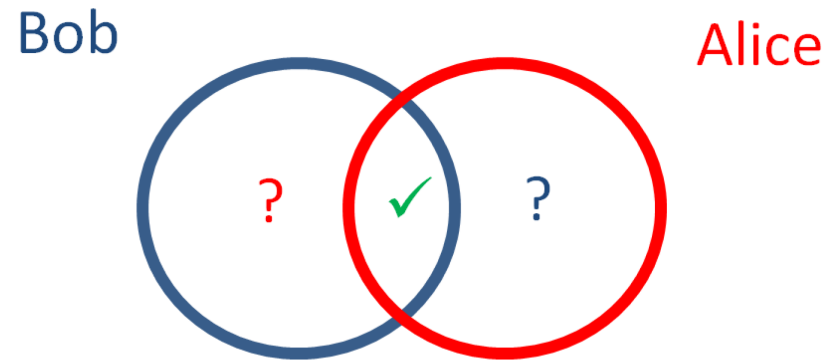
# Multi-Party Computation

- **Distributed computing** considers the scenario where a number of **distinct**, yet **connected, computing devices** (or parties) wish to carry out a **joint computation** of some function.
  - Distributed computing classically deals with **questions of computing** under the threat of **machine crashes** and or **inadvertent faults**.
- **The aim of secure multiparty computation** is to enable parties to carry out such distributed computing tasks in a **secure manner**.
  - Secure multiparty computation is concerned with the possibility of **malicious behaviour** by some adversarial entity

# Overview of SMC Primitives

- Oblivious transfer (OT)
  - Share information without knowing what information was shared
- Bit commitment (BC)
  - How to flip a coin over the telephone?
- Zero Knowledge (ZK)
  - Prove a statement without disclosure
- Private set intersection (PSI)
  - Determine shared cellphone contacts without knowing contacts that are not shared

# Problem: Private Set Intersection



- Two Homeland Security agencies want to cooperate to find potential terrorists
  - Intersection of black-lists
- Two persons want to determine their common contacts (e.g., smartphone)
  - Intersection of contact lists

# Private Set Intersection - Applications

- PSI is a very natural problem
- Private contact discovery:
  - When a new user registers to a service it is often essential to identify current registered users who are also contacts of the new user.
  - This operation can be done by simply revealing the user's contact list to the service, but can also be done in a privacy preserving manner by running a PSI protocol between the user's contact list and the registered users of the service.



# Private Set Intersection - Applications

- Measuring ad conversion rates
  - Is done by comparing the list of people who have seen an **ad** with those who have **completed a transaction**
  - These lists are held by the advertiser (say, Google or Facebook), and by merchants, respectively



# Solution: Private Set Intersection

There are several solutions available:

- a **naive hashing solutions** where elements are hashed and compared
- the **Diffie-Hellman-based PSI protocol**
- the **Server-aided protocol**
- the **OT-based PSI protocol**

# Solution: Private Set Intersection

- Assumption:
  - We only consider semi-honest (passive) adversaries
- Why discuss only semi-honest?
  - There are PSI protocols secure against malicious adversaries [FNP04, JL09, HN10, CKT10, FHNP13]
  - These protocols are much less efficient
  - None of them was implemented

# Private Set Intersection – Scenario 1



**Client**



**Server**

**Input:**

**$X = x_1 \dots x_n$**

**$Y = y_1 \dots y_n$**

**Output:**

**$X \cap Y$  only**

**nothing**

**How to solve this problem?**

# A naïve PSI protocol

- A naïve solution:
  - Have **A** and **B** agree on a “cryptographic hash function”  $H()$
  - **B** sends to **A**:  $H(y_1), \dots, H(y_n)$
  - **A** compares to  $H(x_1), \dots, H(x_n)$  and finds the intersection
- **Problem:** Does not protect **B**’s privacy if inputs do not have considerable entropy
  - E.g., B may have all contacts A has and nothing else.

# Private Set Intersection – Scenario 2



**Client**



**Server**

**Input:**

**$X = x_1 \dots x_n$**

**$Y = y_1 \dots y_n$**

**Output:**

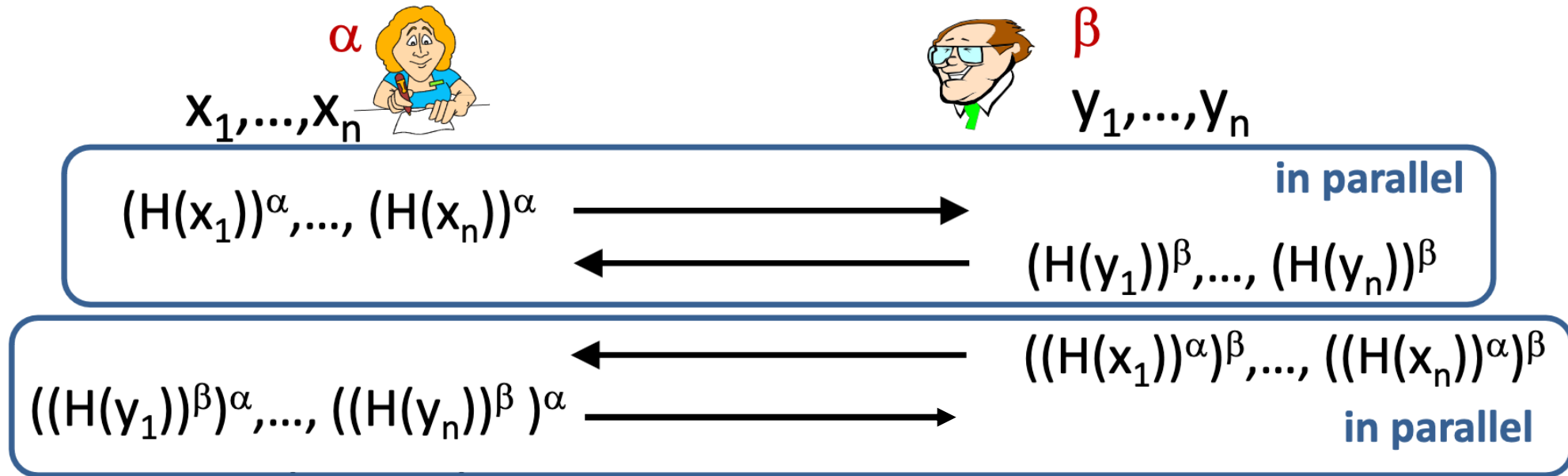
**$X \cap Y$  only**

**$X \cap Y$  only**

**How to solve this problem?**

# PSI based on Diffie-Hellman

Fact:  $(X^y)^z = X^{y \cdot z} = X^{z \cdot y} = (X^z)^y$



# PSI based on Diffie-Hellman - Example

$$X = (1, 2, 3, 4, 5), \quad \alpha = 3, \quad H(x) = 2x$$

$$Y = (1, 2, 6, 7, 8), \quad \beta = 2, \quad H(y) = 2y$$

$$((1*2)^3, (2*2)^3, (3*2)^3, (4*2)^3, (5*2)^3)$$

$$((1*2)^2, (2*2)^2, (6*2)^2, (7*2)^2, (8*2)^2)$$

$$((1*2)^{2*3}, (2*2)^{2*3}, (6*2)^{2*3}, (7*2)^{2*3}, (8*2)^{2*3})$$

$$((1*2)^{3*2}, (2*2)^{3*2}, (3*2)^{3*2}, (4*2)^{3*2}, (5*2)^3)$$

$$((1*2)^{3*2}, (2*2)^{3*2}, (3*2)^{3*2}, (4*2)^{3*2}, (5*2)^3)$$

$$((1*2)^{2*3}, (2*2)^{2*3}, (6*2)^{2*3}, (7*2)^{2*3}, (8*2)^{2*3})$$

Comparison



$$((1*2)^{3*2}, (2*2)^{3*2})$$

$$(1, 2)$$

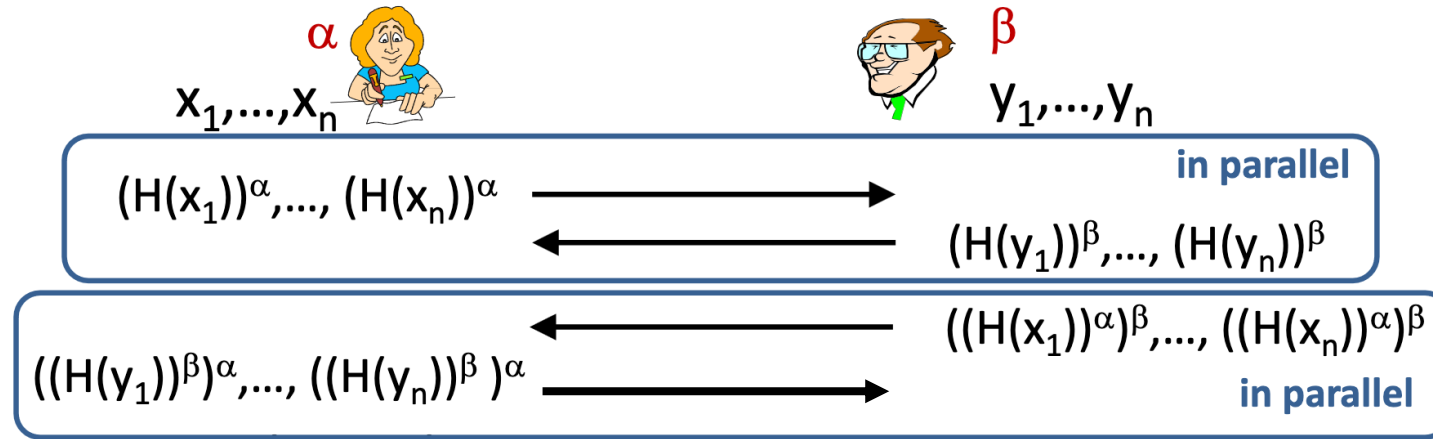
$$((1*2)^{3*2}, (2*2)^{3*2})$$

$$(1, 2)$$

Comparison



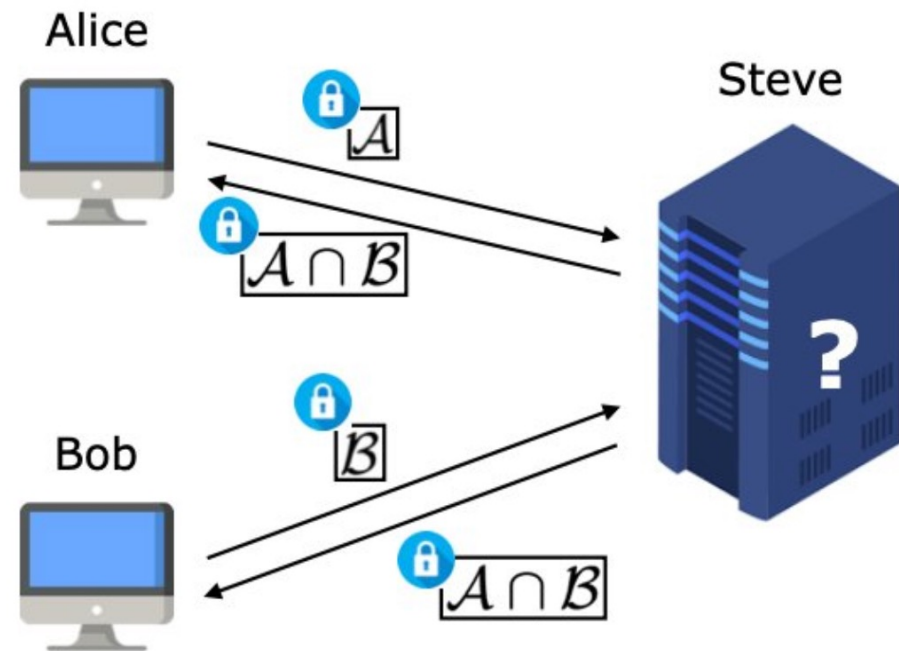
# PSI based on Diffie-Hellman



- **Implementation:** very simple;
- **Limitations:** This protocol clearly assumes a **high degree of trust between A and B**,
  - **A can falsify a match with B by sending B back the message he received in the third step of the protocol.**
  - B (or A) can also find out if he has a match without revealing it to A by **sending garbage in the last step of the protocol.**
  - Does not protect **A** and **B**'s privacy if inputs do not have considerable entropy (similar to the previous solution)

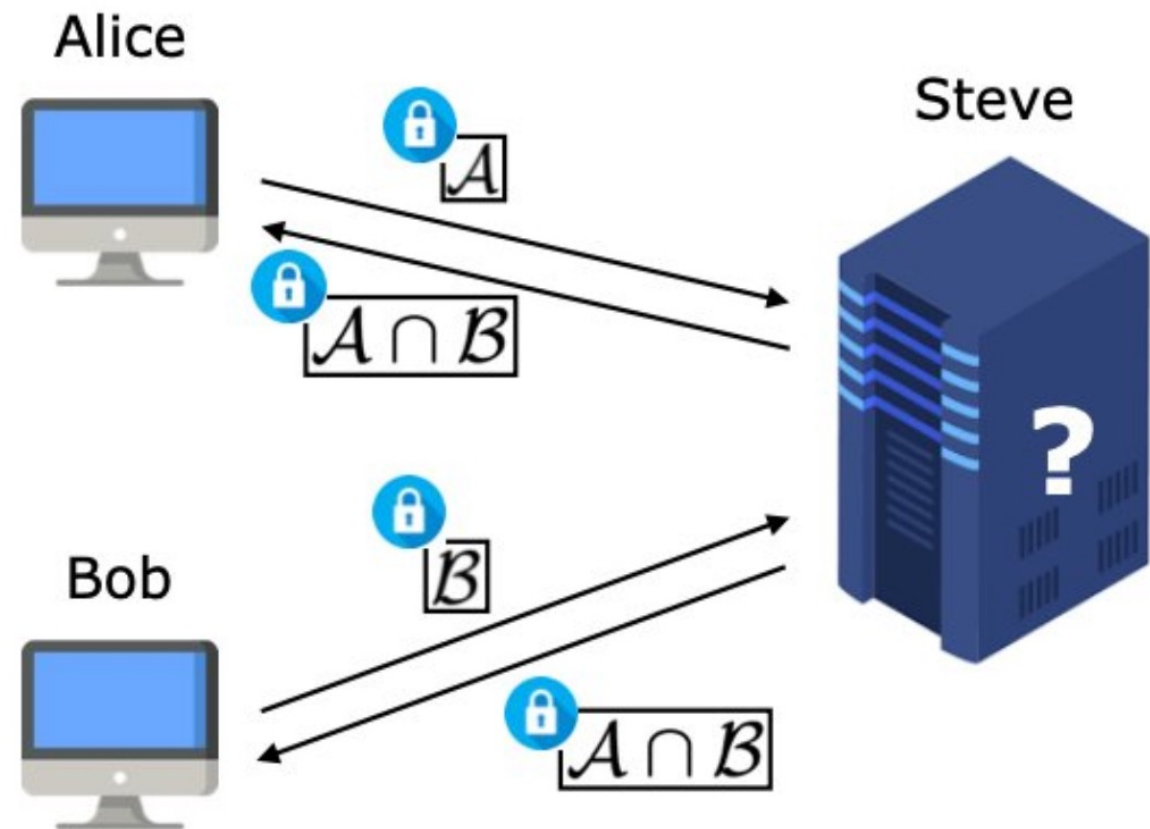
# Server-aided (Third Party-based) protocol for PSI

- There is an additional party, called **Steve**
- Alice and Bob trust Steve
- Privacy requirement: **Steve should not learn information about the items of Alice and Bob.**



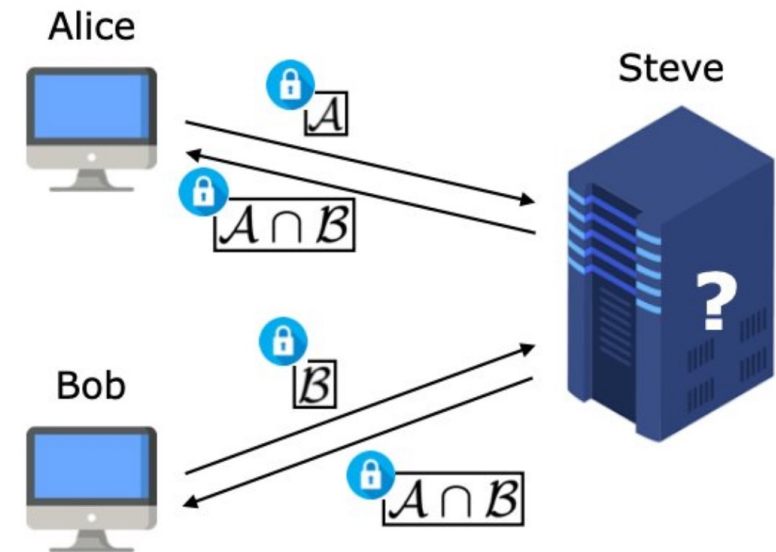
# Server-aided protocol for PSI

- We assume **no collusion**
  - **Alice and Steve do not collude** in order to fool Bob (i.e., make Bob conclude the intersection has more/less items than it really has)
  - **Alice and Steve do not collude** in order to break Bob's privacy (i.e., revealing information about Bob's items).
  - Bob and Steve do not collude as well



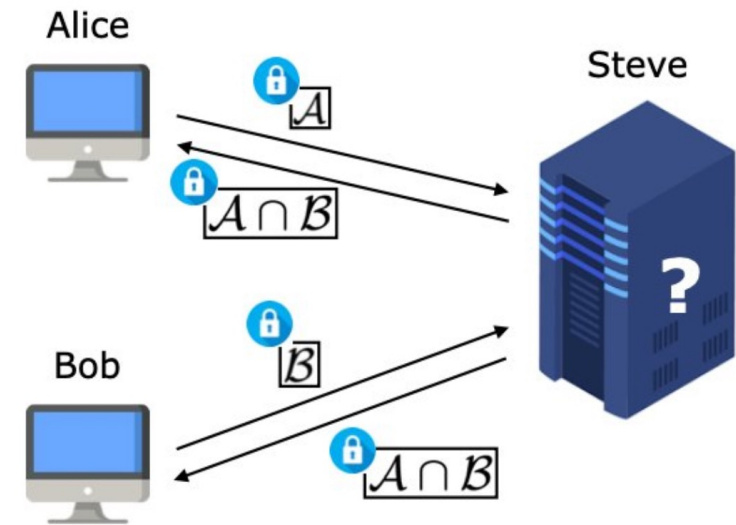
# Server-aided protocol for PSI

- Alice has  $\mathbf{A} = \mathbf{a}_1, \dots, \mathbf{a}_n$
- Bob has  $\mathbf{B} = \mathbf{b}_1, \dots, \mathbf{b}_m$
- Let  $\mathbf{E}$  be an encryption function and  $\mathbf{D}$  be the reverse decryption function
- Alice, Bob, and Steve agree on  $\mathbf{E}$
- Alice and Bob agree on key  $\mathbf{K}$  to be used by  $\mathbf{E}$  and  $\mathbf{D}$



# Server-aided protocol for PSI

- For each  $i = 1, \dots, n$ , Alice calculates  $a'_i = E(K, a_i)$  and send them to Steve
- For each  $i = 1, \dots, m$ , Bob calculates  $b'_i = E(K, b_i)$  and send them to Steve
- Steve calculates  $A' \cap B' = (c'_1, \dots, c'_p)$  and sends the results to Alice and Bob
- For each  $i = 1, \dots, p$ , Alice and Bob calculate  $c_i = D(K, c'_i)$



# Server-aided protocol for PSI

- Is this protocol Private?
  - The only information leaked to Steve is the **number of items Alice and Bob originally have** and the **size of the intersection,  $|A' \cap B'|$**
  - **Steve does not learn anything about the *value* of the items** (neither those in the intersection nor the rest) because all he sees are outputs of  $E$ , and these look random to him, since he doesn't know  $K$
  - Alice and Bob learn only what items are in the intersection and nothing more

# Server-aided protocol for PSI

- Is the protocol Secure?
  - It depends, if we assume that **Steve follows the protocol faithfully** then yes.
  - If we suspect **Steve tries to cheat** (i.e., fool Alice and Bob to conclude with a wrong intersection) then no

# To remember: Oblivious Transfer (OT)

- A sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred.



**Oblivious:** Alice doesn't learn which secret Bob obtains  
**Transfer:** Bob learns one of Alice's secrets

# To Remember: OT Semi-Honest

## 1-out-of-2 Oblivious Transfer (OT)

- Inputs

- Sender has two messages  $m_0$  and  $m_1$
- Receiver has a single bit  $\sigma \in \{0,1\}$

- Outputs

- Sender receives nothing
- Receiver obtains  $m_\sigma$  and learns nothing of  $m_{1-\sigma}$

# To remember - Semi-Honest OT Protocol

## Protocol for Oblivious Transfer

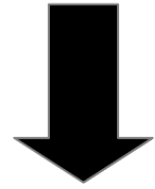
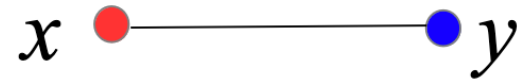
- Receiver (with input  $\sigma$ ):
  1. Receiver chooses one key-pair  $(pk, sk)$  and one public-key  $pk'$  (obliviously of secret-key).
  2. Receiver sets  $pk_\sigma = pk$ ,  $pk_{1-\sigma} = pk'$
  3. Note: receiver can decrypt for  $pk_\sigma$  but not for  $pk_{1-\sigma}$
  4. Receiver sends  $pk_0, pk_1$  to sender
- Sender (with input  $m_0, m_1$ ):
  1. Sends receiver  $c_0 = E_{pk_0}(m_0)$ ,  $c_1 = E_{pk_1}(m_1)$
- Receiver:
  1. Decrypts  $c_\sigma$  using  $sk$  and obtains  $m_\sigma$ .

# To Remember - OT Generalization

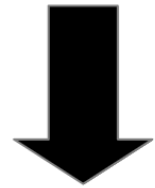
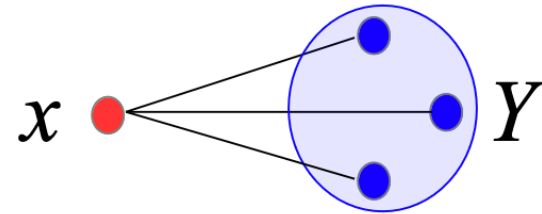
- Can define 1-out-of-k oblivious transfer
- Protocol remains the same:
  - Choose  $k-1$  public keys for which the secret key is unknown
  - Choose 1 public-key and secret-key pair

# PSI based on Oblivious transfer (OT)

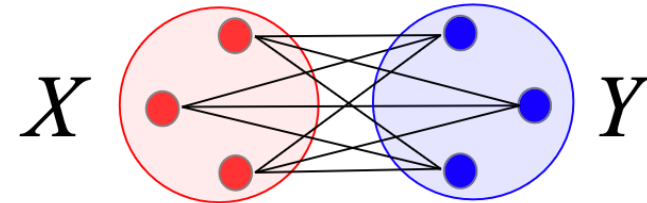
Private Equality Test:



Private Set Inclusion:



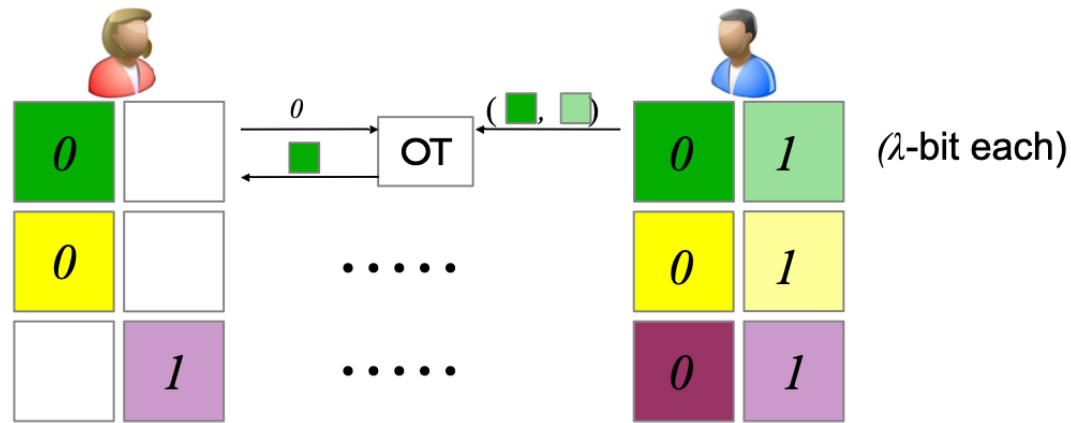
Private Set Intersection:



# PSI based on OT (Equality Test)

- **Input:** Alice has  $x$ , Bob has  $y$ . **Output:**  $x \stackrel{?}{=} y$   $x$  ● ————— ●  $y$

- **Example:**  $x = 001$ ,  $y = 011$



- Bob sends  $\lambda$ -bit mask  $0 \oplus 1 \oplus 1$  to Alice

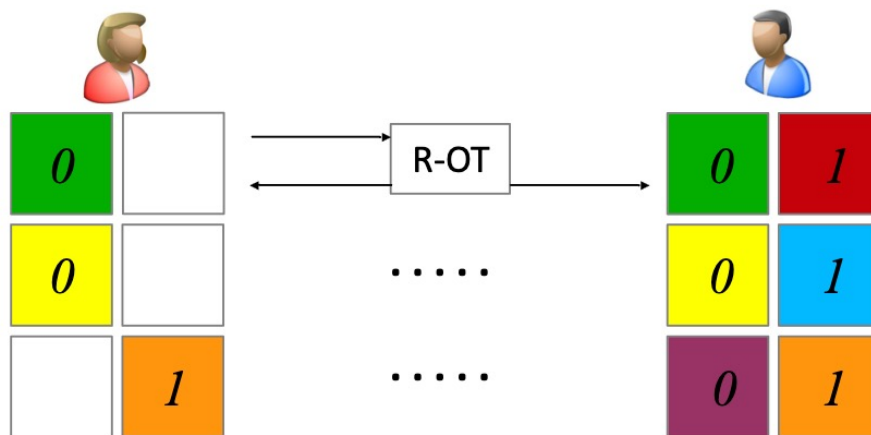
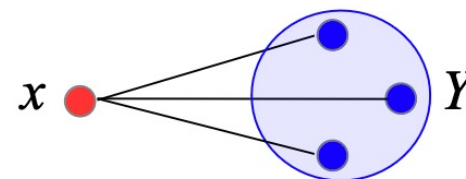
- Alice computes  $0 \oplus 0 \oplus 1$  and compares

# PSI based on OT (Set Inclusion)

**Input:** Alice has  $x$ , Bob has  $Y = \{y_1, \dots, y_n\}$ . **Output:**  $x \in Y$ ?

Run the Private Equality Tests as usual

- Inputs to and outputs of OT are the same  
=> same number of OTs, but on short strings
- Difference: Bob sends  $n\lambda$  bits to Alice



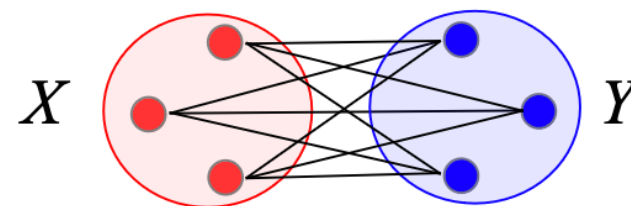
Break correlations using Random Oracle

$$\begin{aligned}
 y_1=011: & \quad H( \text{green } 0 \oplus \text{blue } 1 \oplus \text{orange } 1 ) \\
 y_2=101: & \quad H( \text{red } 1 \oplus \text{yellow } 0 \oplus \text{orange } 1 ) \\
 & \quad \vdots \\
 y_n=110: & \quad H( \text{red } 1 \oplus \text{blue } 1 \oplus \text{purple } 0 )
 \end{aligned}$$

# PSI based on OT (Set Intersection)

**Input:** Alice has  $X = \{x_1, \dots, x_n\}$ , Bob has  $Y = \{y_1, \dots, y_n\}$ .

**Output:**  $X \cap Y$ .



Run  $n$  Private Set Inclusions in parallel

# SMC Application to Private Data Mining

- The setting in distributed data mining:
  - Data is **distributed** at different sites
  - These sites may be third parties (e.g., hospitals, government bodies) or may be the individual him or herself
- The aim:
  - Compute the data mining algorithm on the data so that **nothing but the output is learned**
  - That is, carry out a **secure computation**

# Privacy and Secure Computation

- A two-stage process in secure computation:
  - Decide what function/algorithm should be computed – an issue of **privacy**
  - Apply secure computation techniques to compute it securely – **security**
- As we have mentioned, **secure computation** only deals with the **process of computing the function**
  - It does not ask **whether or not the function should be computed**
- We assume that the function can be computed without privacy loss

# Personalized Newspapers

- The aim:
  - Present a newspaper with a layout that is personalized to a reader's interest
- The problem:
  - We don't want the newspaper to necessarily know what we are interested in
    - Our political opinions may be private
    - Our interest in certain stocks can be confidential
    - Or just because **our interests are our own**



# Personalized Newspapers

- The **non-private** solution:
  - Input:
    - **User input:** answers to an “interest questionnaire” and possible ratings for articles read
    - **Automated input:** the newspaper gathers information about which articles were read by the user, for how long and so on (appropriate for online newspapers)
  - The computation: **data mining algorithms** are run to determine what is of interest to the user and in which layout to present it

# Personalized Newspaper **Danger**

- Why do we want **privacy** regarding our interests and **why is this an important question?**
- Typical answer to why we want privacy
  - A personal feeling of discomfort
- A more concrete answer
  - **A danger to our autonomy**: if the newspaper knows our interests, political leanings etc., it could feed us **slanted information**
  - **Our interests could be used against us**

# Personalized Newspapers

- The solution – **secure computation**
  - The reader inputs its personal data
  - The newspaper inputs the articles and the “**rules**” to define the layout based on the reader’s data
  - The reader receives the personalized newspaper, the newspaper **learns nothing**

# Personalized Newspapers

- Privacy is clearly preserved
  - The newspaper learns nothing about the reader's data (interests, preferences, reading history etc.)
- There is no dilemma here even regarding computing the function
  - The newspaper learns nothing so the function can clearly be computed without compromising privacy

# Another Example: Personalized Shopping Catalogs

- The same aim:
  - Present customers with catalogs containing products that they are interested in
- The same problem:
  - We don't want the store to know all of our shopping history

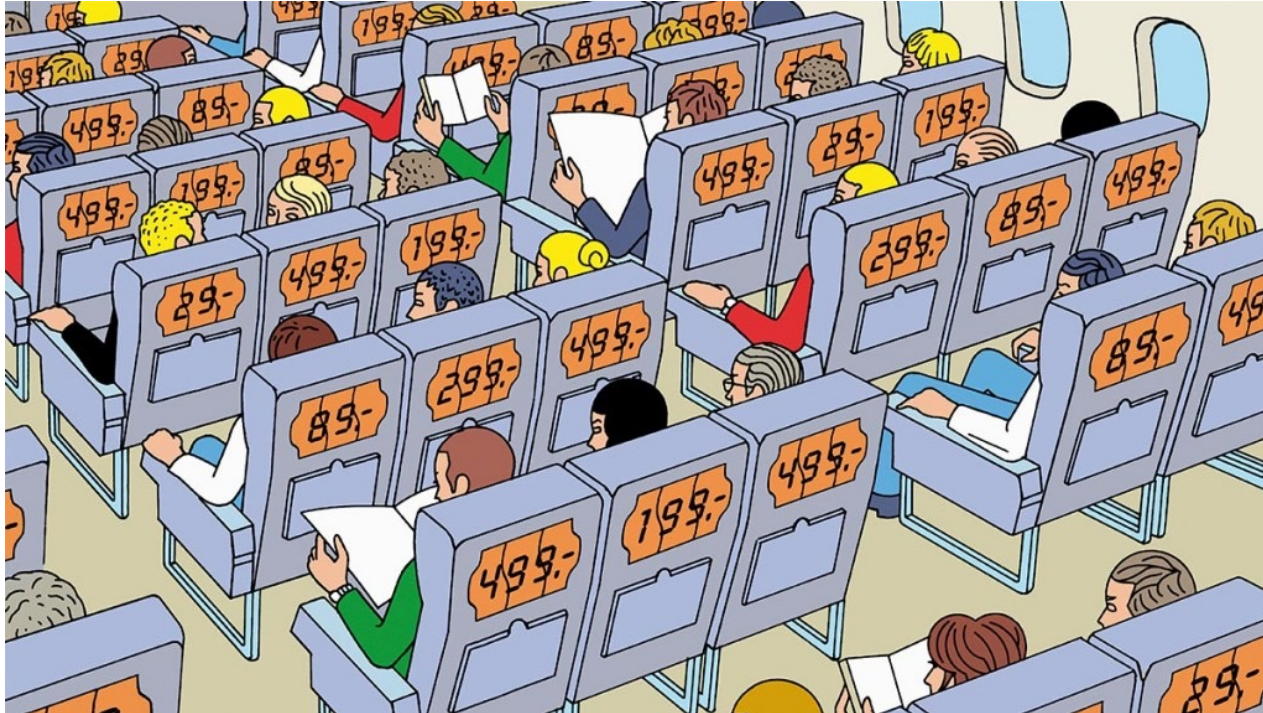
# Personalized Shopping Catalogs **Danger**

- A new problem: **price discrimination**
  - The store can charge much higher prices to customers who are known to not “shop around”
  - The secure computation solution **alone** once again does **not** solve the problem
- **The same solution:** secure computation

# Price Discrimination in the News

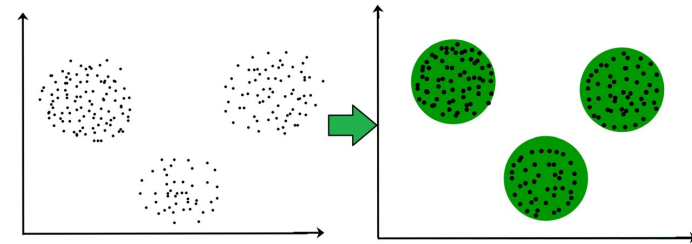
- How Retailers Use Personalized Prices to Test What You're Willing to Pay ([hbr link](#))

Create a **pricing profile** according to your characteristics and shopping history and habits (highest amount each shopper will pay)



# Privacy-Preserving Clustering

- Traditionally clustering algorithms assume **complete access** to data

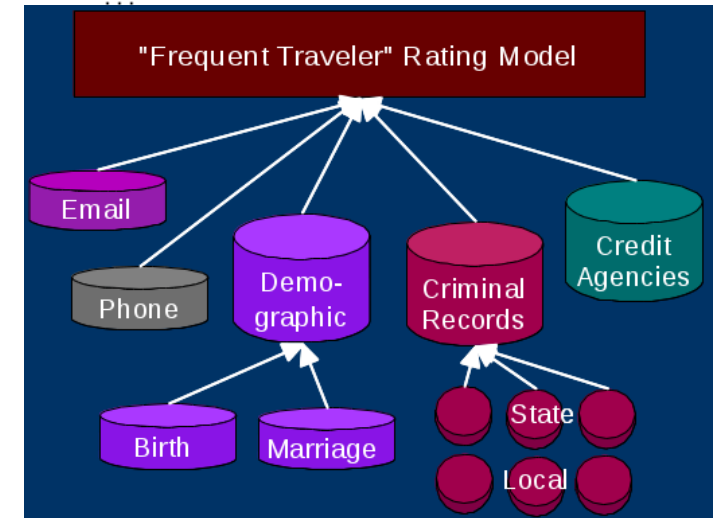


- Nowadays, data collected at several different sites
  - poses privacy and security concerns
- Can we create clusters and analyze them **without complete access to underlying data?**

# Vertically Partitioned Data

	Price	Color	Weight	...
Item 1	200	Blue	900g	...
Item 2	700	Red	300g	...
Item 3	200	Black	800g	...
Item 4	500	Yellow	500g	...
Item 5	800	Red	900g	...
Item 6	300	Yellow	300g	...
Item 7	500	Black	900g	...
⋮	⋮	⋮	⋮	⋮

Site 1      Site 2



# Privacy-Preserving Clustering

- Goal:
  - Cluster a set of entities
  - **Without revealing any value that the clustering is based on**
- Input:
  - Each site provides one attribute (column) of all entities
- Output:
  - Assignment of entities to clusters (nothing else)

# K-means Clustering

- Simple, but popular method for clustering
  - Iterative algorithm
  - Re-compute cluster centers (i.e., mean of data points in cluster)

**Input:** Database  $D$ , integer  $k$

**Output:** Cluster centers  $\mu_1 \dots \mu_k$

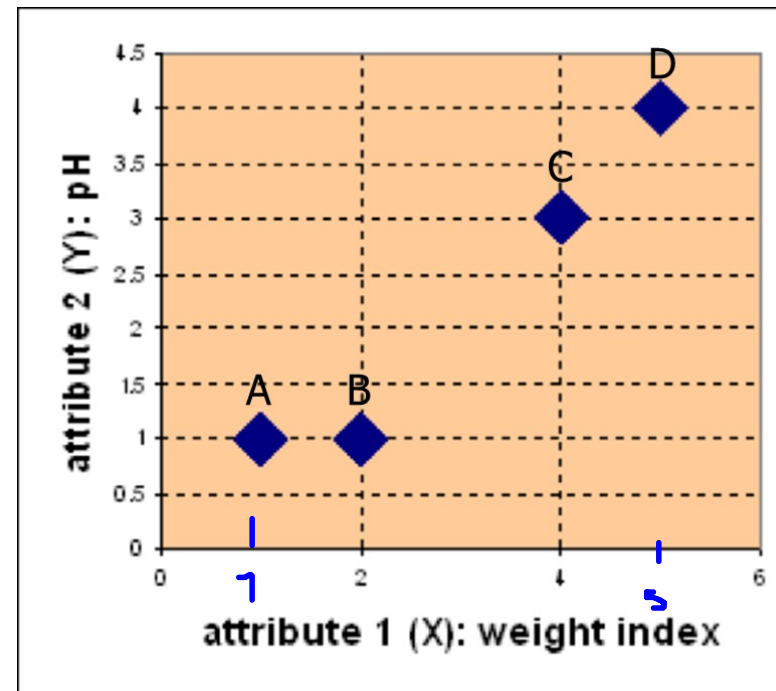
1. Arbitrarily select  $k$  objects from  $D$  as initial cluster centers  $\mu'_1 \dots \mu'_k$ .
  2. Repeat
    - (a)  $(\mu_1 \dots \mu_k) = (\mu'_1 \dots \mu'_k)$
    - (b) Assign each object  $d_i \in D$  to the cluster whose center is closest.
    - (c) Recompute the centers of the  $k$  clusters as  $\mu'_1 \dots \mu'_k$ .
- Until  $(\mu_1 \dots \mu_k)$  is close to  $(\mu'_1 \dots \mu'_k)$

# K-means Clustering Example

- Problem:

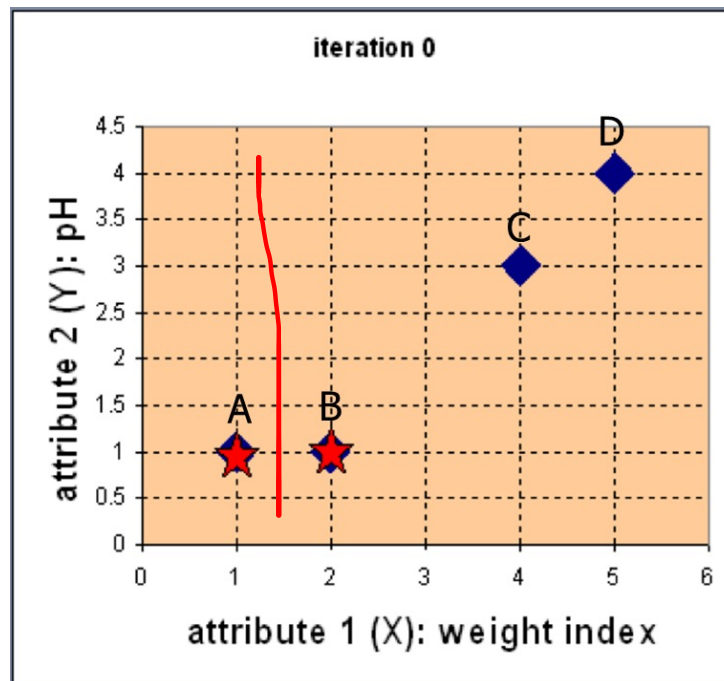
Suppose we have 4 types of medicines and each has two attributes (pH and weight index). Our goal is to group these objects into  $K=2$  group of medicine.

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



# K-means Clustering Example

- Step 1: Use initial seed points for partitioning  $c_1 = A, c_2 = B$



$D^0 =$	0	1	3.61	5	$c_1 = (1,1)$	group - 1
	1	0	2.83	4.24	$c_2 = (2,1)$	group - 2
	A	B	C	D	Euclidean distance	
	1	2	4	5	X	
	1	1	3	4	Y	

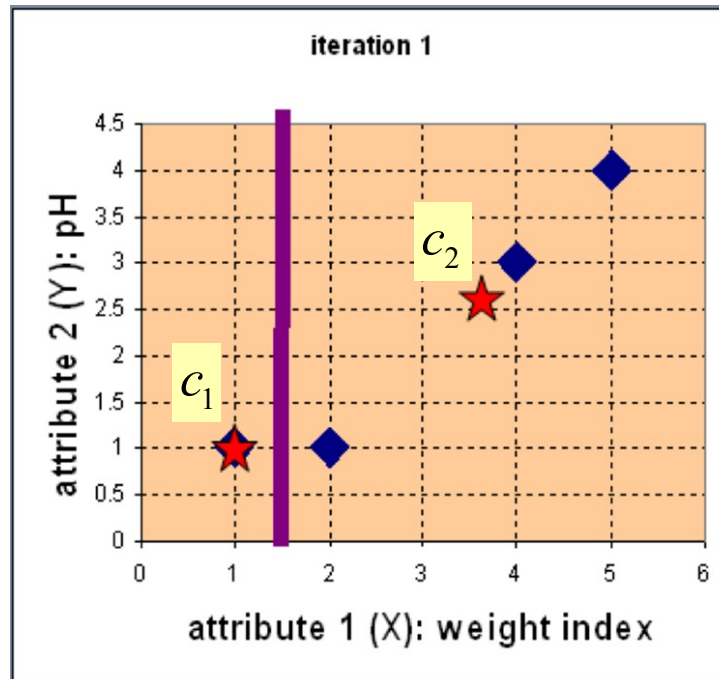
$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

# K-means Clustering Example

- Step 2: Compute new centroids of the current partition



Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

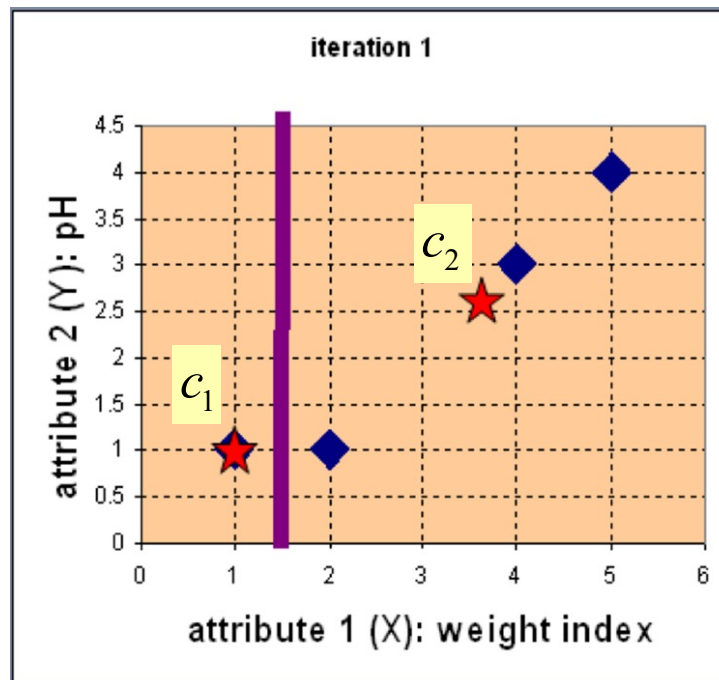
$$c_1 = (1, 1)$$

$$c_2 = \left( \frac{2 + 4 + 5}{3}, \frac{1 + 3 + 4}{3} \right)$$

$$= \left( \frac{11}{3}, \frac{8}{3} \right) = (3.6, 2.6)$$

# K-means Clustering Example

- Step 2: Renew membership based on new centroids



Compute the distance of all objects to the new centroids

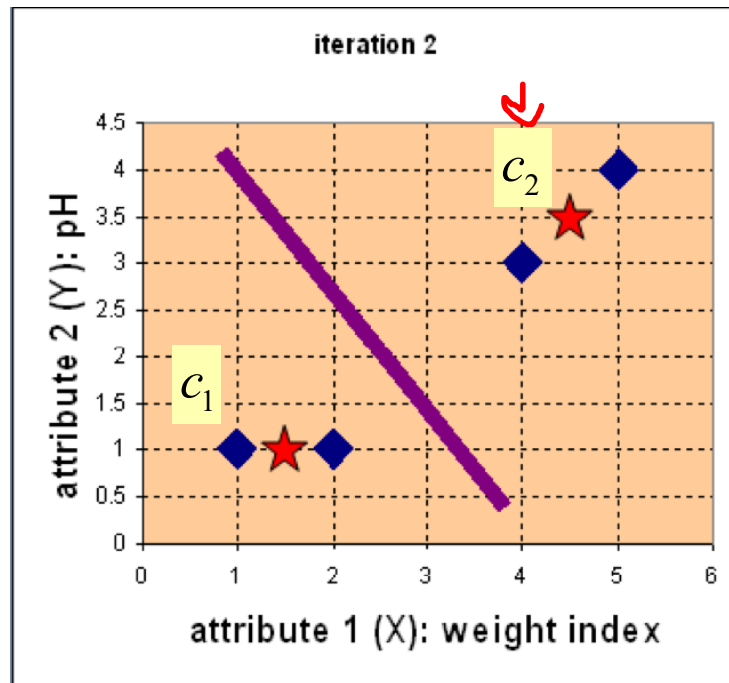
$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1, 1) \text{ group-1} \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) \text{ group-2} \end{array}$$

$A$	$B$	$C$	$D$	
$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix}$	$X$	$Y$		

Assign the membership to objects

# K-means Clustering Example

- Step 3: Repeat the first two steps until its convergence



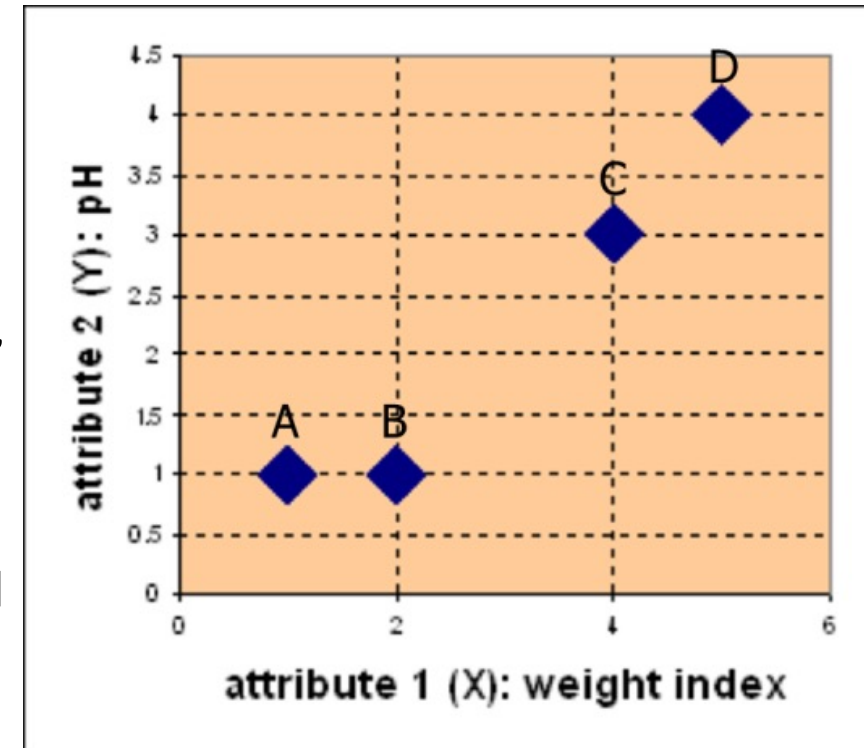
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left( \frac{1+2}{2}, \frac{1+1}{2} \right) = (1.5, 1)$$

$$c_2 = \left( \frac{4+5}{2}, \frac{3+4}{2} \right) = (4.5, 3.5)$$

# Privacy-Preserving K-means Clustering

- $k$  clusters,
  - Cluster centers  $\mu_i, i = 1, \dots, k$
  - $r$  sites:  $P_j, j = 1, \dots, r$
1. Each site has its component (attribute, one has weight and another has pH) of distance to cluster centers (e.g., weight, pH)
  2. To determine the closest cluster, need information on the distances of all sites to the cluster centers
    - the site holding the weight data cannot learn pH distances and vice-versa
  3. How to do this on a **private manner**?



# Privacy-Preserving K-means Clustering

- Based on three key ideas:
  1. **Disguise** the site components of the **distance with random values** that cancel out when combined
  2. **Compare distances** so only the comparison result is learned; no site knows the distances being compared
  3. **Permute the order of clusters** so the real meaning of the comparison results is unknown

# Privacy-Preserving K-means Clustering Algorithm

- Algorithm also requires three non-colluding sites
  - These parties may be among the parties holding data
  - could be external as well
- They need only know the number of sites  $r$  and the number of clusters  $k$ .
- We assume the non-colluding sites are  $P1$ ,  $P2$ , and  $P_r$  among the data holders.

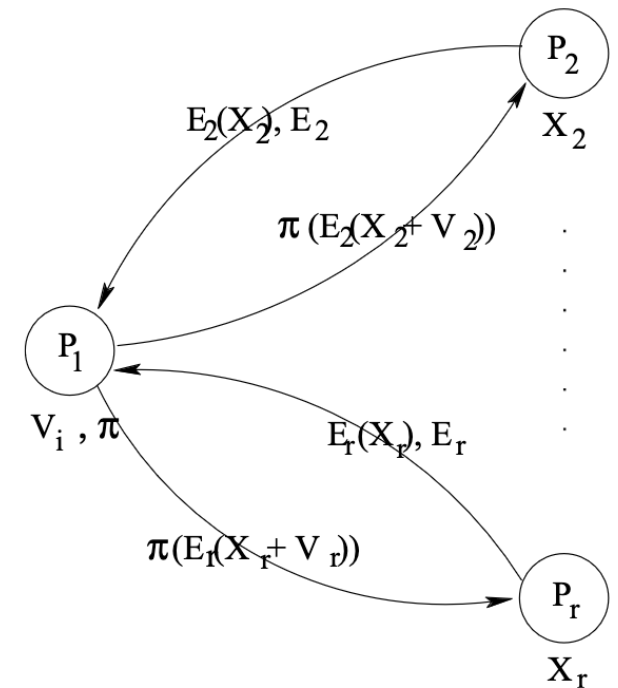
# Privacy-Preserving K-means Clustering Algorithm

1.  $P_1$  generates a length  $k$  random vector  $V_j$  for each site  $P_j$  ( $k$ : number of cluster), such that (random numbers from a uniform random distribution,):  $\sum_{i=1}^r \vec{V}_i = 0$

2.  $P_1$  chooses a permutation  $\pi$  of  $1, \dots, k$

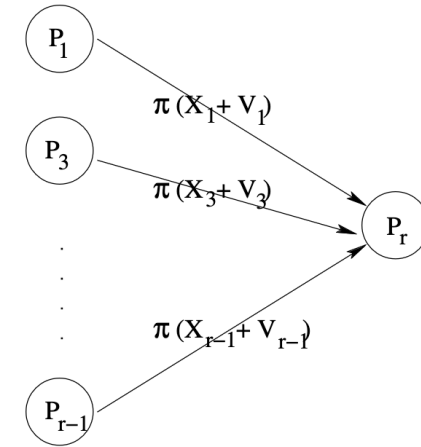
3.  $P_1$  and each  $P_j$  together:

1.  $P_1$  engages each site  $P_j$  in the permutation algorithm to generate the sum of  $V_j$  and  $P_j$ 's distances  $X_j$  ( $V_j + X_j$ )
2. The resulting vector is only known to  $P_j$  and is permuted by  $\pi$  only known to  $P_1$  (i.e.,  $P_j$  has  $\pi(V_j + X_j)$ )



# Privacy-Preserving K-means Clustering Algorithm

1.  $P_1$ , and  $P_2, \dots, P_{r-1}$  send their vectors to  $P_r$



2. Sites  $P_2$  and  $P_r$  engage in a series of **secure addition/ comparison** to find the (permuted) index of the minimum distance
3.  $P_2$  and  $P_r$  send the **minimum permuted index**  $i$  back to  $P_1$
4.  $P_1$  discloses the proper cluster  $\pi^{-1}(i)$  to the point (calculate new  $V$  vectors which are the new centers for the clusters)

# Secure Permutation

- Two parties A and B have n-dimensional vectors  $\vec{V} = (v_1, \dots, v_n)$  and  $\vec{X} = (x_1, \dots, x_n)$  respectively
- A also has a permutation  $\pi$  of the n numbers
- **Goal:** give B the result of  $\pi(\vec{X} + \vec{V})$  without disclosing anything else
  - Neither A nor B can learn the other's vector
  - B does not learn the permutation  $\pi$

# References

- Secure Multiparty Computation for Privacy-Preserving Data Mining, Y. Lindell and B. Pinkas, Journal of Privacy and Confidentiality 2009
- Computing Cooperatively with People You Don't Trust slides, David Evans, 2012
- Secure Multiparty Computation: Introduction slides, Ran Cohen, Seminar on Secure Multi-Party Computation 2017
- Privacy, Profiling, Targeted Marketing, and Data Mining (Jaideep Vaidya and Vijay Atluri)
- Privacy-Preserving K-means Clustering over Vertically Partitioned Data, Vaidya and Clifton, KDD'03
- Tools for Privacy Preserving Data Mining, Clifton et al, SIGKDD'02

# References

- Secure Multiparty Computation for Privacy-Preserving Data Mining, Y. Lindell and B. Pinkas, Journal of Privacy and Confidentiality 2009
- Computing Cooperatively with People You Don't Trust slides, David Evans, 2012
- Secure Multiparty Computation: Introduction slides, Ran Cohen, Seminar on Secure Multi-Party Computation 2017
- Privacy, Profiling, Targeted Marketing, and Data Mining (Jaideep Vaidya and Vijay Atluri)
- Privacy-Preserving K-means Clustering over Vertically Partitioned Data, Vaidya and Clifton, KDD'03
- Tools for Privacy Preserving Data Mining, Clifton et al, SIGKDD'02

# Bibliography

