ILLINOIS INSTITUTE OF TECHNOLOGY

CS 528 (Fall 2021)
Data Privacy & Security

Yuan Hong

Department of Computer Science

Illinois Institute of Technology

Chapter 8
Privacy Preserving Data Mining (Crypto)

PRIVACY PRESERVING DATA MINING

How can we privately mine data?

Perturbation

- Differential Privacy
- …

Cryptographic

- Lindell & Pinkas, Vaidya & Clifton
- Completely accurate, completely secure (tight bound on disclosure), appropriate for small number of parties

Condensation/Hybrid

CRYPTOGRAPHIC SOLUTIONS

Use Secure Multiparty Computation (SMC) techniques

Utilize cryptographic tools

Clear proofs of security

Next – classification, association rule mining, clustering

PRIVACY PRESERVING DATA MINING TOOLKIT

Many different data mining techniques often perform similar computations at various stages (e.g., computing sum, counting the number of items)

Toolkit

- simple computations – sum, union, intersection …
- assemble them to solve specific mining tasks – association rule mining, bayes classifier, …

The protocols may not be <u>truly secure</u> but more efficient than traditional SMC methods

Tools for Privacy Preserving Data Mining, Clifton, 2002

TOOLKIT

Secure functions

- Secure sum
- Secure union
- …

Applications

- Decision tree training based on horizontally partitioned data
- Association rule mining for horizontally partitioned data
- …

SECURE SUM

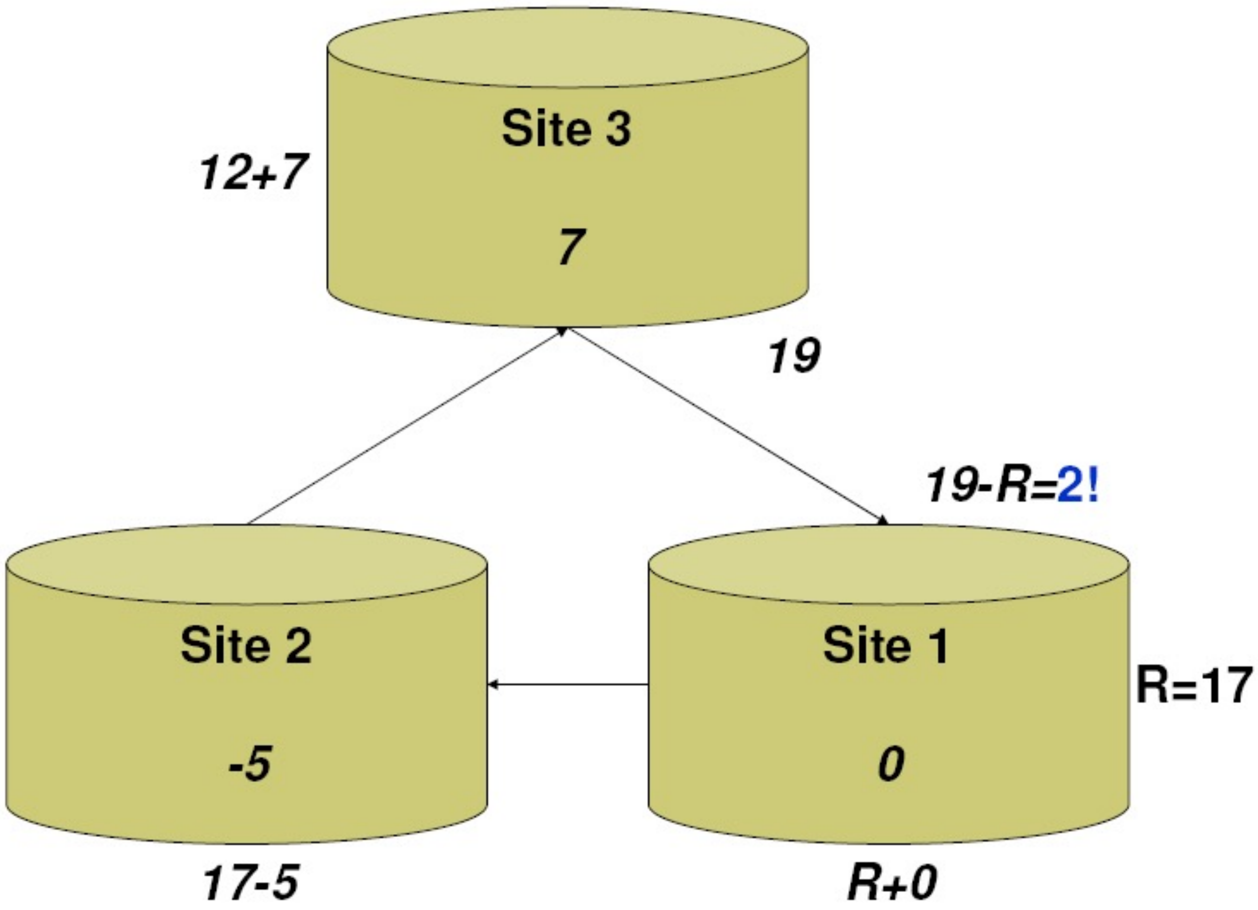
Leading site: (v1 + R) mod n

Site l receives:

$$V = R + \sum_{j=1}^{l-1} v_j \mod n$$

Leading site: (V-R) mod n

SECURE SUM



SECURE SUM - SECURITY

Does not reveal the real number

Is it secure?

- Site can collude!
- Each site can divide the number into shares, and run the algorithm multiple times with permutated nodes

SECURE UNION

Commutative encryption

- For any set of permutated keys and a message M

$$E_{K_{i_1}}(\ldots E_{K_{i_n}}(M)\ldots) = E_{K_{j_1}}(\ldots E_{K_{j_n}}(M)\ldots)$$

- For any set of permutated keys and message M1 and M2

$$Pr(E_{K_{i_1}}(\ldots E_{K_{i_n}}(M_1)\ldots) = E_{K_{j_1}}(\ldots E_{K_{j_n}}(M_2)\ldots)) < \epsilon$$

e.g., RSA, Pohlig-Hellman

Secure union

- Each site encrypts its items and items from other site, removes duplicates, and decrypts

SECURE UNION SECURITY

Does not reveal which item belongs to which site

Is it secure under the definition of secure multi-party computation?

- It reveals the number of items that are common in the sites!
- Revealing innocuous information leakage allows a more efficient algorithm than a fully secure algorithm

PRIVACY-PRESERVING ID3 ALGORITHM (FOR DECISION TREE)

CONSTRUCTION OF DECISION TREES

Training set: a data sample in which the classification is already known.

Greedy top down generation of decision trees.

- Each internal node of the tree partitions the data into groups based on a **partitioning attribute**, and a **partitioning condition** for the node
- **Leaf** node:
  - all (or most) of the items at the node belong to the same class, or
  - all attributes have been considered, and no further partitioning is possible.

BEST SPLITS

Pick best attributes and conditions on which to partition

The purity of a set S of training instances can be measured quantitatively in several ways.

- Notation: number of classes = k, number of instances = |S|, fraction of instances in class $i = p_i$.

The Gini measure of purity is defined as

$$\textbf{Gini (S) = 1 - } \sum_{i-1}^{k} p^2{}_i$$

- When all instances are in a single class, the Gini value is 0
- It reaches its maximum (of 1 –1/k) if each class the same number of instances.

BEST SPLITS (CONT.)

Another measure of purity is the entropy measure, which is defined as

$$\text{entropy}(S) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

When a set S is split into multiple sets Si, I=1, 2, …, r, we can measure the purity of the resultant set of sets as:

$$\text{purity}(S_1, S_2, \ldots.., S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \text{purity}(S_i)$$

The information gain due to particular split of S into S_i, i = 1, 2, …., r

Information-gain $(S, \{S_1, S_2, \ldots., S_r\}) = \text{purity}(S) - \text{purity}(S_1, S_2, \ldots S_r)$

BEST SPLITS (CONT.)

Measure of "cost" of a split:

Information-content (*S*, {*S*$_1$, *S*$_2$, ….., *S*$_r$})) = – $\sum_{i=1}^{r} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$

$$\textbf{Information-gain ratio} = \frac{\textbf{Information-gain } (S, \{S_1, S_2, \ldots\ldots, S_r\})}{\textbf{Information-content } (S, \{S_1, S_2, \ldots.., S_r\})}$$

The best split is the one that gives the maximum information gain ratio

FINDING BEST SPLITS

Categorical attributes (with no meaningful order):

- Multi-way split, one child for each value
- Binary split: try all possible breakup of values into two sets, and pick the best

Continuous-valued attributes (can be sorted in a meaningful order)

- Binary split:
    - Sort values, try each as a split point
        - E.g., if values are 1, 10, 15, 25, split at ≤ 1, ≤ 10, ≤ 15
    - Pick the value that gives best split
- Multi-way split:
    - A series of binary splits on the same attribute has roughly equivalent effect

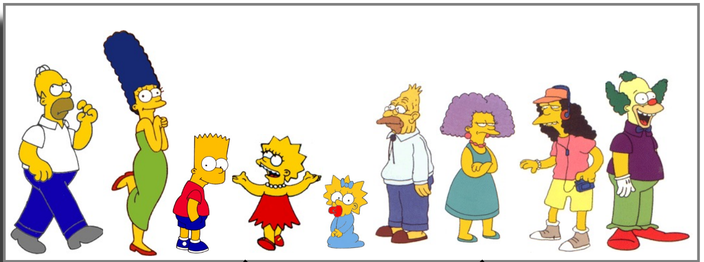
DECISION-TREE CONSTRUCTION ALGORITHM

Procedure *GrowTree* (S)
    Partition (S);

Procedure Partition (S)
    if (*purity* (S) > δ_p or $|S| < \delta_s$) **then**
        return;
    for each attribute A
        evaluate splits on attribute A;
    Use best split found (across all attributes) to partition
        S into $S_1, S_2, \ldots, S_r$,
    for $i = 1, 2, \ldots, r$
        Partition (S_i);

Person	Hair Length	Weight	Age	Class
 Homer	0”	250	36	M
 Marge	10”	150	34	F
 Bart	2”	90	10	M
 Lisa	6”	78	8	F
 Maggie	4”	20	1	F
 Abe	1”	170	70	M
 Selma	8”	160	41	F
 Otto	10”	180	38	M
 Krusty	6”	200	45	M

 Comic	8”	290	38	?



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

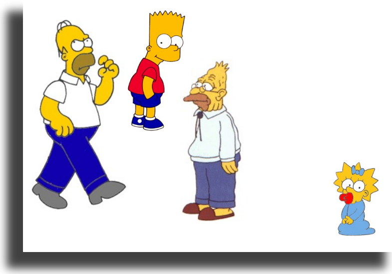
Entropy(4**F**,5**M**) = $-(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$

= **0.9911**

yes

no

Hair Length <= 5?



Let us try splitting on *Hair length*

Entropy(1**F**,3**M**) = $-(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$

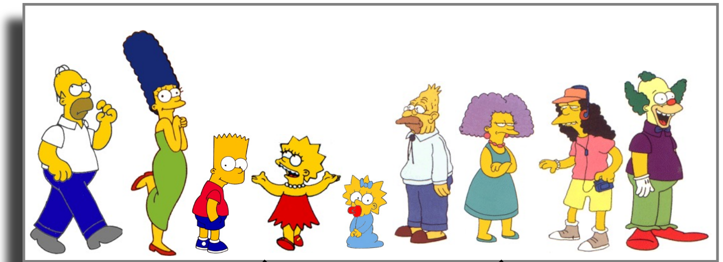
= **0.8113**

Entropy(3**F**,2**M**) = $-(3/5)\log_2(3/5) - (2/5)\log_2(2/5)$

= **0.9710**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Hair Length <= 5) = **0.9911** – (4/9 * **0.8113** + 5/9 * **0.9710**) = **0.0911**



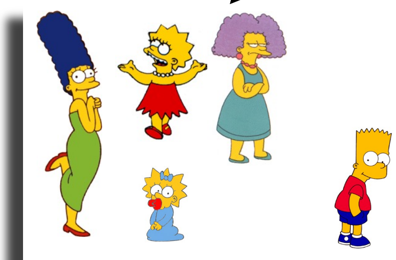
$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$

$= \mathbf{0.9911}$

yes — no

Weight <= 160?



Let us try splitting on *Weight*

$Entropy(4\mathbf{F},1\mathbf{M}) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5)$

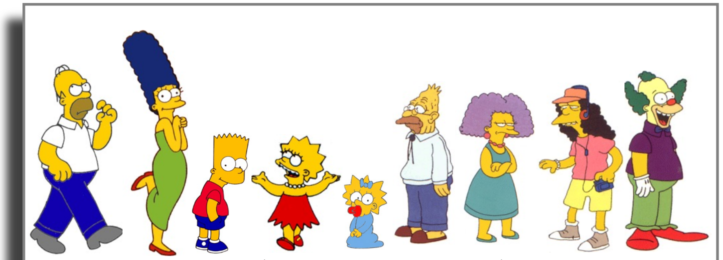
$= \mathbf{0.7219}$

$Entropy(0\mathbf{F},4\mathbf{M}) = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4)$

$= \mathbf{0}$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$Gain(\text{Weight} <= 160) = \mathbf{0.9911} - (5/9 * \mathbf{0.7219} + 4/9 * \mathbf{0}) = \mathbf{0.5900}$



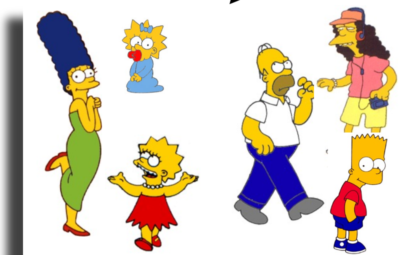
$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

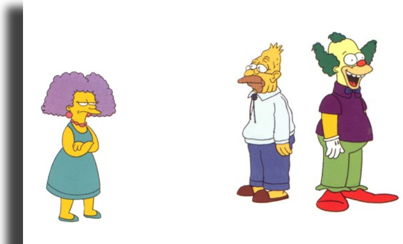
Entropy(4**F**,5**M**) = $-(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$
= **0.9911**

yes

no

age <= 40?





Let us try splitting on *Age*

Entropy(3**F**,3**M**) = $-(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$
= ***1***

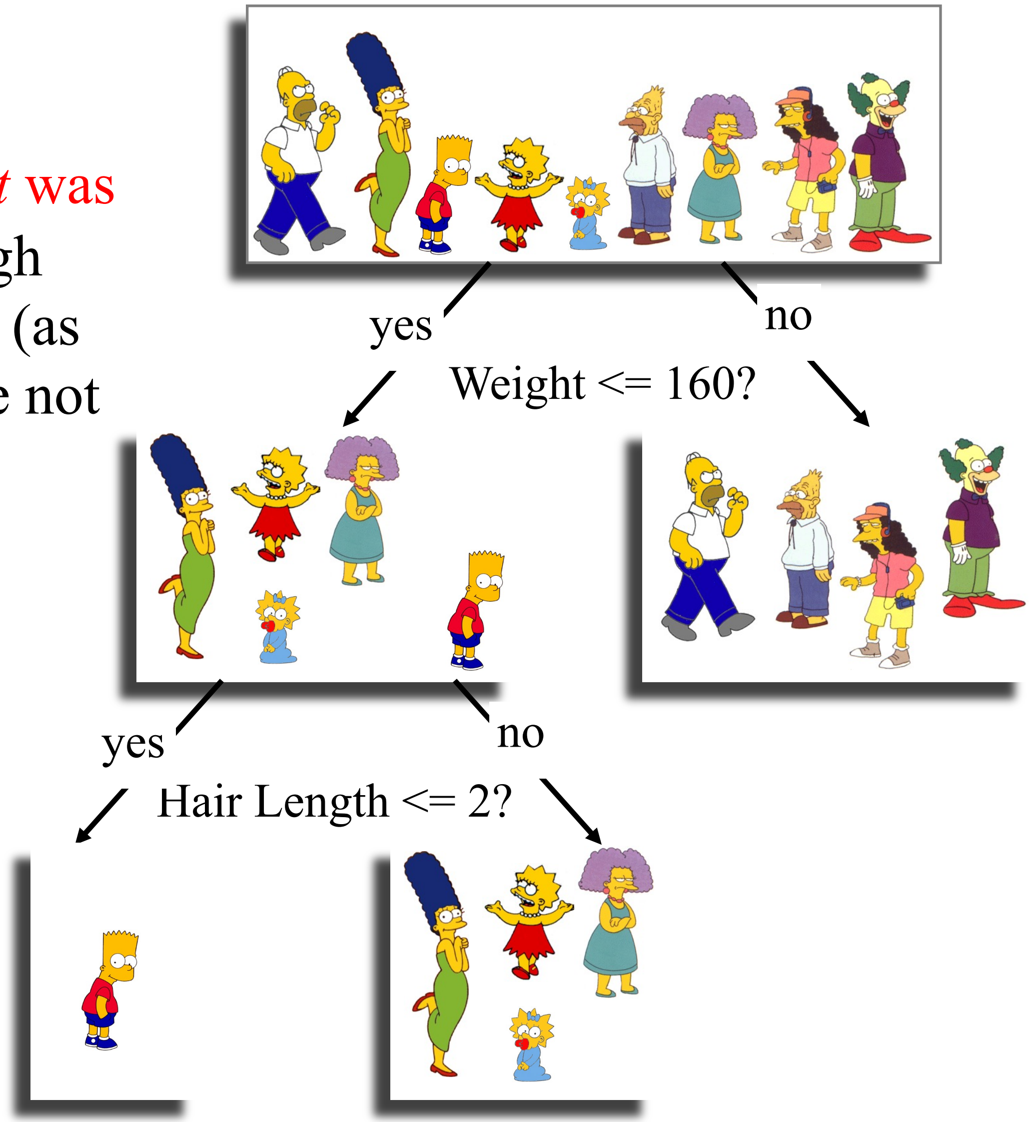
Entropy(1**F**,2**M**) = $-(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
= ***0.9183***

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Age <= 40) = **0.9911** – (6/9 * **1** + 3/9 * **0.9183**) = **0.0183**

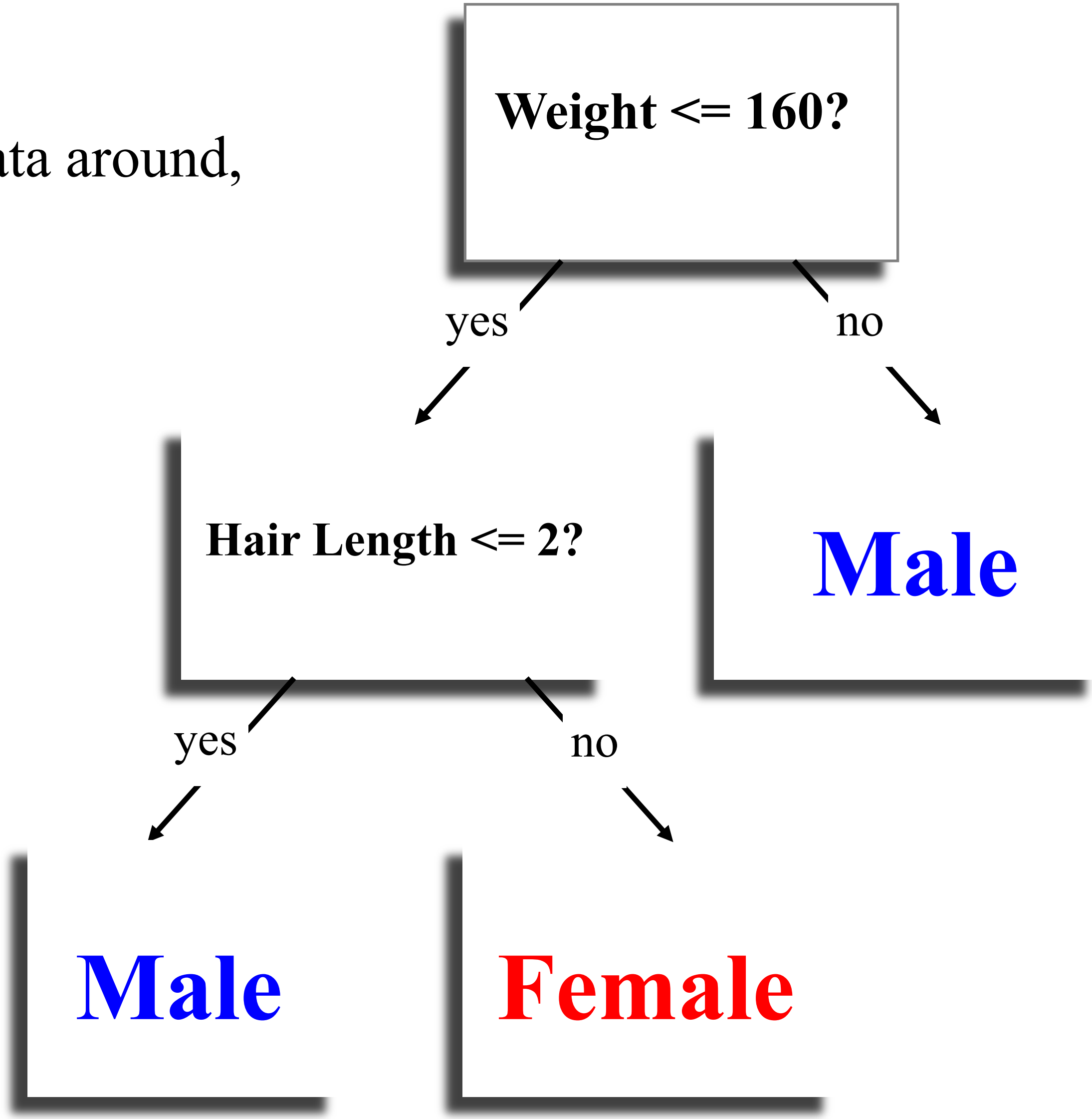
Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… So we recursively find the best split!

This time we find that we can split on *Hair length,* and we are done!



We don’t need to keep the data around, just the test conditions.

How would these people be classified?



ID3 Algorithm for Decision Tree Induction

- Greedy algorithm - tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - A test attribute is selected that “best” separates the data into partitions - information gain
  - Samples are partitioned recursively based on selected attributes

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

PRIVACY PRESERVING ID3 (HORIZONTAL PARTITION)

Input:

- R – the set of attributes
- C – the class attribute
- T – the set of transactions

Step 1: *If **R is empty**, return a leaf-node with the class value with the most transactions in T*

- Set of attributes is public
  - Both know if R is empty
- Use secure protocol for majority voting
  - Yao's protocol
    - Inputs $(|T_1(c_1)|,\ldots,|T_1(c_L)|)$, $(|T_2(c_1)|,\ldots,|T_2(c_L)|)$
    - Output i where $|T_1(c_i)|+|T_2(c_i)|$ is largest

PRIVACY PRESERVING ID3

Step 2: *If T consists of transactions which have all the* ***same value c*** *for the class attribute, return a leaf node with the value c*

Check equality to decide if at leaf node for class c_i

Various approaches for equality checking

- Yao'86
- Fagin, Naor '96
- Naor, Pinkas '01

PRIVACY PRESERVING ID3

Step 3:(a) *Determine the attribute that best classifies the transactions in T, let it be A*

$$Entropy(S) = -\sum_{v \in label(S)} P(v)\log P(v) = -\sum_{v \in label(S)} \frac{n_v}{n}\log\frac{n_v}{n}$$

- Essentially done by securely computing **x*(ln x)** where x is the sum of values from the two parties

P1 and P2 , i.e., x1 and x2, respectively

Step 3: (b,c) *Recursively call $ID3_\delta$ for the remaining attributes on the transaction sets $T(a_1),\ldots,T(a_m)$ where $a_1,\ldots, a_m$ are the values of the attribute A*

- Since the results of 3(a) and the attribute values are public, both parties can individually partition the database and prepare their inputs for the recursive calls

Protocol 1 (Protocol $\ln x$)

- **Input:** P_1 and P_2 have respective inputs v_1 and v_2 such that $v_1 + v_2 = x$. Denote $x = 2^n(1+\varepsilon)$ for n and ε as described above.

- **The protocol:**

  1. P_1 and P_2, upon input v_1 and v_2 respectively, run Yao's protocol for a circuit that outputs the following: (1) Random shares α_1 and α_2 such that $\alpha_1 + \alpha_2 = \varepsilon 2^N \bmod |\mathcal{F}|$, and (2) Random shares β_1,β_2 such that $\beta_1 + \beta_2 = 2^N \cdot n \ln 2 \bmod |\mathcal{F}|$.
  2. P_1 chooses $z_1 \in_R \mathcal{F}$ and defines the following polynomial

$$Q(z) = \mathrm{lcm}(2, \ldots, k) \cdot \sum_{i=1}^{k} \frac{(-1)^{i-1}}{2^{N(i-1)}} \frac{(\alpha_1 + z)^i}{i} - z_1$$

  3. P_1 and P_2 then execute a private polynomial evaluation with P_1 inputting $Q(\cdot)$ and P_2 inputting α_2, in which P_2 obtains $z_2 = Q(\alpha_2)$.
  4. P_1 and P_2 define $u_1 = \mathrm{lcm}(2, \ldots, k)\beta_1 + z_1$ and $u_2 = \mathrm{lcm}(2, \ldots, k)\beta_2 + z_2$, respectively. We have that $u_1 + u_2 \approx \mathrm{lcm}(2, \ldots, k) \cdot 2^N \cdot \ln x$

$$\mathrm{lcm}(2, \ldots k) \cdot 2^N \left(n \ln 2 + \varepsilon - \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3} - \cdots \frac{\varepsilon^k}{k} \right) \approx \mathrm{lcm}(2, \ldots k) \cdot 2^N \cdot \ln x$$

Protocol 2 (Protocol $Mult(a_1, a_2)$)

1. P_1 chooses a random value $b_1 \in \mathcal{F}$ and defines the linear polynomial $Q(z) = a_1 z - b_1$.

2. P_1 and P_2 engage in a private evaluation of Q, in which P_2 obtains $b_2 = Q(a_2) = a_1 \cdot a_2 - b_1$.

3. The respective outputs of P_1 and P_2 are defined as b_1 and b_2, giving us that $b_1 + b_2 = a_1 \cdot a_2$.

The correctness of the protocol (i.e., that b_1 and b_2 are uniformly distributed in $\mathcal{F}$ and sum up to $a_1 \cdot a_2$) is immediate from the definition of Q. Furthermore, the privacy follows from the privacy of the polynomial evaluation. We thus have the following proposition:

Protocol 3 (Protocol $x \ln x$)

1. P_1 and P_2 run Protocol 1 for privately computing shares of $\ln x$ and obtain random shares u_1 and u_2 such that $u_1 + u_2 \approx \ln x$.

2. P_1 and P_2 use two invocations of Protocol 2 in order to obtain shares of $u_1 \cdot v_2$ and $u_2 \cdot v_1$.

3. P_1 (resp., P_2) then defines his output w_1 (resp., w_2) to be the sum of the two *Mult* shares and $u_1 \cdot v_1$ (resp., $u_2 \cdot v_2$).

4. We have that $w_1 + w_2 = u_1 v_1 + u_1 v_2 + u_2 v_1 + u_2 v_2 = (u_1 + u_2)(v_1 + v_2) \approx x \ln x$ as required.

SECURITY PROOF TOOLS

- Real/ideal model: the real model can be simulated in the ideal model
  - Key idea – Show that whatever can be computed by a party participating in the protocol can be computed based on its input and output only
  - $\exists$ polynomial time S such that $\{S(x,f(x,y))\} \equiv \{View(x,y)\}$

SECURITY PROOF TOOLS

Composition theorem

- If a protocol is secure in the hybrid model *where the protocol uses* a trusted party that computes the (sub) functionalities, and we replace the calls to the trusted party by calls to secure protocols, *then the resulting protocol is secure*

- Prove that component protocols are secure, then prove that the combined protocol is secure

SECURE SUB-PROTOCOLS FOR PPDM

In general, PPDM protocols depend on few common sub-protocols.

Those common sub-protocols could be re-used to implement PPDM protocols

PRIVACY PRESERVING DISTRIBUTED MINING OF ASSOCIATION RULES ON HORIZONTALLY PARTITIONED DATA

ASSOCIATION RULES MINING

Assume data is horizontally partitioned

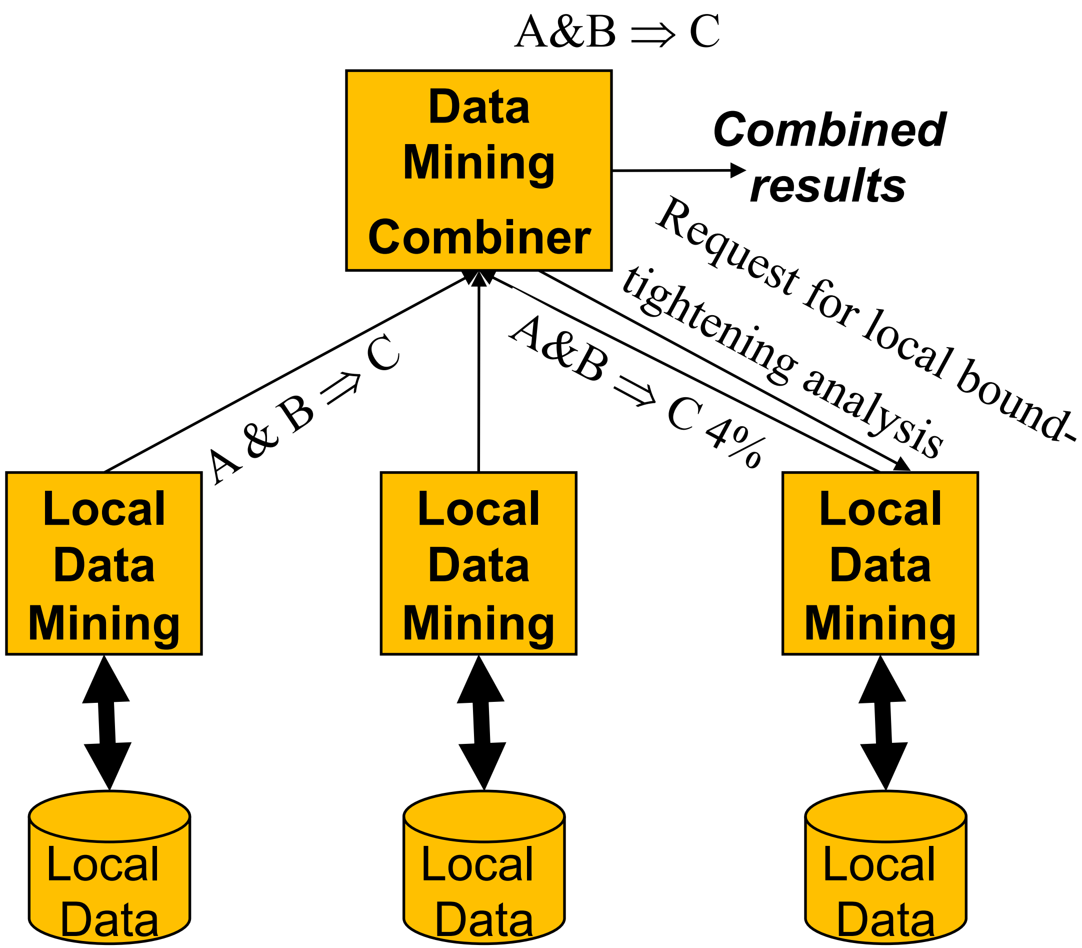
- Each site has complete information on a set of entities
- Same attributes at each site

If goal is to avoid disclosing entities, problem is easy

Basic idea: Two-Phase Algorithm

- First phase: compute candidate rules
    - Frequent globally $\Rightarrow$ frequent at **some site**
- Second phase: Compute frequency of candidates

ASSOCIATION RULES IN HORIZONTALLY PARTITIONED DATA



ASSOCIATION RULE MINING: HORIZONTAL PARTITIONING

What if we do not want to reveal which rule is supported at which site, the support count of each rule, or database sizes?

- Hospitals want to participate in a medical study
- But rules only occurring at one hospital may be a result of bad practices

PRIVACY-PRESERVING ASSOCIATION RULE MINING FOR HORIZONTALLY PARTITIONED DATA

Find the union of the locally large candidate itemsets securely

After the local pruning, compute the globally supported large itemsets securely

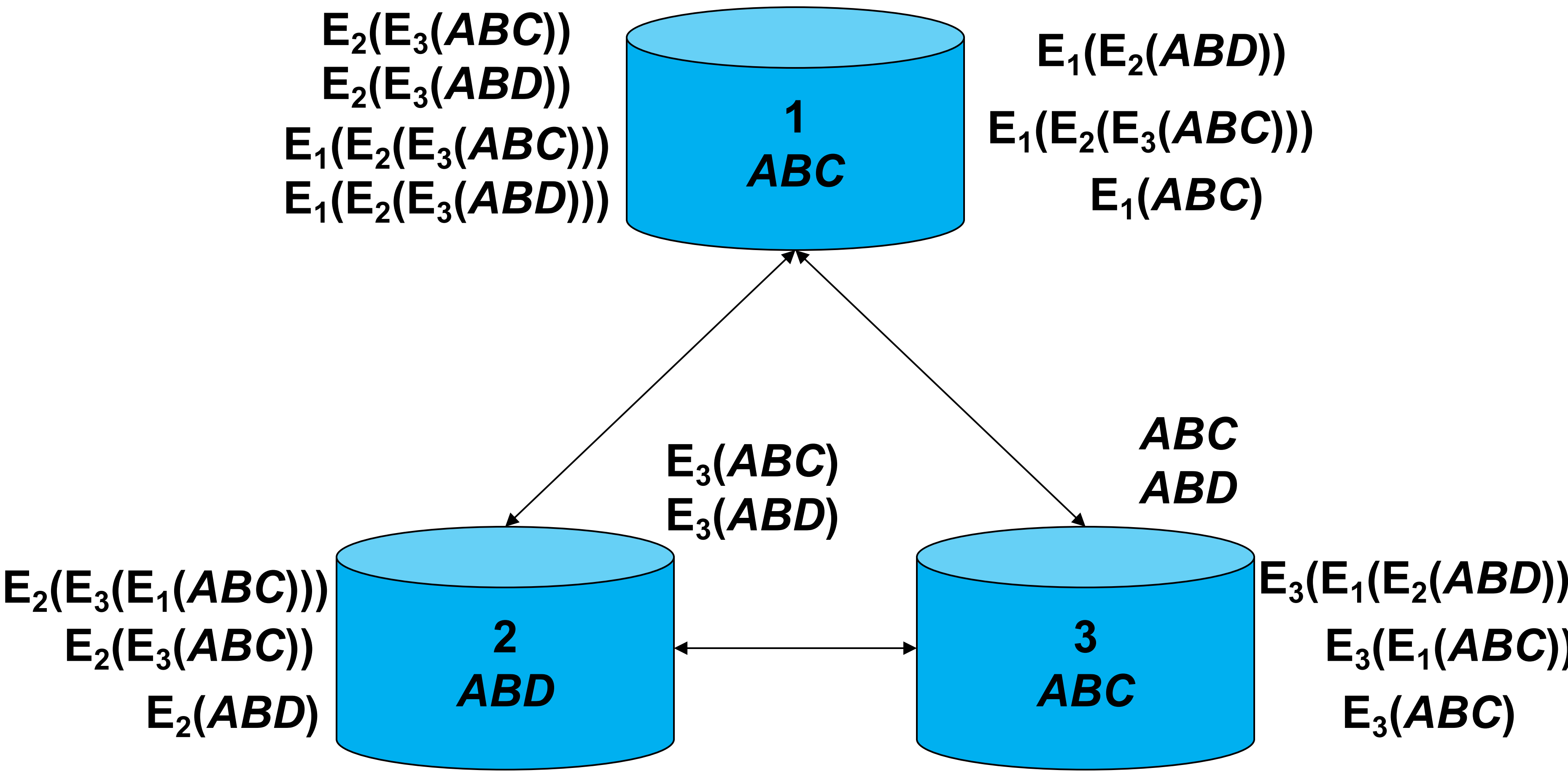
At the end check the confidence of the potential rules securely

SECURELY COMPUTING CANDIDATES

Compute local candidate set

Using secure union!

COMPUTING CANDIDATE SETS (SECURE UNION)



COMPUTE WHICH CANDIDATES ARE GLOBALLY SUPPORTED?

Goal: To check whether

(1) $$X.\text{sup} \geq s * \sum_{i=1}^{n} |DB_i|$$

(2) $$\sum_{i=1}^{n} X.\text{sup}_i \geq \sum_{i=1}^{n} s * |DB_i|$$

(3) $$\sum_{i=1}^{n} (X.\text{sup}_i - s * |DB_i|) \geq 0$$

WHICH CANDIDATES ARE GLOBALLY SUPPORTED? (CONTINUED)

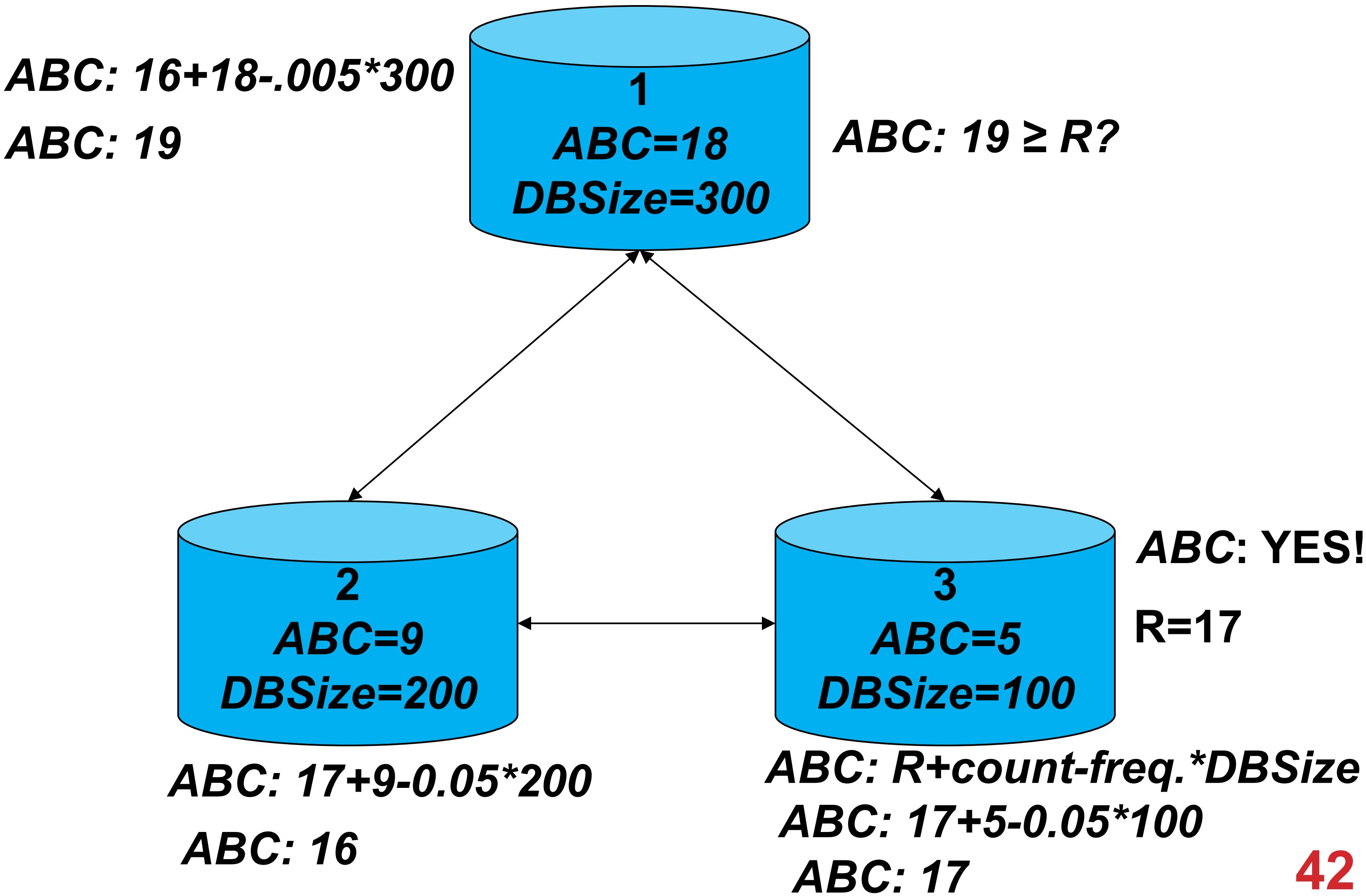
- Securely compute sum then check if sum ≥ 0
- Is this a good approach?

- Sum is disclosed!

Securely compute Sum - R

Securely compare Sum ≥ R?

COMPUTING FREQUENT: IS *ABC* ≥ 5%?



COMPUTING CONFIDENCE

Checking confidence can be done by the previous protocol. Note that checking confidence for X ⇒ Y

$$\frac{\{X \cup Y\}.\sup}{X.\sup} \geq c \Rightarrow \frac{\sum_{i=1}^{n} XY.\sup_i}{\sum_{i=1}^{n} X.\sup_i} \geq c$$

$$\Rightarrow \sum_{i=1}^{n} (XY.\sup_i - c * X.\sup_i) \geq 0$$

PRIVACY PRESERVING K-MEANS CLUSTERING OVER VERTICALLY PARTITIONED DATA

PROBLEM DEFINITION

Goal:

- Cluster the known set of common entities without revealing any value that the clustering is based on.

Input:

- Each user provides one attribute of all items.

Output:

- Assignment of entities to clusters.
- Cluster centers themselves.

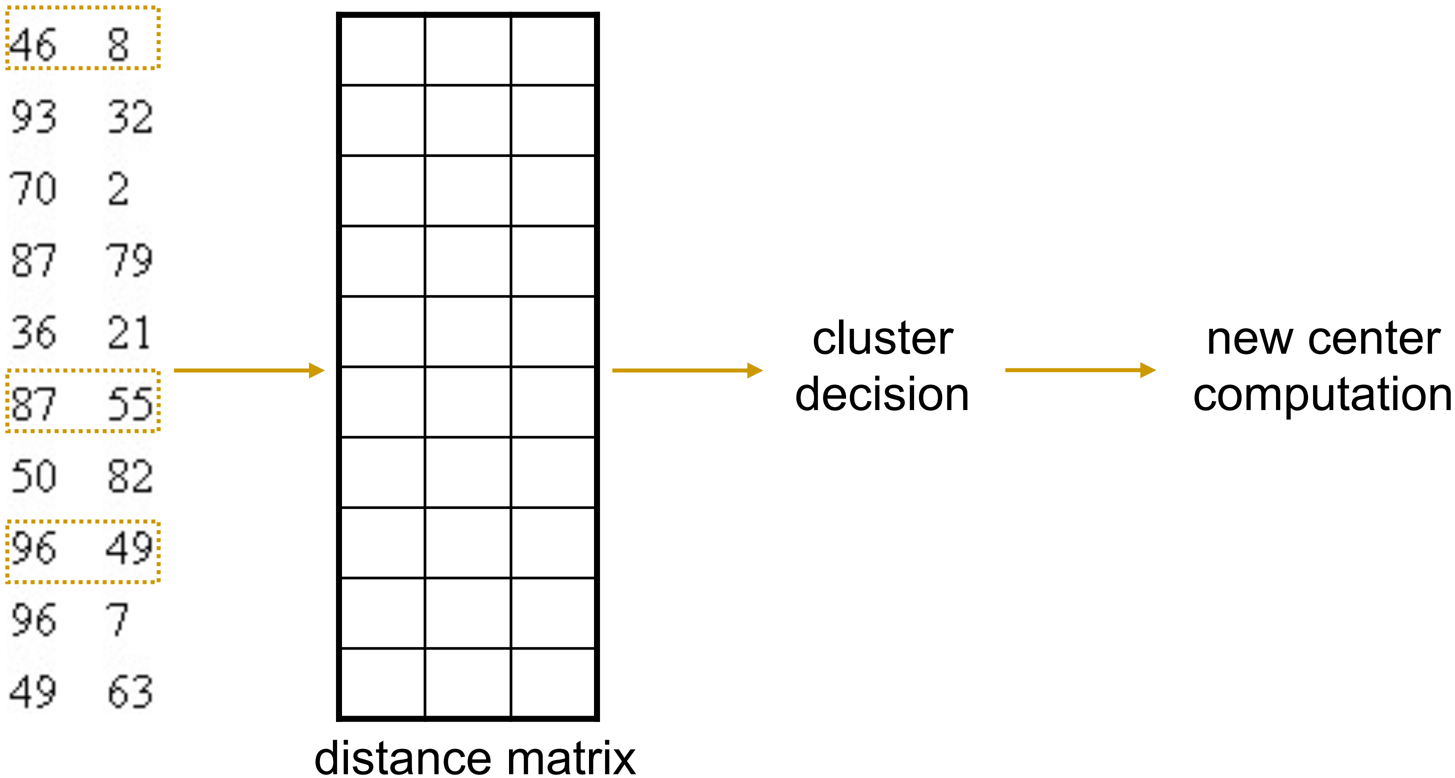
K-MEANS CLUSTERING

Input: Database D, integer k
Output: Cluster centers $\mu_1 \ldots \mu_k$

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

   Until $(\mu_1 \ldots \mu_k)$ is close to $(\mu'_1 \ldots \mu'_k)$

K-MEANS CLUSTERING



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	...
⋮	⋮	⋮	⋮	

User 1 User 2

TERMINOLOGY

r: # of users, each having different attributes for the same set of items.

n: # of the common items.

k: # of clusters required.

u_i: each cluster mean, i = 1, …, k.

u_{ij}: projection of the mean of cluster i on user j (w.r.t. the attributes)

Final result for user j:

- The final value / position of u_{ij}, i = 1, …, k.
- Cluster assignments: $clust_i$ for all points i = 1, …, n.

repeat
  for all $j = 1 \ldots r$ **do**
    for $i = 1 \ldots k$ **do**
      $\mu_{ij} \leftarrow \mu'_{ij}$
      $Cluster[i] = \emptyset$
    end for
  end for
  for $g = 1 \ldots n$ **do**
    for all $j = 1 \ldots r$ **do**
      {Compute the distance vector $\vec{X}_j$ (to each cluster) for point g.}
      for $i = 1 \ldots k$ **do**
        $x_{ij} = |data_{gj} -_D \mu_{ij}|$
      end for
    end for
    Each site puts g into $Cluster[closest\_cluster]$ {$closest\_cluster$ is Algorithm 3}
  end for
  for all $j = 1 \ldots r$ **do**
    for $i = 1 \ldots k$ **do**
      $\mu'_{ij} \leftarrow$ mean of j's attributes for points in $Cluster[i]$
    end for
  end for
until $checkThreshold$ {Algorithm 2}

SECURELY FINDING THE CLOSEST CLUSTER

The problem is formally defined as follows. Consider r parties $P_1, \ldots, P_r$, each with their own k-element vector $\vec{X}_i$:

$$P_1 \text{ has } \vec{X}_1 = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{k1} \end{bmatrix}, P_2 \text{ has } \begin{bmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{k2} \end{bmatrix}, \ldots, P_r \text{ has } \begin{bmatrix} x_{1r} \\ x_{2r} \\ \vdots \\ x_{kr} \end{bmatrix}$$

The goal is to compute the index l that represents the row with the minimum sum. Formally, find

$$\underset{i=1..k}{argmin}\left(\sum_{j=1..r} x_{ij}\right)$$

SECURELY FINDING THE CLOSEST CLUSTER

The security of the algorithm is based on three key ideas.

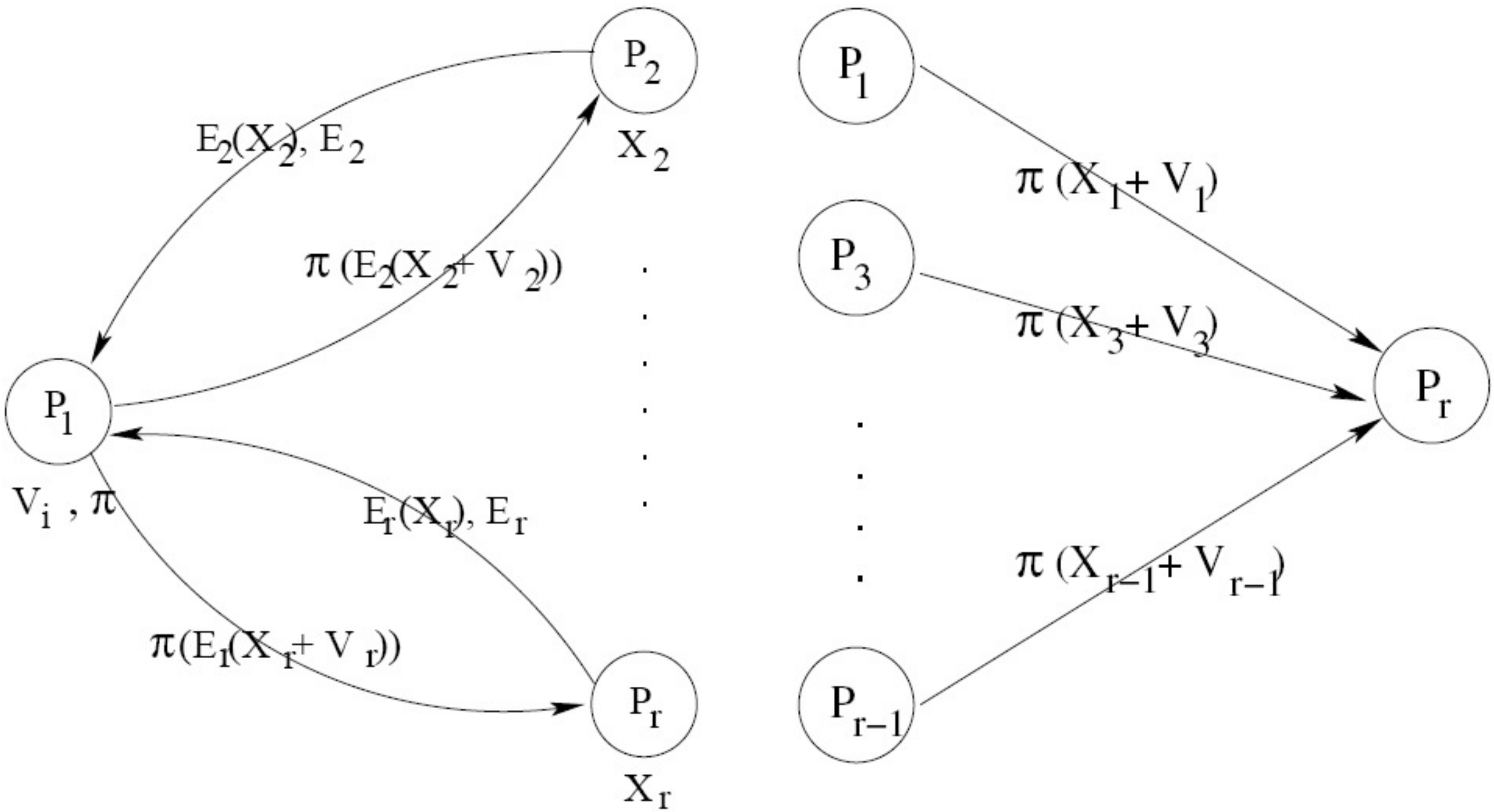
- Disguise the site components of the distance with random values that cancel out when combined.
- Permute the order of clusters so the real meaning of the comparison results is unknown.
- Compare distances so only the comparison result is learned; no party knows the distances being compared.

SECURELY FINDING THE CLOSEST CLUSTER

{Stage 1: Between P_1 and all other parties}
P_1 generates r random vectors $\vec{V_i}$ summing to $\vec{0}$ (see Algorithm 4).
P_1 generates a random permutation π over k elements
for all $i = 2 \dots r$ **do**
  $\vec{T_i}$ (at P_i) $= add\_and\_permute(\vec{V_i}, \pi(\text{at } P_1), \vec{X_i}(\text{at } P_i))$ {This is the permutation algorithm described in Section 2.2}
end for
P_1 computes $\vec{T_1} = \pi(\vec{X_1} + \vec{V_1})$

{Stage 2: Between all but P_2 and P_r}
for all $i = 1, 3 \dots r-1$ **do**
  P_i sends $\vec{T_i}$ to P_r
end for
P_r computes $\vec{Y} = \vec{T_1} + \sum_{i=3}^{r} \vec{T_i}$

SECURELY FINDING THE CLOSEST CLUSTER



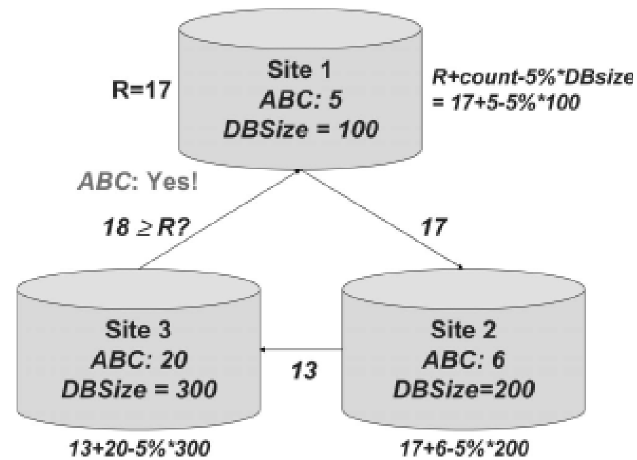
SECURELY FINDING THE CLOSEST CLUSTER

{Stage 3: Involves only P_2 and P_r}
$minimal \leftarrow 1$
for j=2..k **do**
  if $secure\_add\_and\_compare(Y_j + T_{2j} < Y_{minimal} + T_{2minimal})$ **then**
    $minimal \leftarrow j$
  end if
end for

{Stage 4: Between P_r (or P_2) and P_1}
Party P_r sends $minimal$ to P_1
P_1 broadcasts the result $\pi^{-1}(minimal)$

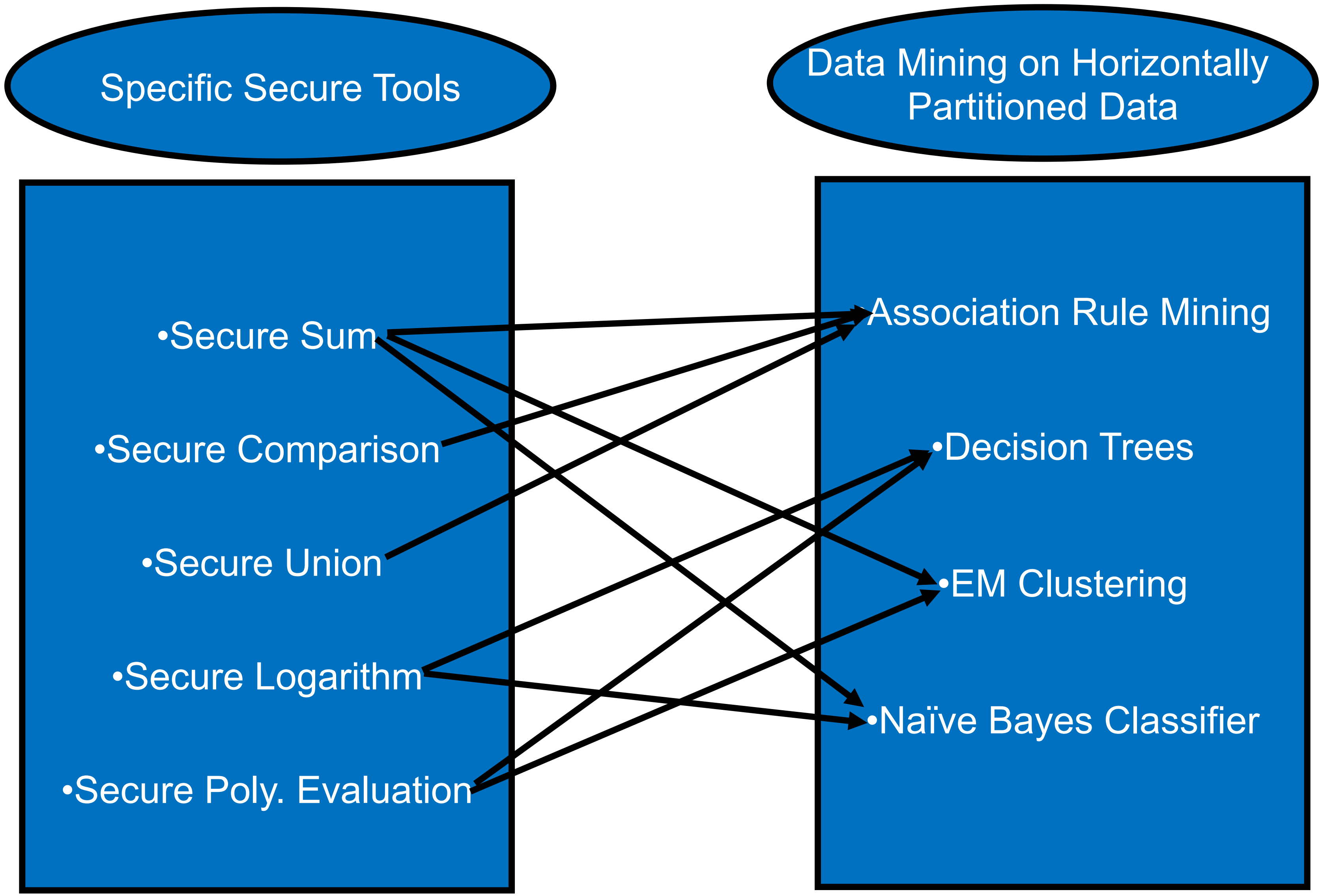
CHECK THRESHOLD

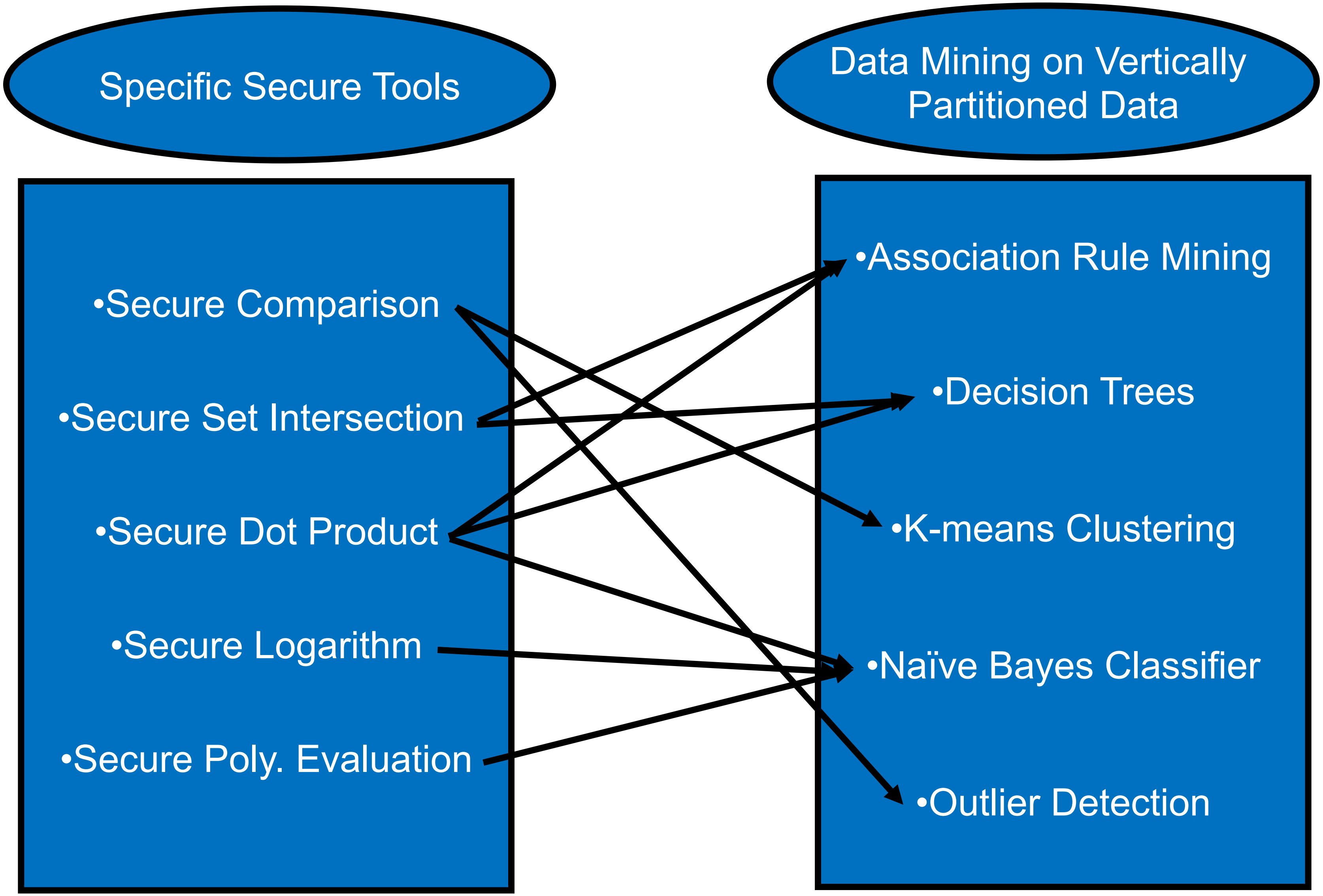
for all $j = 1 \ldots r$ **do**
  $d_j \leftarrow 0$
  for $i = 1 \ldots k$ **do**
    $d_j \leftarrow d_j + |\mu'_{ij} -_D \mu_{ij}|$
  end for
end for
{Securely compute if $\sum d_j \leq Th$.}
At P_1: m = rand()
for j=1 … r-1 **do**
  P_j sends $m + d_j \pmod{n}$ to P_{j+1}
end for
At P_r: $m = m + d_r$
At P_1: $Th' = Th + m$
P_1 and P_r return $secure\_add\_and\_compare(m - Th' \pmod{n} > Th' - m \pmod{n})$ {Secure comparison is described in Section 2.3.}



IF HORIZONTALLY PARTITIONED

	Price	Color	Weight	…	
Item 1	200	Blue	900g	…	User 1
Item 2	700	Red	300g	…	User 2
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	…	
⋮	⋮	⋮	⋮		


Specific Secure Tools
Data Mining on Horizontally Partitioned Data
•Secure Sum
•Secure Comparison
•Secure Union
•Secure Logarithm
•Secure Poly. Evaluation
•Association Rule Mining
•Decision Trees
•EM Clustering
•Naïve Bayes Classifier


Specific Secure Tools
Data Mining on Vertically Partitioned Data
•Secure Comparison
•Secure Set Intersection
•Secure Dot Product
•Secure Logarithm
•Secure Poly. Evaluation
•Association Rule Mining
•Decision Trees
•K-means Clustering
•Naïve Bayes Classifier
•Outlier Detection

SUMMARY OF SMC BASED PPDM

Mainly used for distributed data mining.

Provably secure under some assumptions.

Efficient/specific cryptographic solutions for many distributed data mining problems are developed.

Mainly semi-honest assumption (i.e. parties follow the protocols)

DRAWBACKS FOR SMC BASED PPDM

Drawbacks:

- Still not efficient enough for very large datasets
- Semi-honest model may not be realistic
- Malicious model is even slower

POSSIBLE NEW DIRECTIONS

New models that can trade-off better between efficiency and security

Game theoretic / incentive issues in PPDM

Combining anonymization and cryptographic techniques for PPDM

ACKNOWLEDGMENTS

Note: Some of the slides in this lecture are based on material created by

- Dr. Jaideep Vaidya at Rutgers University
- Dr. Li Xiong at Emory University
- Dr. Murat Kantarcioglu at UT Dallas