Collaborative Privacy Preserving Data Mining in Vertically Partitioned Databases

Ehud Gudes

Ben-Gurion University, Israel

This talk presents joint work with Boris Rozenberg

Talk Outline

- ***Motivation for Privacy-Preserving Distributed Data Mining***

    Overview of association rules

- ***Overview of Previous techniques(Clifton et al)***
    - Secure Multi-party computation
    - Horizontal Association Rules
    - Vertical Association Rules
- ***Our technique – Vertical association Rules***
    - Two Party Algorithm
    - Multi-party Algorithm
    - Analysis and comparison to Clifton's
- ***Conclusions***

Public Perception of Data Mining

- Fears of loss of privacy constrain data mining
  - Protests over a National Registry
    - *In Japan*
  - Data Mining Moratorium Act
    - *Would stop all data mining R&D by DoD*

- But data mining gives summary results
  - Does this violate privacy?
- *The problem isn't Data Mining, it is the infrastructure to support it!*

Privacy constraints don't prevent data mining

- Goal of data mining is summary results
  - Association rules
  - Classification
  - Clusters
- The results alone need not violate privacy
  - Contain no individually identifiable values
  - Reflect overall results, not individual organizations

The problem is computing the results without access to the private data!

European Union Data Protection Directives

- Directive 95/46/EC
  - Passed European Parliament 24 October 1995
  - Goal is to ensure free flow of information
    - *Must preserve privacy needs of member states*
  - Effective October 1998
- Effect
  - Provides guidelines for member state legislation
    - Not directly enforceable
  - Forbids sharing data with states that don't protect privacy
    - Non-member state must provide adequate protection,
    - Sharing must be for "allowed use", or
    - Contracts ensure adequate protection
  - US "Safe Harbor" rules provide means of sharing (July 2000)
    - Adequate protection
    - But voluntary compliance
- Enforcement is happening
  - Microsoft under investigation for Passport (May 2002)
  - Already fined by Spanish Authorities (2001)

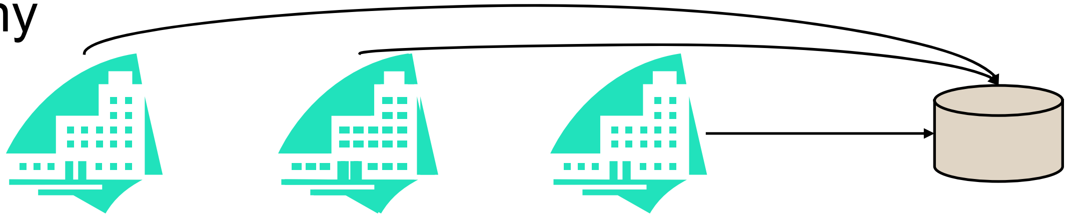
EU 95/46/EC: Meeting the Rules

- Personal data is any information that can be traced directly *or indirectly* to a specific person
- Use allowed if:
  - Unambiguous consent given
  - Required to perform contract with subject
  - Legally required
  - Necessary to protect vital interests of subject
  - In the public interest, or
  - Necessary for legitimate interests of processor and doesn't violate privacy
- Some uses specifically proscribed
  - Can't reveal racial/ethnic origin, political/religious beliefs, trade union membership, health/sex life
- Must make data available to subject
  - Allowed to object to such use
  - Must give advance notice / right to refuse direct marketing use
- Limits use for automated decisions

europa.eu.int/comm/internal_market/en/dataprot/law

Example: Patient Records

- My health records split among providers
  - Insurance company
  - Pharmacy
  - Doctor
  - Hospital



- Each agrees not to release the data without my consent
- Medical study wants correlations across providers
  - Rules relating complaints/procedures to “unrelated” drugs
- Does this need my consent?
  - *And that of every other patient!*
- It shouldn’t
  - **Rules don’t disclose my individual data!**

Techniques - Data Obfuscation

- Agrawal and Srikant, SIGMOD’00
  - Added noise to data before delivery to the data miner
  - Technique to reduce impact of noise on learning a decision tree
  - Improved by Agrawal and Aggarwal, SIGMOD’01
- Several later approaches for Association Rules
  - Evfimievski et al., KDD02
  - Rizvi and Haritsa, VLDB02
  - Kargupta, NGDM02

a different approach:
Use Secure Computation

- Goal: Only trusted parties see the data
  - They already have the data
  - Cooperate to share only global data mining results
- Proposed by Lindell & Pinkas, CRYPTO'00
  - Two parties, each with a portion of the data
  - Learn a **decision tree** without sharing data
- Can we do this for other types of data mining?

YES!

Review - Association Rules

- Retail shops are often interested in associations between different items that people buy.
  - Someone who buys bread is likely also to buy milk
  - A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*.
- Associations information can be used in several ways.
  - E.g. when a customer buys a particular book, an online shop may suggest associated books.
- **Association rules**:

  bread $\Rightarrow$ *milk ;*

  DB-Concepts, OS-Concepts $\Rightarrow$ Networks

Association Rules (Cont.)

- Rules have an associated support, as well as an associated confidence.
- **Support** is a measure of what fraction of the population satisfies both the antecedent and the consequent of the rule.
  - E.g. suppose only 0.001 percent of all purchases include milk and screwdrivers. The support for the rule *milk* $\Rightarrow$ *screwdrivers* is low.
  - We usually want rules with a reasonably high support
- **Confidence** is a measure of how often the consequent is true when the antecedent is true.
  - E.g. the rule *bread* $\Rightarrow$ *milk* has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk.

    Note that the confidence of *bread* $\Rightarrow$ *milk* may be very different from the confidence of *milk* $\Rightarrow$ *bread*, although both have the same support.

Finding Association Rules

- We are generally only interested in association rules with reasonably high support (e.g. support of 5% or greater)
- Naïve algorithm
  1. Consider all possible sets of relevant items.
  2. For each set find its support
     1. **Large itemsets**: sets with sufficiently high support
  3. Use large itemsets to generate association rules.
     1. From itemset A generate rule $A - \{b\} \Rightarrow b$ for each $b \in A$.
        - Support of rule = support (A).
        - Confidence of rule = support (A) / support (A - $\{b\}$)

The Naïve approach requires exponential space!

Finding Association Rules (Cont)

The Apriori Principle:

- *All subsets of a frequent itemset are frequent*
- *e.g if ABC is frequent then AB, BC and AC must be frequent*

The Apriori algorithm:

- *At iteration k, generate k-size candidates for which all k-1 subsets are frequent and then count their support*
- *Most popular association rules algorithm!*

Apriori Algorithm

Init: Scan the transactions to find $\mathcal{F}_1$, the set of all frequent 1-itemsets, together with their counts;

For (k=2; $\mathcal{F}_{k-1} \neq \emptyset$; k++)

1) Candidate Generation - C_k, the set of candidate k-itemsets, from $\mathcal{F}_{k-1}$, the set of frequent (k-1)-itemsets found in the previous step;

2) Candidates pruning - a necessary condition of candidate to be frequent is that each of its (k-1)-itemset is frequent.

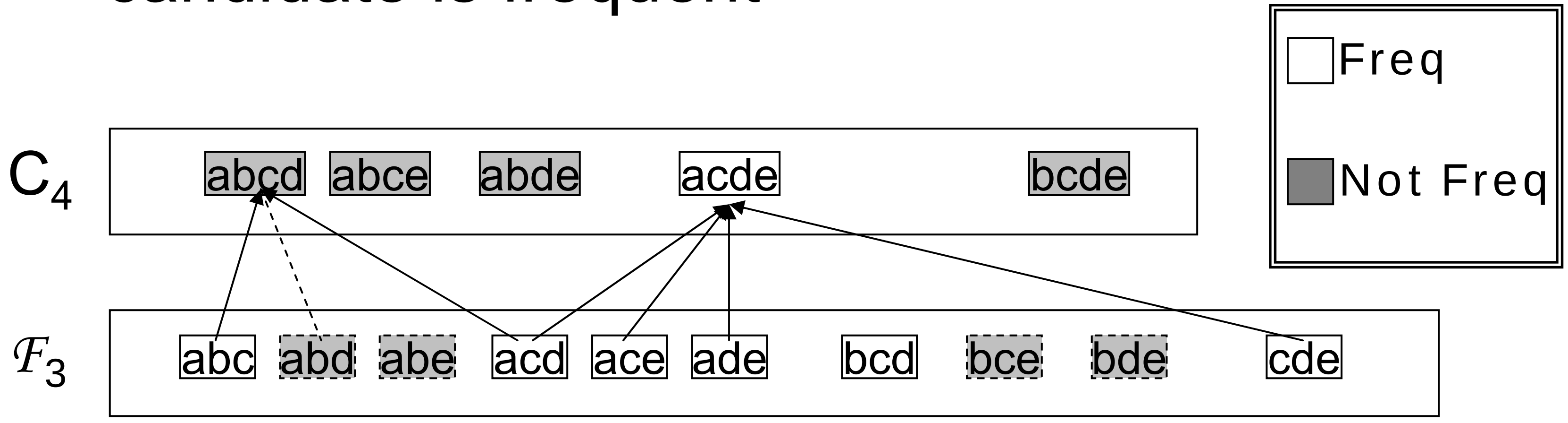
3) Frequency counting - Scan the transactions to count the occurrences of itemsets in C_k;

4) $\mathcal{F}_k = \{ c \in C_K \mid c$ has counts no less than $\#minSup \}$

Return $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots\ldots \cup \mathcal{F}_k$ (= $\mathcal{F}$)

Itemsets: Candidate Generation

- From $\mathcal{F}_{k-1}$ to C_k
  - Join: combine frequent (k-1)-itemsets to form candidate k-itemsets
  - Prune: ensure every size (k-1) subset of a candidate is frequent



Talk Outline

- ***Motivation for Privacy-Preserving Distributed Data Mining***
  - ***Overview of association rules***

  Overview of Previous techniques(Clifton et al)
  - Secure Multi-party computation
  - Horizontal Association Rules
  - Vertical Association Rules
- ***Our technique – Vertical association Rules***
  - Two Party Algorithm
  - Multi-party Algorithm
  - Analysis and comparison to Clifton’s
- ***Conclusions***

Secure Multiparty Computation *It can be done!*

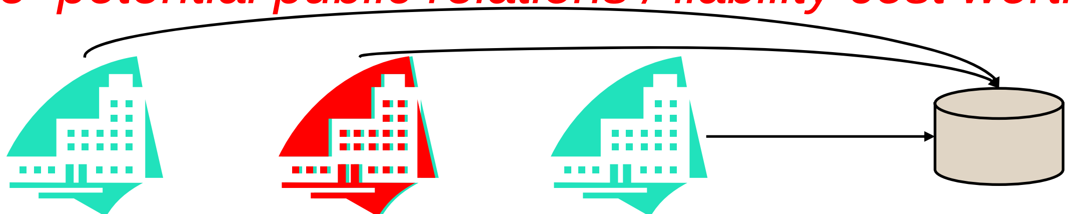
- Goal: Compute function when each party has some of the inputs
- Yao's Millionaire's problem *(Yao '86)*
  - Secure computation possible if function can be represented as a circuit
- Works for multiple parties as well *(Goldreich, Micali, and Wigderson '87)*

Why aren't we done?

- Secure Multiparty Computation is possible
  - But is it practical?
- Circuit evaluation: Build a circuit that represents the computation
  - For all possible inputs
  - Impossibly large for typical data mining tasks
- The next step: *Efficient techniques*

Association Rule Mining: Horizontal Partitioning

- Distributed Association Rule Mining: Easy without sharing the individual data *[Cheung+'96]* *(Exchanging support counts & database sizes)*
- What if we do not want to reveal *<u>which rule is supported at which site</u>*, 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
    - *Is the potential public relations / liability cost worth it?*



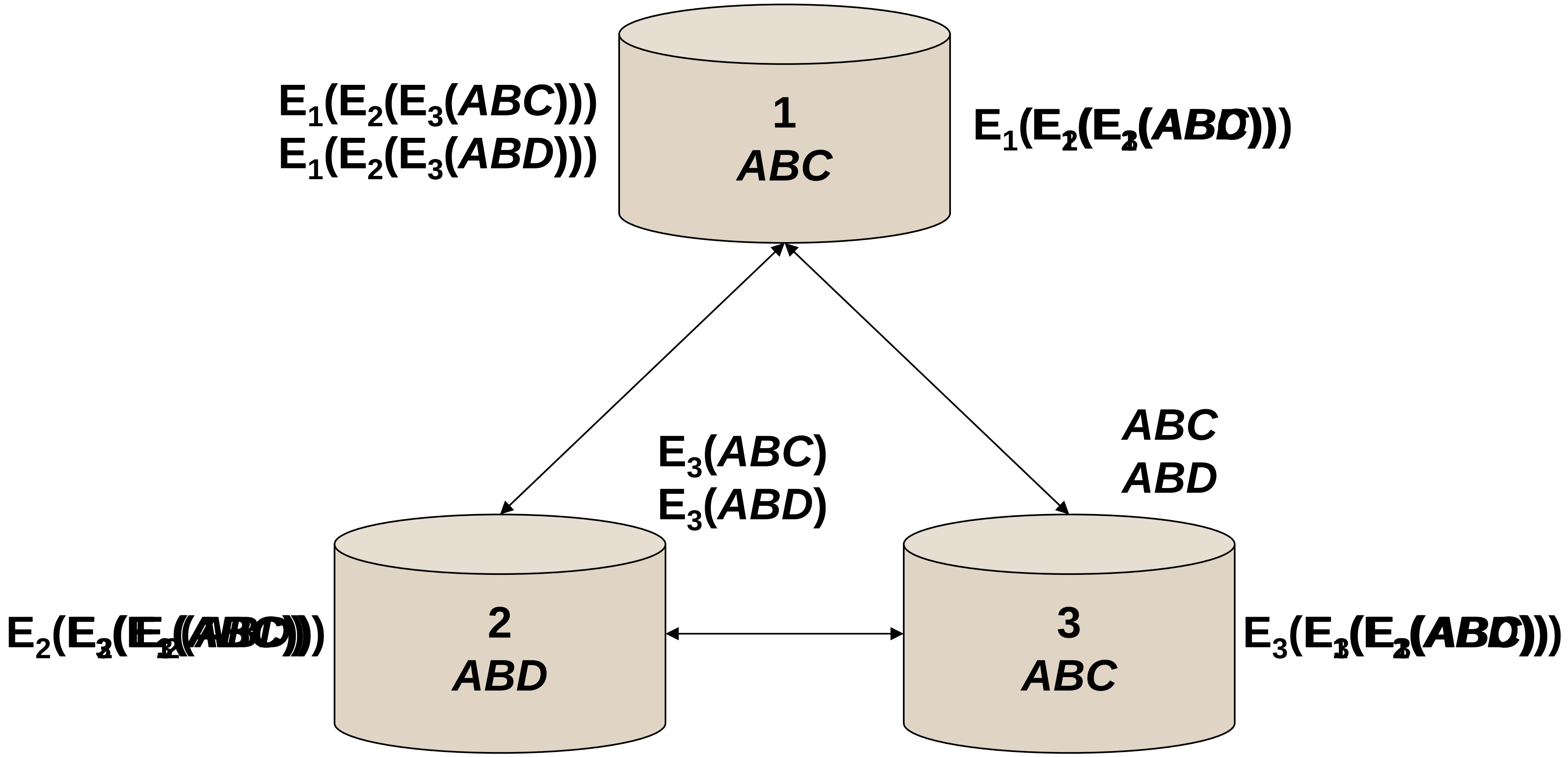
Overview of the Method *(Kantarcioglu and Clifton '02)*

- Find the union of the locally large candidate itemsets securely (a large itemset must be large in at least one local database)
- 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

- Goal: Don’t disclose who is frequent where, just collect all candidates
- Key: Commutative Encryption
  - $E_a(E_b(x) = E_b(E_a(x))$
- Compute local (large) candidate set
- Encrypt and send to next site
  - Continue until all sites have encrypted all itemsets
- Eliminate duplicates
  - Commutative encryption ensures if itemsets the same, encrypted itemsets the same, regardless of order
- Each site decrypts
  - After all sites have decrypted, itemsets left
  - So now each site has all itemsets which are large in at least one site without knowing which site it is

Computing Candidate Sets



Compute Which Candidates Are Globally Supported?

- Goal: To check whether
  X.sup $\geq s * \sum_{i=1}^{n} |DB_i|$ (1)

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

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

Note that checking inequality (1) is equivalent to checking inequality (3)

Which Candidates Are Globally Supported? (Continued)

- Securely compute Sum ≥ 0:
  - Site$_0$ generates random *R*

    Sends $R + count_0$ - $frequency * dbsize_0$ to site$_1$
  - Site$_k$ adds $count_k$ - $frequency * dbsize_k$, sends to site$_{k+1}$
- Is sum at site$_n$ - $R \geq 0$?
  - Use Secure Two-Party Comparison between Site$_n$ and Site$_{k+1}$ *(basically Millionaire problem)*

Association Rules in Vertically Partitioned Data

- Two parties – Alice (A) and Bob (B)
- Same set of entities (Same transaction IDs, e.g. same people)
- A has *p* attributes, $A_1 \dots A_p$
- B has *q* attributes, $B_1 \dots B_q$
- Total number of transactions, *n*
- Support Threshold, *k*

DVD	Digital Camera	USB	*John Grisham*	*Dan Brown*	*Clancey*	*Asimov*

Vertically Partitioned Data *(Vaidya and Clifton '02)*

- Learn globally valid association rules
- Prevent disclosure of individual relationships
  - Join key revealed
  - Universe of attribute values revealed
- Many real-world examples
  - Ford / Firestone
  - FBI / IRS
  - Medical records

Basic idea

- Find out if itemset $\{A_1, B_1\}$ is frequent (i.e., If support of $\{A_1, B_1\} \geq k$)

A

Key	A_1
k_1	1
k_2	0
k_3	0
k_4	1
k_5	1

B

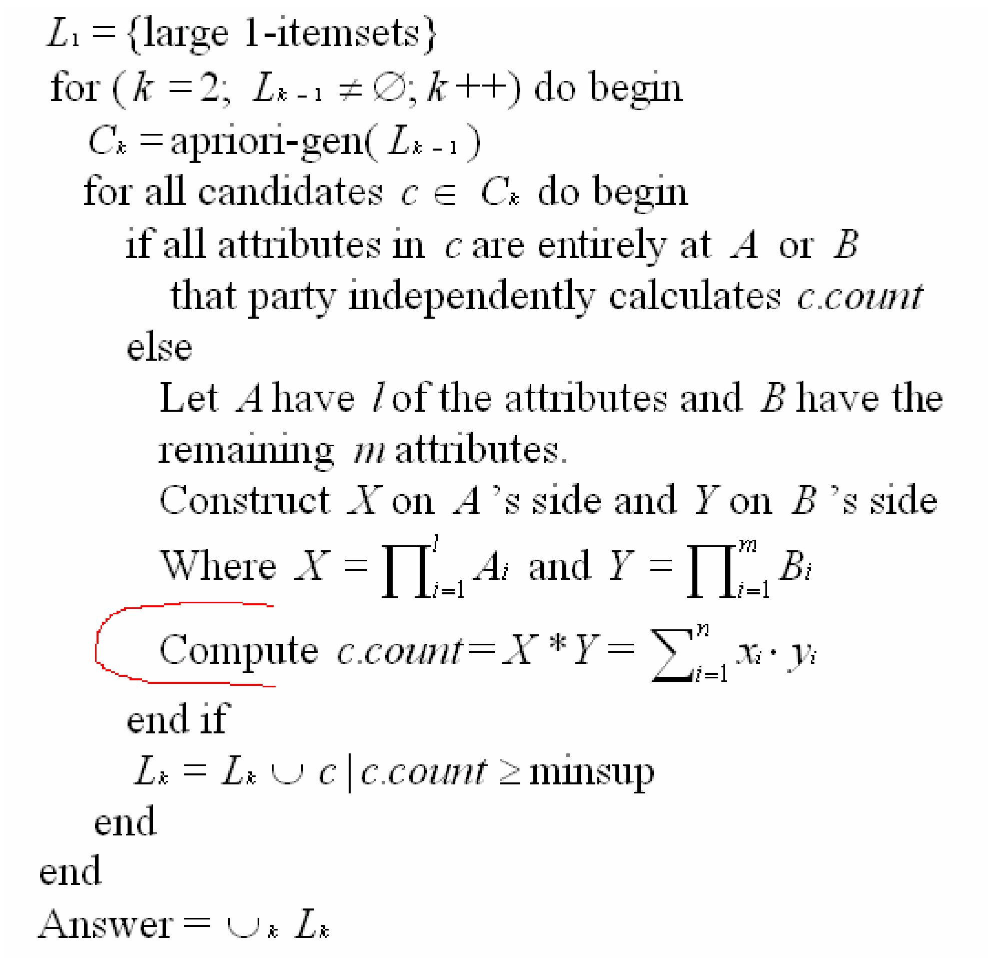
Key	B_1
k_1	0
k_2	1
k_3	0
k_4	1
k_5	1

- Support of itemset is defined as number of transactions in which all attributes of the itemset are present
- For binary data, support $=|A_i \wedge B_i|$. (i.e. the size of the scalar product)

Basic idea

- Thus, $$Support = \sum_{i=1}^{n} A_i \times B_i$$
- This is the scalar (dot) product of two vectors
- To find out if an arbitrary (shared) itemset is frequent, create a vector on each side consisting of the component multiplication of all attribute vectors on that side (contained in the itemset)
- E.g., to find out if $\{A_1, A_3, A_5, B_2, B_3\}$ is frequent
  - A forms the vector X = $\prod A_1\, A_3\, A_5$
  - B forms the vector Y = $\prod B_2\, B_3$
  - Securely compute the dot product of X and Y
- Note, at each step both the itemset and its global support is known to both sides!

VDC - The algorithm



```
$L_1$ = {large 1-itemsets}
for ($k = 2$; $L_{k-1} \neq \emptyset$; $k$++) do begin
  $C_k$ = apriori-gen( $L_{k-1}$ )
  for all candidates $c \in C_k$ do begin
    if all attributes in $c$ are entirely at $A$ or $B$
       that party independently calculates $c.count$
    else
      Let $A$ have $l$ of the attributes and $B$ have the
      remaining $m$ attributes.
      Construct $X$ on $A$'s side and $Y$ on $B$'s side
      Where $X = \prod_{i=1}^{l} A_i$ and $Y = \prod_{i=1}^{m} B_i$
      Compute $c.count = X * Y = \sum_{i=1}^{n} x_i \cdot y_i$
    end if
    $L_k = L_k \cup c \mid c.count \geq$ minsup
  end
end
Answer = $\cup_k L_k$
```

Secure Scalar Product

- A generates $n/2$ randoms, $R_1 \ldots R_{n/2}$
- A sends the following n values to B

$$\left\langle x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n/2} * R_{n/2} \right\rangle$$

$$\left\langle x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n/2} * R_{n/2} \right\rangle$$

$$\vdots$$

$$\left\langle x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n/2} * R_{n/2} \right\rangle$$

- The $(n^2/2)$ $a_{i,j}$ values are known to both A and B
- Continue – see paper…

Security Analysis

- Security based on the premise of revealing less equations than the number of unknowns – possible solutions infinite!
- Just from the protocol, nothing can be found out
- Everything is revealed *only* when about half the values are revealed
- **Note, however, Itemset is known and its support value is broadcasted to all!**
- **Similar situation in the N-parties algorithm**

VDC - Disclosed information

Let's **A** be any set of attributes on the first database (Master).
Let's **B** be any set of attributes on the second database (Slave).

Let's g_{AB} be the size of the global support,

l_A be the size of the local Master's support and

l_B be the size of the local Slave's support.

Let's $P(A)_B$ be the probability that Slave will learn

that a set of attributes in the Master's database (A)
have a given property (B). Note that :

$$P(A)_B = \frac{g_{AB}}{l_B}$$

Disclosed information(cont)

If $g_{AB} = 0 \rightarrow P(A)_B = 0$

If $g_{AB} = l_B \rightarrow P(A)_B = 1$

- This means that in some cases knowing the global support discloses full information on the transactions containing the itemset
- Also large amount of information can be disclosed by using intersection of such B’s with some other sets - A more detailed analysis later…
- This motivated our work…

Talk Outline

- ***Motivation for Privacy-Preserving Distributed Data Mining***

- ***Overview of Previous techniques(Clifton et al)***
  - Secure Multi-party computation
  - Horizontal Association Rules
  - Vertical Association Rules
- ***Our technique – Vertical association Rules***
  - Two Party Algorithm
  - Multi-party Algorithm
  - Analysis and comparison to Clifton's
- ***Conclusions***

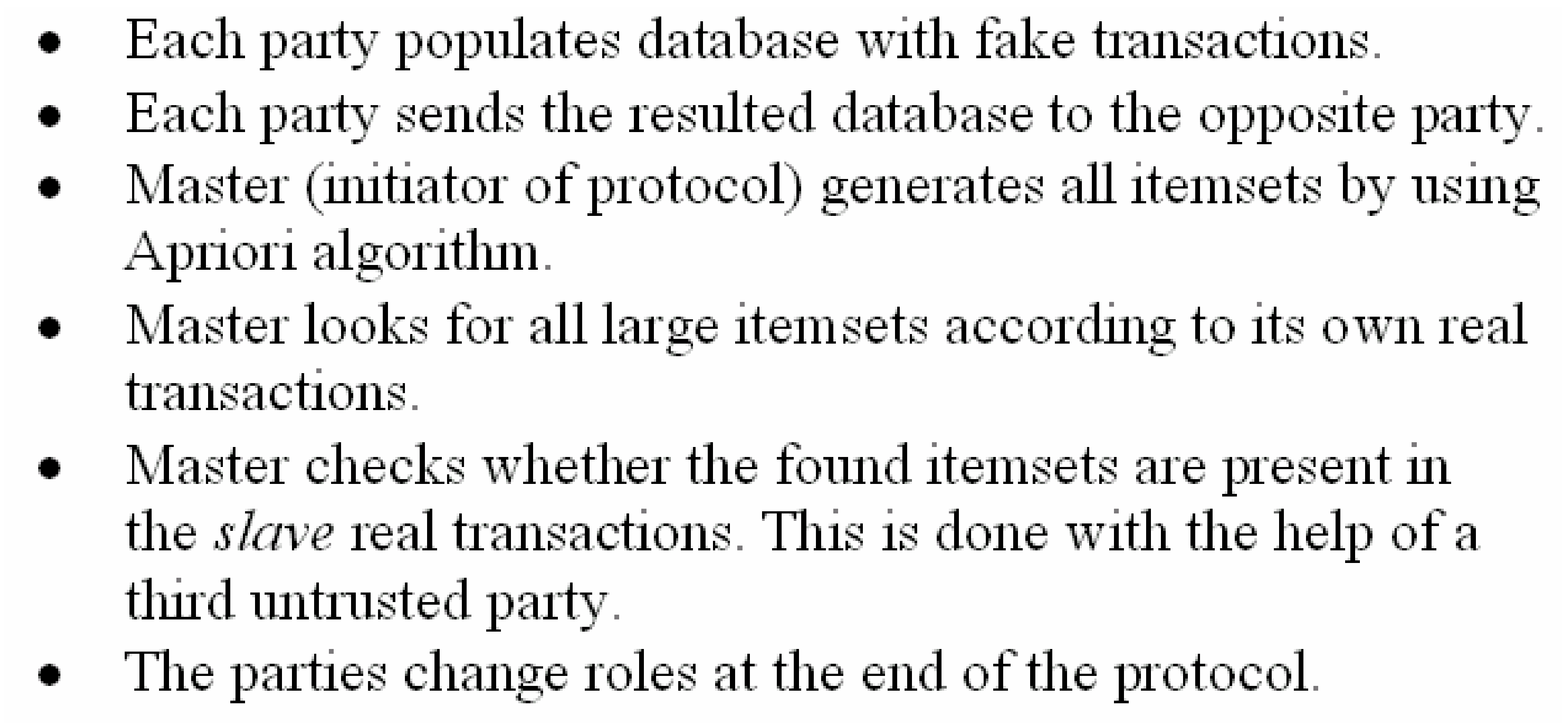
Association Rules in Vertically Partitioned Data

- Two parties – Alice (A) and Bob (B)
- Same set of entities and a common unique ID domain
- Total number of transactions, *n* with overlap in their unique Ids (sufficient to do mining!), but domain of Id is much larger
- A has *p* attributes, $A_1 \dots A_p$
- B has *q* attributes, $B_1 \dots B_q$
- Support Threshold, *k*
- *The problem is to find all frequent item sets (and rules later)*

Our model assumptions

- In any site there is no external information about any other database.
- There is no collusion between parties.
- The various parties follow the protocol honestly. They may try to use a correct protocol to infer information, but we shall show that this will not be helpful for them The parties may store intermediate or final results. (i.e. semi-honest behaviour)
- The analysis of privacy is in terms of what can be inferred from those stored results only! i.e. a semi-honest model
- **First appeared in IFIP WG11.3 2003, journal version in DKE Nov2006**

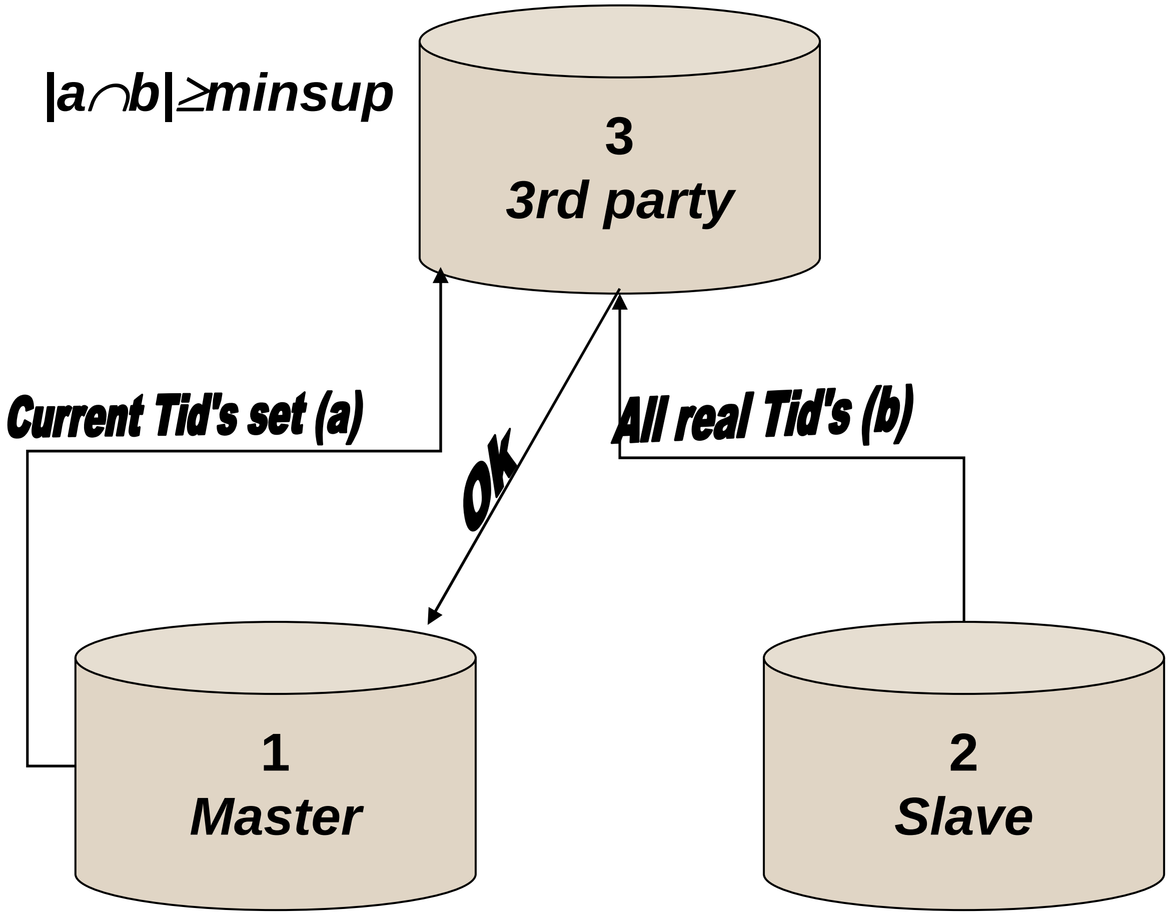
Our algorithm
Two-party - basic ideas



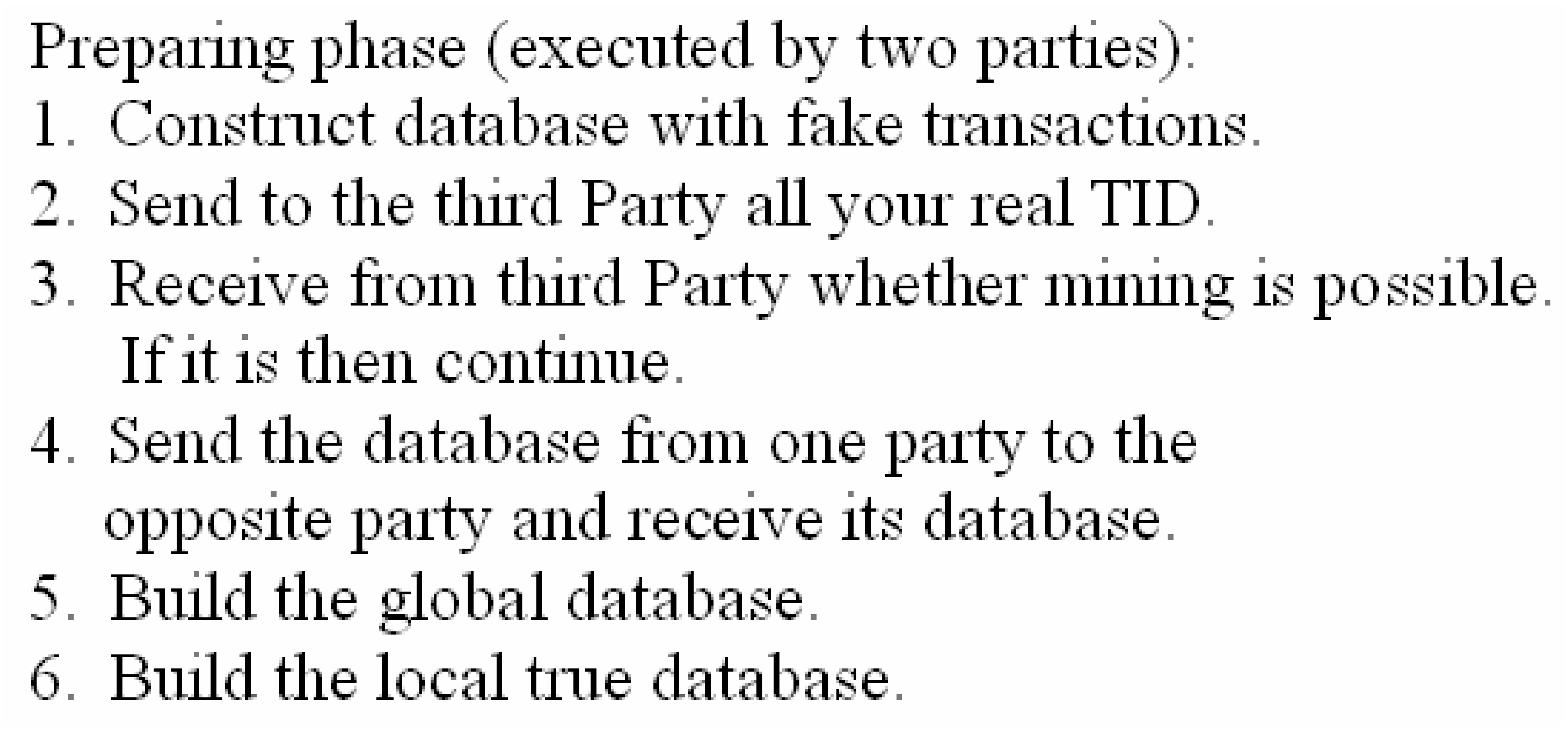
- Each party populates database with fake transactions.
- Each party sends the resulted database to the opposite party.
- Master (initiator of protocol) generates all itemsets by using Apriori algorithm.
- Master looks for all large itemsets according to its own real transactions.
- Master checks whether the found itemsets are present in the *slave* real transactions. This is done with the help of a third untrusted party.
- The parties change roles at the end of the protocol.

•Note, only False positives are possible!

Explanation



The Two-party algorithm



Preparing phase (executed by two parties):

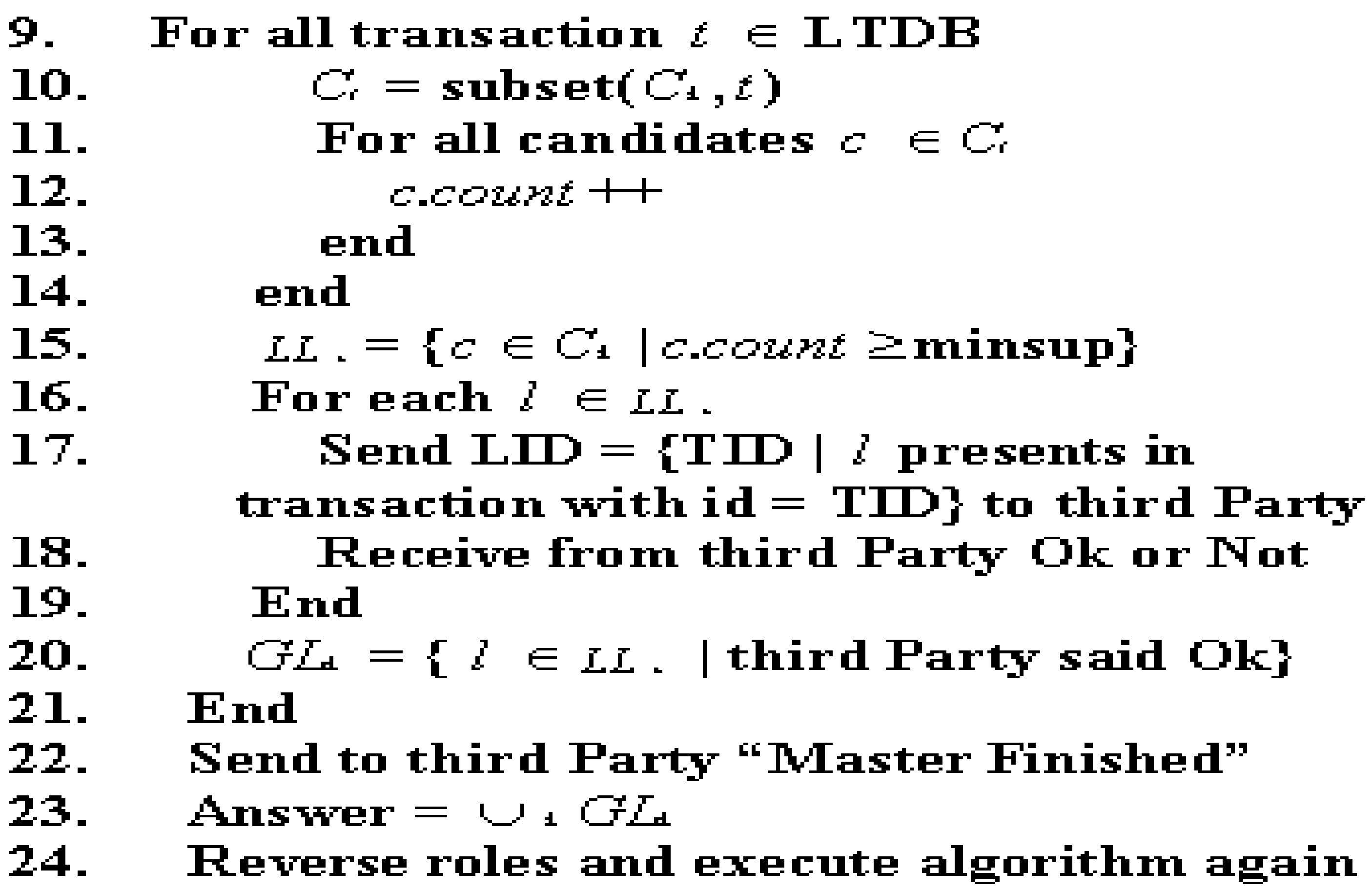
1. Construct database with fake transactions.
2. Send to the third Party all your real TID.
3. Receive from third Party whether mining is possible.
   If it is then continue.
4. Send the database from one party to the
   opposite party and receive its database.
5. Build the global database.
6. Build the local true database.

Two-party algorithm(cont)

Master's Execution phase:

```
1. Find $LL_1$ – all large 1-itemsets in the local database from 6.
2. For each $l \in LL_1$:
3.     Send LID = {TID | $l$ present in transaction
         with id = TID} to third Party
4.     Receive from third Party Ok or Not
5. End
6. $GL_1 = \{l \in LL_1$ | third Party said Ok}
7. For ($k = 2$; $GL_{k-1} \neq \emptyset$; $k$++)
8.     $C_k$ = apriori-gen($GL_{k-1}$)
```

Two-party algorithm(cont)



```
-> 
   9.   For all transaction $t \in$ LTDB
   10.        $C_t$ = subset($C_k$, $t$)
   11.        For all candidates $c \in C_t$
   12.            $c.count$++
   13.        end
   14.     end
   15.     $LL_k = \{c \in C_k \mid c.count \geq$ minsup$\}$
   16.     For each $l \in LL_k$
-> 17.        Send LID = {TID | $l$ presents in
           transaction with id = TID} to third Party
-> 18.        Receive from third Party Ok or Not
-> 19.     End
-> 20.     $GL_k = \{ l \in LL_k \mid$ third Party said Ok$\}$
   21.   End
   22.   Send to third Party "Master Finished"
   23.   Answer = $\cup_k GL_k$
   24.   Reverse roles and execute algorithm again
```

Two-party algorithm(cont)

Third Party Execution:

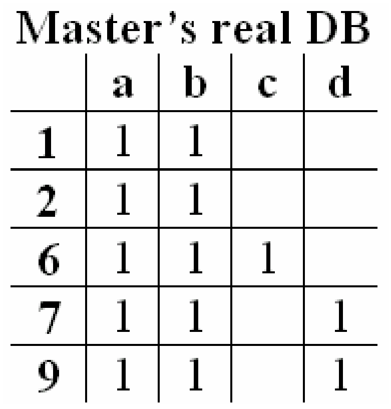
1. Check initial condition whether mining is at all possible (overlap of IDs is sufficient)
2. For each set of IDs sent by the Master (itemset is not known) compute real set size and return OK or NOT-OK

Two-party algorithm(cont)

Slave execution phase:

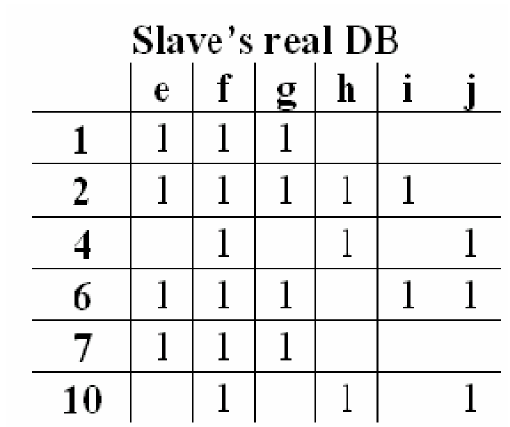
1. Execute preparing phase.
2. Wait for Master to finish.
3. Accept results from Master (with trust) or
   reverse roles and run algorithm again (without trust) .

Two-party algorithm - Example



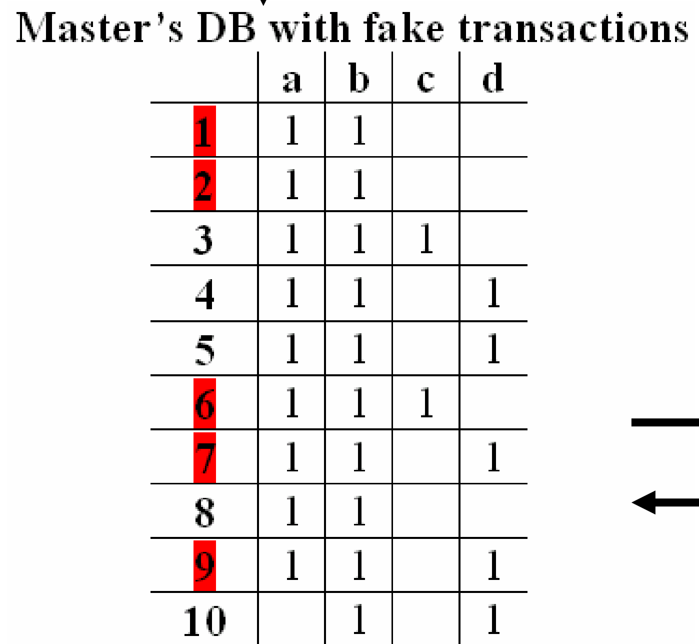
Master's real DB

	a	b	c	d
1	1	1		
2	1	1		
6	1	1	1	
7	1	1		1
9	1	1		1



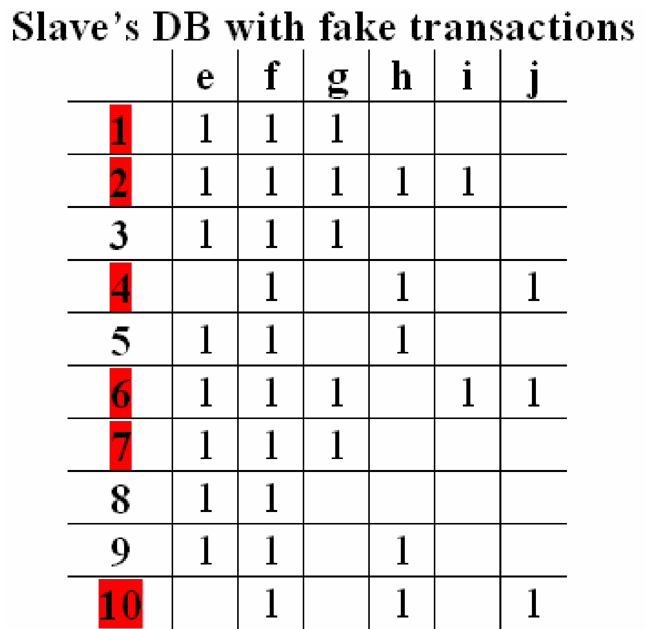
Slave's real DB

	e	f	g	h	i	j
1	1	1	1			
2	1	1	1	1	1	
4		1		1		1
6	1	1	1		1	1
7	1	1	1			
10		1		1		1



Master's DB with fake transactions

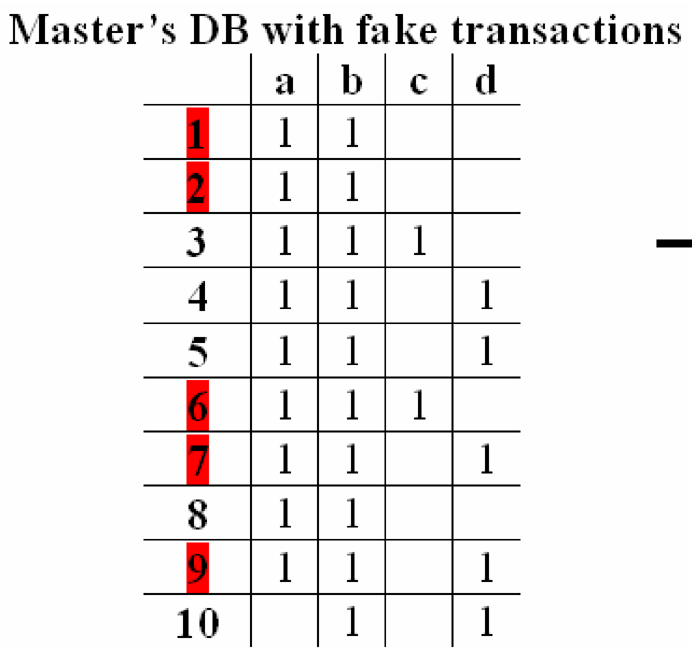
	a	b	c	d
1	1	1		
2	1	1		
3	1	1	1	
4	1	1		1
5	1	1		1
6	1	1	1	
7	1	1		1
8	1	1		
9	1	1		1
10		1		1



Slave's DB with fake transactions

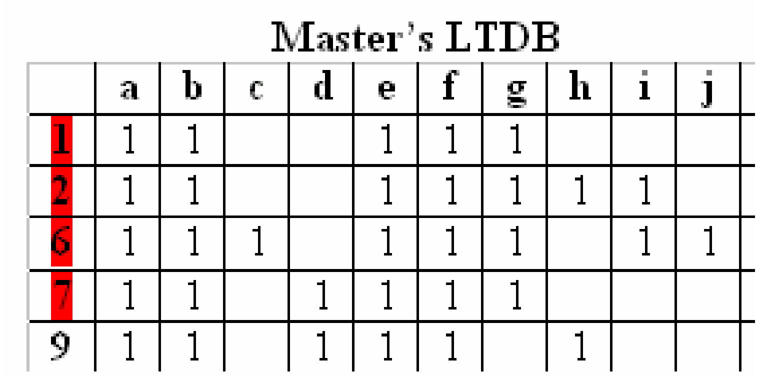
	e	f	g	h	i	j
1	1	1	1			
2	1	1	1	1	1	
3	1	1	1			
4		1		1		1
5	1	1		1		
6	1	1	1		1	1
7	1	1	1			
8	1	1				
9	1	1		1		
10		1		1		1

Two-party algorithm(cont)



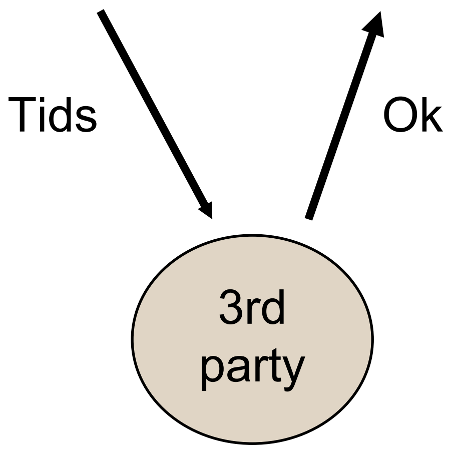
Master's DB with fake transactions

	a	b	c	d
1	1	1		
2	1	1		
3	1	1	1	
4	1	1		1
5	1	1		1
6	1	1	1	
7	1	1		1
8	1	1		
9	1	1		1
10		1		1



Master's LTDB

	a	b	c	d	e	f	g	h	i	j
1	1	1			1	1	1			
2	1	1			1	1	1	1	1	
6	1	1	1		1	1	1		1	1
7	1	1		1	1	1	1			
9	1	1		1	1	1		1		



Secure Computation

- Instead of a Third party – use Secure computation.
- Atallah and Du proposed a more efficient technique for Two-Party Scalar Product computation.
- We use a modified version of this protocol in our method

M.J.Atallah and W.Du Scalar Product Protocol

Inputs:

Alice has a vector $X = (x_1, x_2, \ldots, x_n)$
Bob has a vector $Y = (y_1, y_2, \ldots, y_n)$**.**

Outputs:

Alice (but not Bob) gets $X \bullet Y + v$
v **is random scalar known to Bob only.**

(details of algorithm in the paper)

Note, in our algorithms Alice is the Master and Bob is the Slave,
Therefore the Master does not know the value of support, only
The OK/NOT-OK returned by the slave.
The slave doesn't know the itemset, since its vector is All real trans-IDs

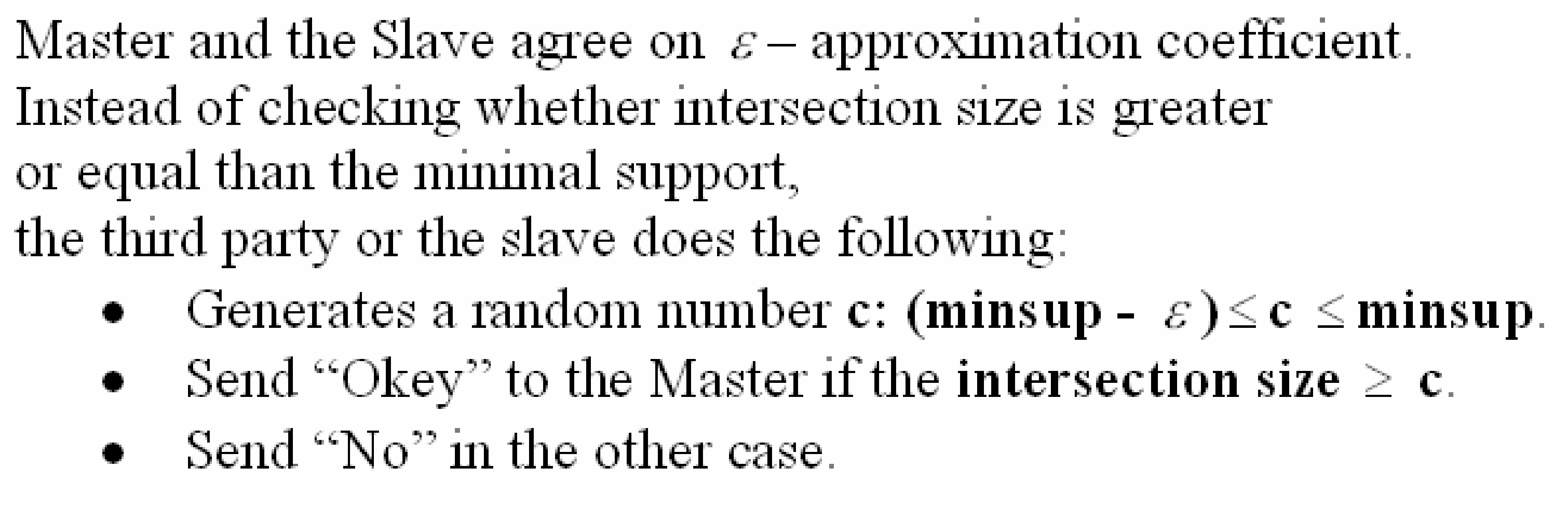
Advantages of the first algorithm

- **Performance -** Computation is done only for the itemsets with enough local support using the assumption that fake transactions only add “1”s… (although there is pre-processing step…)
- **Privacy** - The slave who knows the support value does not know to which item set it belongs. The master does not get the support value, just OK/Not OK

Problem of the first algorithm - Probing

- Assume that the minimal support threshold is 4. The Master sends to the trusted party sets of exactly four TIDs until it receives an "OK" answer, which means all four TIDs are not fake. Then it chooses three of these, and for every other TID j it sends to the trusted party a set containing these three TIDs together with j. The answer of the trusted party is "OK" if and only if j is not a fake TID!
- Solution – The **support approximation** method

Support approximation method



Master and the Slave agree on ε – approximation coefficient.
Instead of checking whether intersection size is greater
or equal than the minimal support,
the third party or the slave does the following:

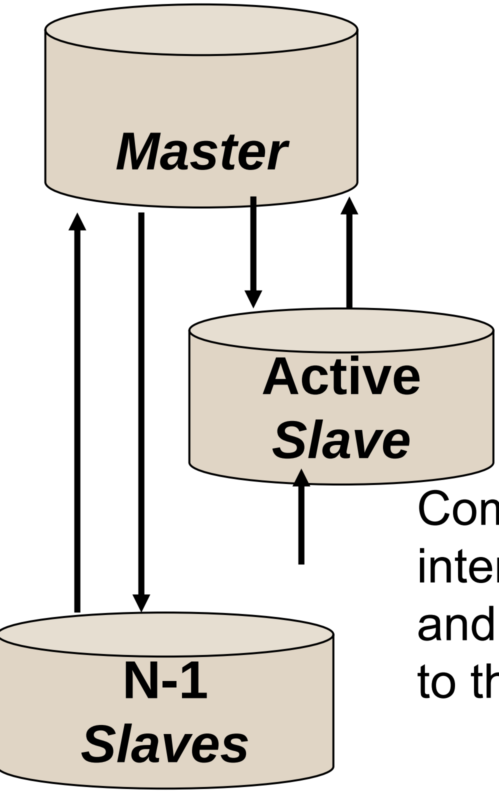
- Generates a random number **c: (minsup -** ε **)** $\leq$ **c** $\leq$ **minsup**.
- Send “Okey” to the Master if the **intersection size** $\geq$ **c**.
- Send “No” in the other case.

Therefore the Master cannot use probing since the exact value of support is not known!

2nd algorithm - Three or More Parties

- We assume that we have one Master and n Slaves.
- We do not use a third party.
- We do not use secure computation.
- Each Slave computes the intersection itself.
- The Master starts the computation with the first Slave and waits for the last Slave for a positive or negative result.

2nd algorithm - Three or More Parties



Receive DBs with fake transactions

Build DB with own real TIDs

Use Aprioiri to find all frequent itemsets (*L*)

For each *l* $\in$ *L* build binary vector $X = (x_1, x_2, \ldots, x_n)$

Send to relevant Slaves new π and new *R*

Fix one *Slave* that does not have currently checked attributes (Active Slave)

Send to Active Slave π ***(X+R)***

All relevant Slave *i* sends to the Active Slave π **(X_i +R),**

$X_i = (x_1, x_2, \ldots, x_n)$ such that $x_j = 1$ if transaction with $ID = j$ is a real transaction of the Slave $(1 \le j \le n)$.

Compute intersection size and send "Yes/No" to the Master.

Computing confidence

- First, find all frequent itemsets.
- For each such set *Z*, Master generates all possible rules of the form *X->Y*, such that *Z={X,Y}*.
- For each such rule, Master generates two sets of ids: *TIDx* and *TIDxy* and sends them to the Third Party/Slaves.
- Third Party/Slaves calculates $\frac{TIDxy \cap STID}{TIDx \cap STID}$ and sends "*OK*" if result > c(minimal confidence value) or "*NOT*" otherwise.
- At the end of the execution, Master receives from the Third Party/ Slave "*OK*" or "*NOT*" – that determines whether *X->Y* is rule or not.

Communication Cost

Algorithm/Protocol name	Number of messages for each party	Size of 1 message
Modified Scalar Product Protocol	*p*m* messages	*N values*
2-party frequent itemsets mining with Third Party	*C* - the maximal number of item sets tested by the Apriori algorithm.	*N values*
2-party frequent itemsets mining with Secure Computation	*C* * Communication cost of scalar product protocol (*p*m*)	*N values*
n-party frequent itemsets mining	*C* - the maximal number of item sets tested by the Apriori algorithm.	*N values*
2-party association rules mining	*R* – the maximal number of possible rules.	*2N values*
n-party association rules mining	*R* – the maximal number of possible rules.	*2N values*

Disclosed information - Analysis and Comparison

In VDC/N Master and Slave/s are symmetric.

We will analyze the information disclosed by the Slave but identical analysis is right for the Master.

The main idea of analysis:

- For each itemset, each party knows the support value.
- From this information the Slave learns the probability that an item in the set supported by the Master has a property in the Slave's database, which is computed as the ratio of the global support to the Slave's support, **whether the item set is frequent or not**!

Disclosed information(notation)

Let's **A** be any set of attributes on the first database (Master).
Let's **B** be any set of attributes on the second database (Slave).

Let's g_{AB} be the size of the global support,

l_A be the size of the local Master's support and

l_B be the size of the local Slave's support.

Let's $P(A)_B$ be the probability that Slave will learn

that a set of attributes in the Master's database (A)
have a given property (B). Note that :

$$P(A)_B = \frac{g_{AB}}{l_B}$$

Disclosed information (One support computation)

If $g_{AB} = 0 \text{ -> } P(A)_B = 0$

If $g_{AB} = l_B \text{ -> } P(A)_B = 1$

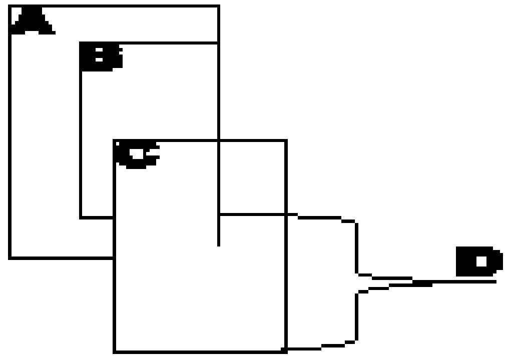
- This means that in some cases knowing the global support discloses full information on the transactions containing the itemset
- Also large amount of information can be disclosed by using intersection of such B’s with some other sets.

Disclosed information(Two or more Support computations)

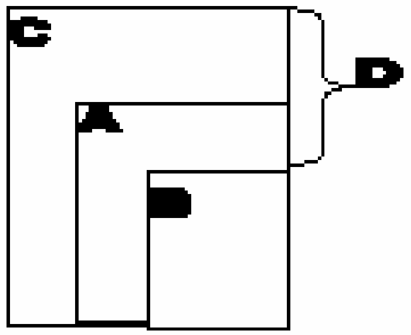
Rule 1: Once Slave knows, that $g_{AB} = l_B \rightarrow P(A)_B = 1$ he knows that:

$\forall T(C) \in STID: [T(C) \cap T(B)] \neq \phi,\ P(A)_D = \frac{g_{AC} - l_{B \cap C}}{l_D}$

where $T(D) = T(C) - [T(C) \cap T(B)]$.



If $T(C) \supset T(B)$, then $P(A)_D = \frac{g_{AC} - l_B}{l_D}$



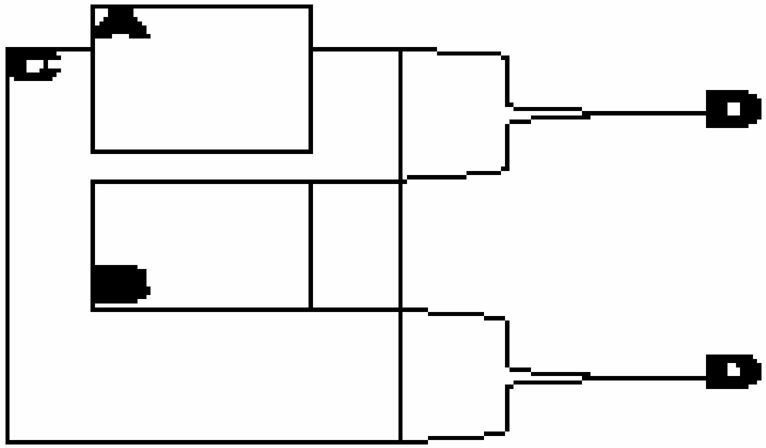
Disclosed information(Two or more Support computations)

Rule 2: Once Slave knows, that

$T(A) \cap T(B) = \phi \Rightarrow g_{AB} = 0 \Rightarrow P(A)_B = 0$, he knows that:

$$\forall\, T(C) \in STID: [T(C) \cap T(B)] \neq \phi,\ P(A)_{B \cap C} = 0;\ P(A)_D = \frac{g_{AC}}{l_D};$$

where $T(D)=T(C)- [T(C) \cap T(B)]$.



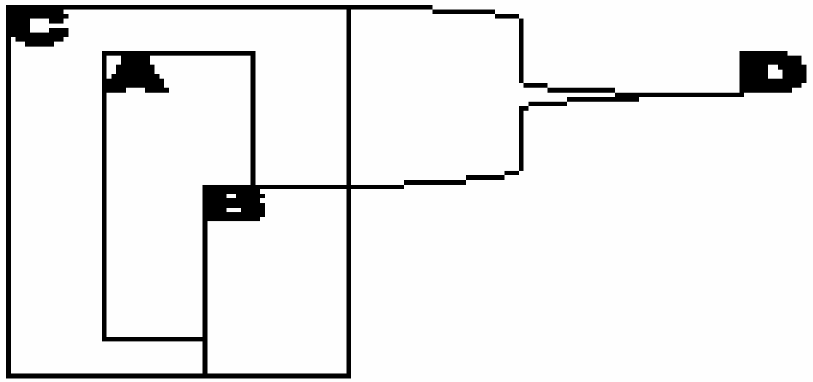
Disclosed information(Two or more Support computations)

Rule 3: Once Slave knows, that

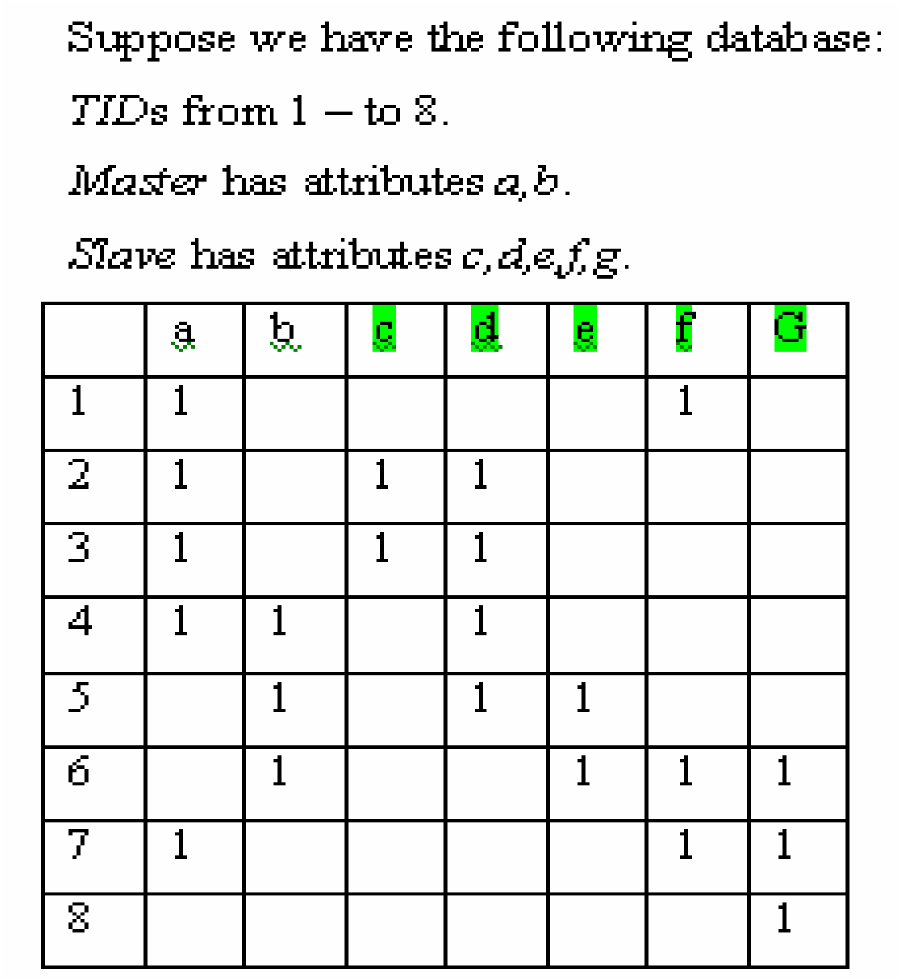
$T(A) \cap T(B) \neq \phi \; (T(B) \not\subset T(A))$, he knows that:

$$\forall\, T(C) \in STID: T(C) \supset T(B),\; P(A)_D = \frac{g_{AC} - g_{AB}}{l_D},$$

where $T(D) = T(C) - T(B)$. Note that if $g_{AC} = g_{AB}$, then $P(A)_D = 0$.



Disclosed information (Example)



Suppose we have the following database:

TIDs from 1 – to 8.

Master has attributes *a,b*.

Slave has attributes *c,d,e,f,g*.

	a	b	c	d	e	f	G
1	1					1	
2	1		1	1			
3	1		1	1			
4	1	1		1			
5		1		1	1		
6		1			1	1	1
7	1					1	1
8							1

1. $g_{ac} = 2 = l_c \Rightarrow P(a)_c = P(a)_{2,3} = 1$

2. $g_{ad} = 3 \Rightarrow P(a)_d = P(a)_{2,3,4,5} = \frac{3}{4}$

2.1 From 1,2 by rule 1: $P(a)_{4,5} = \frac{1}{2}$

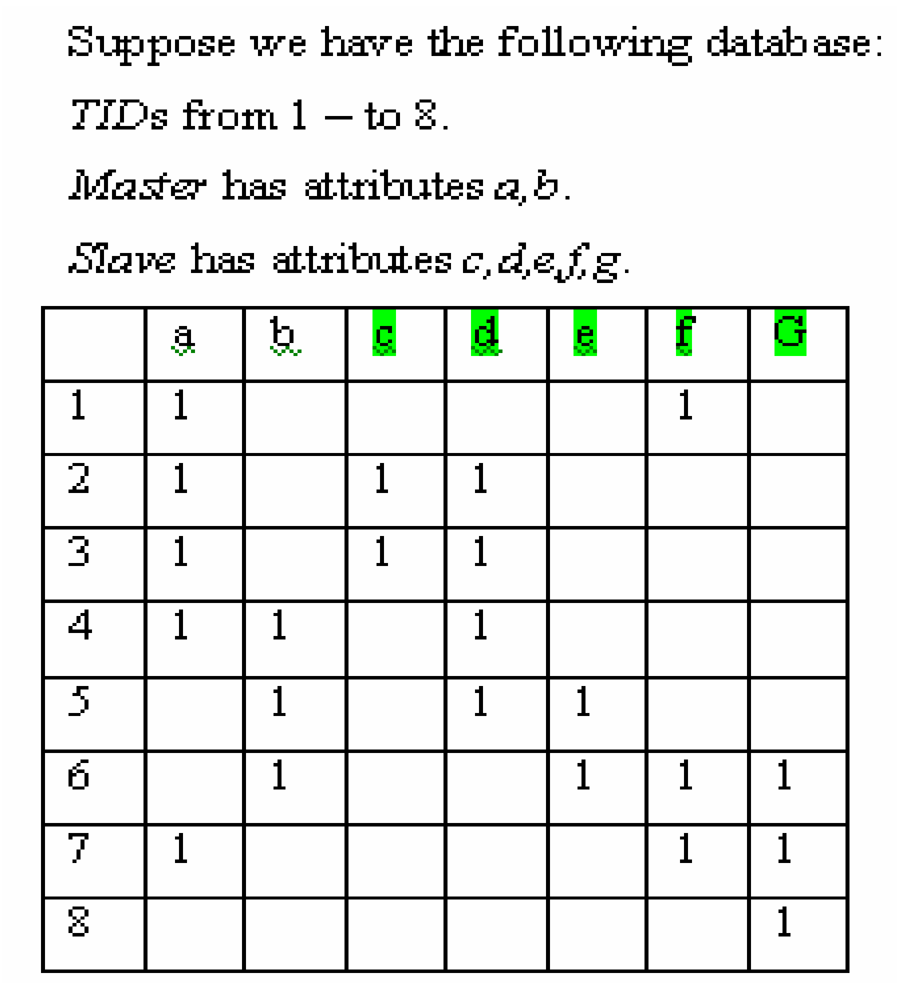
3. $g_{ae} = 0 \Rightarrow P(a)_e = P(a)_{5,6} = 0.$

3.1 From 2.1,3 by rule 2:
$P(a)_4 = 1$ and $P(a)_5 = 0$

4. $g_{af} = 2 \Rightarrow P(a)_f = P(a)_{1,6,7} = \frac{2}{3}$

4.1 From 3,4 by rule 2:
$P(a)_6 = 0$ and $P(a)_{1,7} = 1$

Disclosed information (Example)



Suppose we have the following database:

TIDs from 1 – to 8.

Master has attributes *a*, *b*.

Slave has attributes *c*, *d*, *e*, *f*, *g*.

	a	b	c	d	e	f	G
1	1					1	
2	1		1	1			
3	1		1	1			
4	1	1		1			
5		1		1	1		
6		1			1	1	1
7	1					1	1
8							1

5. $g_{ag} = 1 \Rightarrow P(a)_g = P(a)_{6.7.8} = \frac{1}{3}$

5.1 From 5, 4.1 by rule 2: $P(a)_8 = 0$.

- So, Slave knows the exact distribution of the attribute ***a.***
- Transaction containing attribute ***b*** are also disclosed.

Disclosed information
Our Algorithms

L – a set of all real *TIDs* of the *Master*

$A \subset L$ – any set of real TIDs, l_A = |*A*|

m – minimal support

Q:L->(T,F) – is a function returned by the algorithm

$P(A)_T$ - the probability that *Master* will learn that a transaction $a \in A$ is a real transaction in the *Slave's* database.

Disclosed information (One support computation)

- if *Q(A)=T* => $P(A)_T = \frac{m}{l_A}$

  Note, that if $m = l_A$ => $P(A)_T = 1$ => **full disclosure.** (same as in VDC**)**

- if *Q(A)=F* => $P(A)_T \leq \frac{m-1}{l_A}$ Not exact probability like in VDC!

So in case the support value is below the threshold the information disclosed is much less!

And in case it is above, it is bounded by m / lA!

With Support approximation method

In this case if $Q(A) = T$ then

$$P(A)_T = [\frac{m-\varepsilon}{l_A}; \frac{m}{l_A}]$$

Even if $m = l_A$ we do not have the problem as in the case without ε.

Note that it's enough to set ε to be 1 to eliminate the exact disclosure.

The multiple Inference rules also disclose much less information

Comparison Example (conclusion)

The following tables summarizes the information learned by each side about transactions on the opposite side.

Our algorithm

	Master	Slave
1	1/2	1/2
2	1/3	1/2
3	1/3	1/2
4	1/2	1/2
5	1/2	1/2
6	1/2	1/2
7	1/2	1/2
8	-	-

Methods that reveal exact support (e.g.,VDC)

	a	b	c	d	e	f	g
1	1	0	1/2	1/2	0	2/5	1/5
2	1	0	1/2	1/2	0	2/5	1/5
3	1	0	1/2	1/2	0	2/5	1/5
4	1	1	0	1	0	1/3	1/3
5	0	1	0	1/2	1	1/3	1/3
6	0	1	0	1/2	1	1/3	1/3
7	1	0	1/2	1/2	0	2/5	1/5
8	0	0					

Conclusions and future work

- We presented algorithms for discovering all large item sets in vertically partitioned databases without the sources revealing their individual transaction values.
- We also presented algorithms for computing the resulted association rules
- We analyzed the privacy properties of our algorithms and compared them to Vaydia and Clifton (VDC/N)

Future work

Future work includes experimental evaluation of the probabilities of disclosure in various cases.

The number of types of data mining techniques continues to grow; each new type generates a need for several privacy-preserving data mining algorithms (depending on how data is partitioned, privacy constraints, assumptions on external knowledge, etc.)

New research – not using Secure computation or fake Transactions, instead separate mining and calculation – to appear in IDEAS6