Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Vorlesung

Datenschutz und Privatheit in vernetzten Informationssystemen

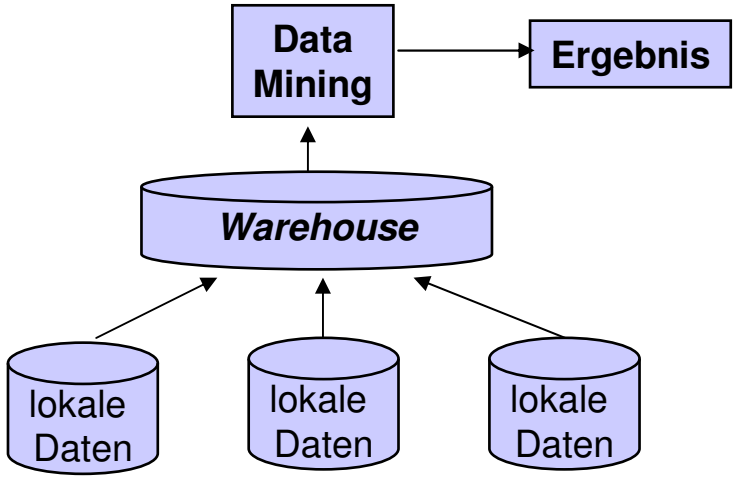
Kapitel 7: Privacy Preserving Data Mining

Thorben Burghardt, Erik Buchmann
buchmann@ipd.uka.de

Thanks to Chris Clifton & Group

Motivation

- Data-Mining identifiziert interessante Muster und Trends in Datenbeständen.
- Dabei wird oftmals Data-Warehousing und Data-Mining komibiniert.
- Zweistufiges Vorgehen
  - Zusammenführen aller Informationen an einer Stelle (Warehouse)
  - Anwenden des Data-Mining Algorithmus auf diese Daten.



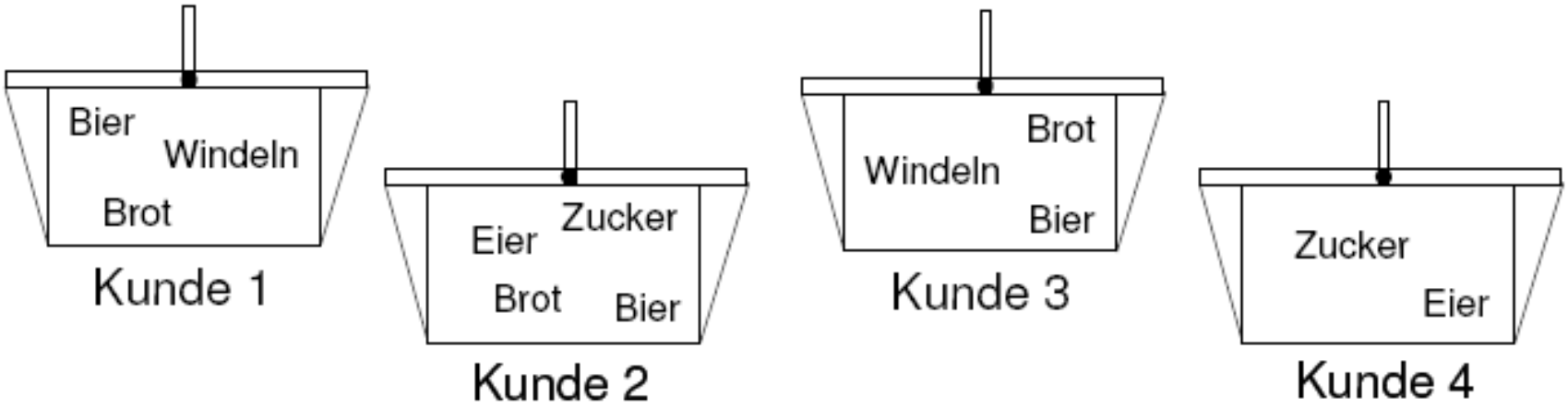
Einschub Data-Mining Grundlagen

Dank an Frank Eichinger, Matthias Brach und Stephan Schosser

Motivation - Warenkorbanalyse

- Gesucht: Einkaufsgewohnheiten
  - Höhere Kundenzufriedenheit durch günstige Anordnung
  - Höherer Absatz durch ungünstige Anordnung

- Warenkörbe (Beispiel)



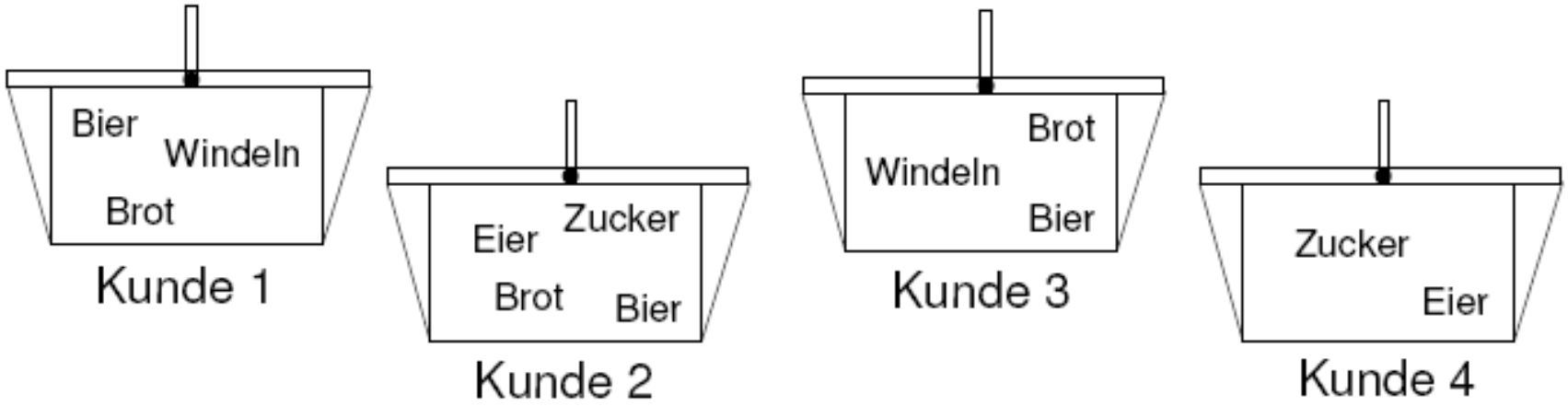
- Fragestellung: Welche Kombinationen werden häufig gekauft (Frequent Itemsets)?

Assoziationsregeln

- Darstellung von Assoziationsregeln
  - Antecedent X ⇨ Consequent Y
- Wahrscheinlichkeitsbasierter Charakter
  - Consequent Y ist mit der Wahrscheinlichkeit P wahr,
  - … wenn der Antecedent X wahr ist
  - Bedingte Wahrscheinlichkeit P(Y | X)!
- Zugelassene Wertebereiche
  - Besonders geeignet für kategorische Daten
  - Möglichkeit Grenzwerte für kontinuierliche Werte zu setzen

Motivation – Warenkorbanalyse II

- Warenkörbe



- Frequent Itemsets (mit mind. 2 Items)
  - {Brot, Bier}, {Brot, Bier, Windeln}, {Zucker, Eier}, {Bier, Windeln}, {Brot, Windeln}
- Wie lassen sich aus Frequent Itemsets Assoziationsregeln ableiten?
  - Beispiel: Wer Windeln kauft, kauft auch Brot

Wichtige Begriffe - Support

- Alternative Namen
  - Häufigkeit, Abdeckung

- Angabe bezüglich der Häufigkeit eines Portfolios
- Anzahl bzw. Anteil der Transaktionen, die $X \cup Y$ enthalten
- Formal: $P(X \cup Y)$

- Beispiel:
  - Die Kombination Windeln, Bier tritt in 50% der Warenkörbe auf.
  - Support = 50%

Wichtige Begriffe – Confidence

- Alternative Namen
  - Genauigkeit

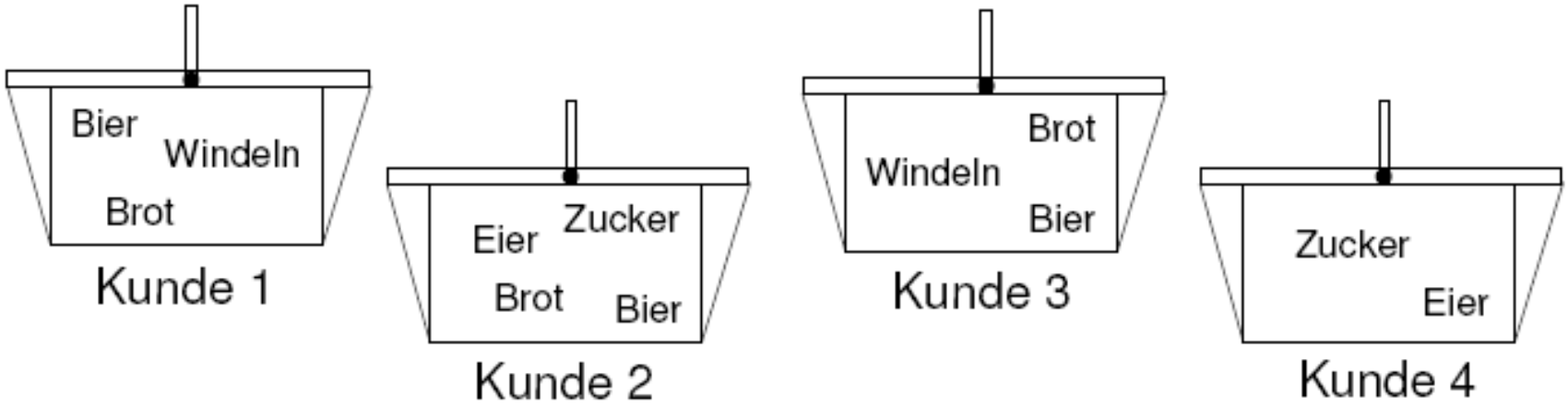
- „Überraschungsmass“
- Wenn eine Transaktion X enthält, dann auch Y (mit gegebener Genauigkeit)
- Formal: $P(Y \mid X) = |X \cup Y| \; / \; |X|$

- Beispiel:
  - Wenn Windeln gekauft wurden, wurde in 100% aller Fälle auch Bier gekauft
  - Confidence = 100%

- Ziel: Finden von Regeln mit
  - … hohem Support (support > minSup) und …
  - … hoher Confidence (confidence > minConf)

Beispiel - Warenkorbanalyse

- Warenkörbe



- Frequent Itemsets (minSup = ½)
  - {Brot, Bier} (Support = ¾); {Brot, Bier, Windeln} (Support = ½); {Zucker, Eier} (Support = ½); {Bier, Windeln} (Support = ½); {Brot, Windeln} (Support = ½)
- Assoziationsregeln (Auswahl)
  - Brot ⇒ Bier (Confidence = 100%)
  - Brot, Bier ⇒ Windeln (Confidence = 67%)
  - Zucker ⇒ Eier (Confidence = 100%)

A-Priori Eigenschaft

- Itemset häufig, wenn Supermenge häufig
- Beispiel:
  {Bier, Windeln, Brot} häufig
  ⇒ {Bier, Windeln}, {Bier, Brot}, {Windeln, Brot} und {Bier}, {Windeln}, {Brot} häufig
- Itemset kann nur häufig sein, …
  … wenn alle Teilmengen häufig
- Dadurch:
  - Bestimmung von Frequent Itemset Kandidaten mit n Elementen aus solchen mit (n - 1) Elementen möglich

A-Priori Algorithmus – Frequent Itemsets

Finden aller Itemsets mit ausreichendem Support:

- Beginn mit ein-elementigen Sets (1)-Sets:
  - einfaches Abzählen
- Berechnung der k-Sets aus den (k-1)-Sets:
  - Join-Step: Ermittlung von Kandidaten;
    Aus A-Priori Eigenschaft:
    Alle (k-1)-elementigen Teilmengen
    eines k-Sets sind (k-1)-Sets,
  - Prune-Step: Löschen aller Kandidaten,
    die eine „unzulässige“
    (k-1)-elementige Teilmenge haben.
  - Support Counting, d. h. Abzählen,
    wie häufig die Kandidaten wirklich sind.

A-Priori – Frequent Itemset (Bsp.) I

- Beispieltupel:
  - {A, B, E}
  - {B, D}
  - {B, C}
  - {A, B, D}
  - {A, C, D}
  - {B, C}
  - {A, C}
  - {A, B, C, E}
  - {A, B, C}
- MinSup: 2/9,
  - d.h. Itemset ist häufig, wenn 2 Tupel es enthalten

A-Priori – Frequent Itemset (Bsp.) II

- Ein-elementige Frequent Itemsets
  - {A}: 6
  - {B}: 7
  - {C}: 6
  - {D}: 3
  - {E}: 2
- D.h. Alle Items sind häufig!

Beispieltupel:
{A, B, E}
{B, D}
{B, C}
{A, B, D}
{A, C, D}
{B, C}
{A, C}
{A, B, C, E}
{A, B, C}

A-Priori – Frequent Itemset (Bsp.) III

- Ein-elementige Frequent Itemsets
  - {A}: 6, {B}: 7, {C}: 6, {D}: 3, {E}: 2

- Zwei-elementige Frequent Itemsets
  - {A, B}: 4
  - {A, C}: 4
  - {A, D}: 2
  - {A, E}: 2
  - {B, C}: 4
  - {B, D}: 2
  - {B, E}: 2
  - ~~{C, D}: 1~~
  - ~~{C, E}: 1~~
  - ~~{D, E}: 0~~

Beispieltupel:
{A, B, E}
{B, D}
{B, C}
{A, B, D}
{A, C, D}
{B, C}
{A, C}
{A, B, C, E}
{A, B, C}

A-Priori – Frequent Itemset (Bsp.) IV

- Zweielementige Frequent Itemsets
  - {A, B}: 4, {A, C}: 4, {A, D}: 2, {A, E}: 2, {B, C}: 4, {B, D}: 2, {B, E}: 2

- Dreielementige Frequent Itemsets
  - {A, B, C}: 2
  - ~~{A, B, D}: 1~~
  - {A, B, E}: 2
  - ~~{A, C, D}~~
  - ~~{A, C, E}~~
  - ~~{A, D, E}~~
  - ~~{B, C, D}~~
  - ~~{B, C, E}~~
  - ~~{B, D, E}~~

Beispieltupel:
{A, B, E}
{B, D}
{B, C}
{A, B, D}
{A, C, D}
{B, C}
{A, C}
{A, B, C, E}
{A, B, C}

A-Priori – Frequent Itemset (Bsp.) V

- Drei-elmentige Frequent Itemsets
  - {A, B, C}: 2, {A, B, E}: 2
- Vier-elementige Frequent Itemsets
  - ~~{A, B, C, E}~~

Beispieltupel:
{A, B, E}
{B, D}
{B, C}
{A, B, D}
{A, C, D}
{B, C}
{A, C}
{A, B, C, E}
{A, B, C}

A-Priori – Assoziationsregeln

- Assoziationsregel-Kandidaten
  - Aufteilen der Frequent Itemsets in Antecedents und Consequents
- Berechnung der Confidence pro Kandidat
  - Erfüllt Kandidat gegebene minimal Confidence ⇒ Association Rule
  - Sonst verwerfen
- Modifikation: Andere Evaluierungsmasse
  - Chi-Quadrat-Maß, Informationsgewinn, …

A-Priori – Assoziationsregeln (Bsp.)

- Gesucht Assoziationsregeln mit mind. 2 Antecedents
  MinConf = 5/9

- Frequent Itemsets mit 3 oder mehr Elementen:
  - {A, B, C}
  - {A, B, E}

- Mögliche Assoziationsregeln
  - ~~A, B ⇒ C; Confidence = 2/4~~
  - ~~A, C ⇒ B; Confidence = 2/4~~
  - ~~B, C ⇒ A; Confidence = 2/4~~
  - ~~A, B ⇒ E; Confidence = 2/4~~
  - A, E ⇒ B; Confidence = 2/2
  - B, E ⇒ A; Confidence = 2/2

Beispieltupel:
{A, B, E}
{B, D}
{B, C}
{A, B, D}
{A, C, D}
{B, C}
{A, C}
{A, B, C, E}
{A, B, C}

Ende Einschub Data-Mining Grundlagen

Distributed Data-Mining

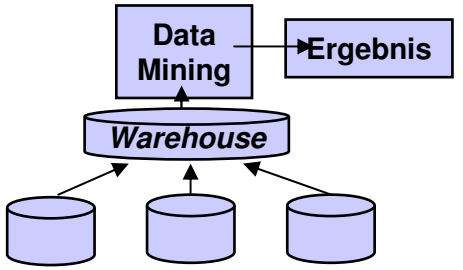
Motivation

- Data-Mining erzeugt (unter anderem)
  - Frequent Itemsets
  - Association Rules
  - Classifiers
  - Clusters
- Diese Ergebnisse von Data-Mining (in gewisser Weise aggregiert) lassen im Normalfall keine Rückschlüsse auf Individuen zu.
  → Wo soll da das Privatheitsproblem liegen?
  (abgesehen von outlier detection)

Das Problem:

- Wie berechne ich die Data-Mining Ergebnisse, ohne die zur Berechnung erforderlichen Daten zu kennen?

Motivation



- Beispiel Komplikationen bei Behandlung
  - Eine Gesundheitsorganisation möchte in allen deutschen Krankenhäusern aufgetretene Komplikationen identifizieren.
  - Alternative 1: Jedes Krankenhaus schickt alle Krankenakten an die Gesundheitsorganisation.
  - Alternative 2: Alle einigen sich auf eine Trusted-Third Party.
    - Wer stellt sich hierfür zur Verfügung?
    - Gefahr der zentralen Datenhaltung?
  - Lösungsansatz: Gibt es einen Weg, so dass keine Seite seine Daten preisgeben muss, die Analyse aber trotzdem möglich ist?

Grundlagen

Motivation - Partitionierung

- Horizontal partitionierte Daten

  Daten *gleicher Semantik* über *unterschiedliche Individuen* sind auf mehrere Parteien verteilt.

- Vertikal partitionierte Daten

  Daten *unterschiedlicher Semantik* über die *gleichen Individuen* sind auf mehrere Parteien verteilt.

- Kombiniert (hier wenig interessant) Warum?

  Daten *unterschiedlicher Semantik* über *unterschiedliche Individuen* sind auf mehrere Parteien verteilt.

Motivation

ID	Adresse	Einkommen
1	68766	40k
2	76131	50k
3	68259	90k
4	68766	15k

Unser Fokus

Horizontal partitioniert

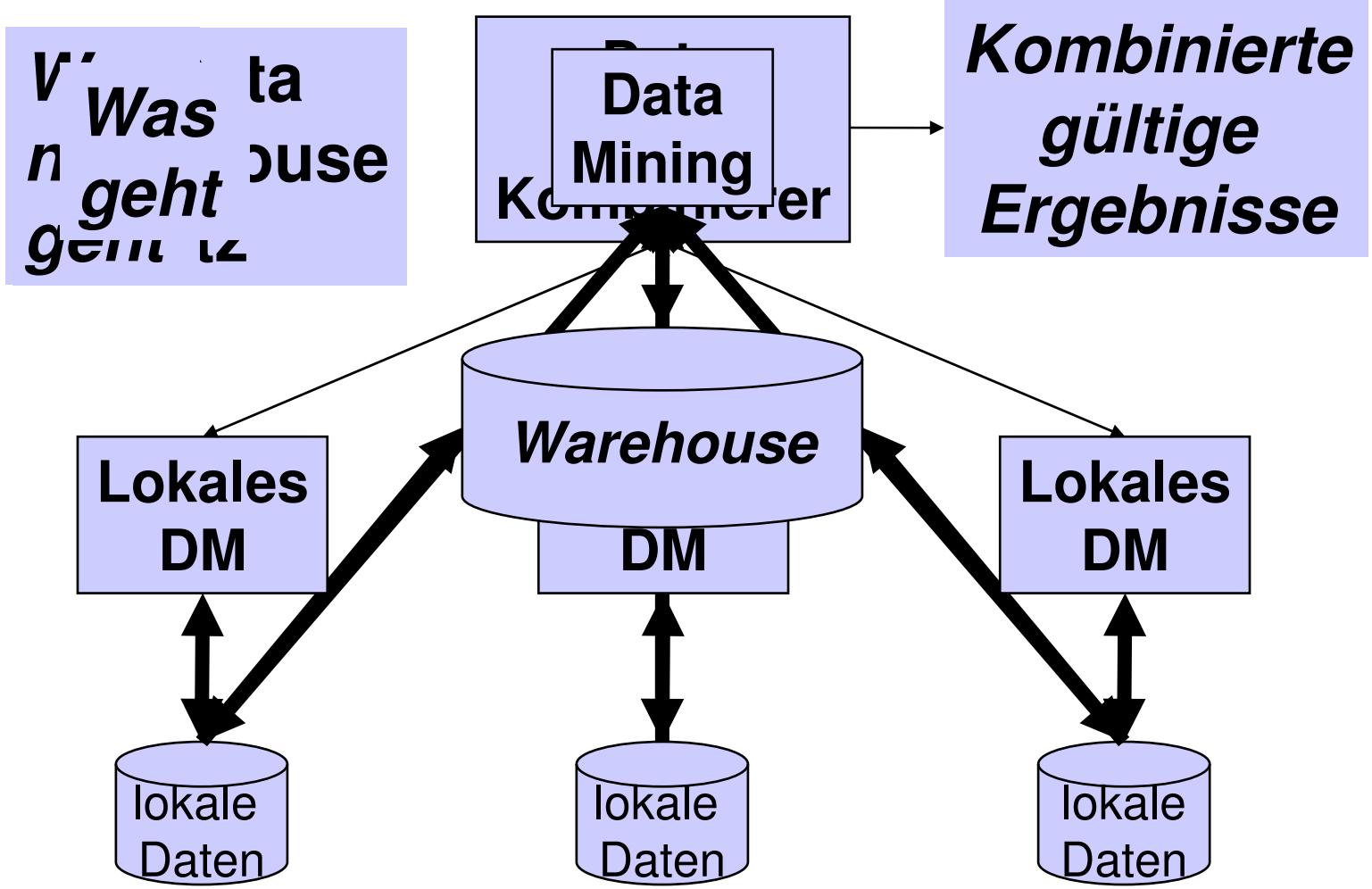
ID	Adresse	Einkommen
1	68766	40k
2	76131	50k

ID	Adresse	Einkommen
3	68259	90k
4	68766	15k

Vertikal partitioniert

ID	Adresse
1	68766
2	76131
3	68259
4	68766

ID	Einkommen
1	40k
2	50k
3	90k
4	15k

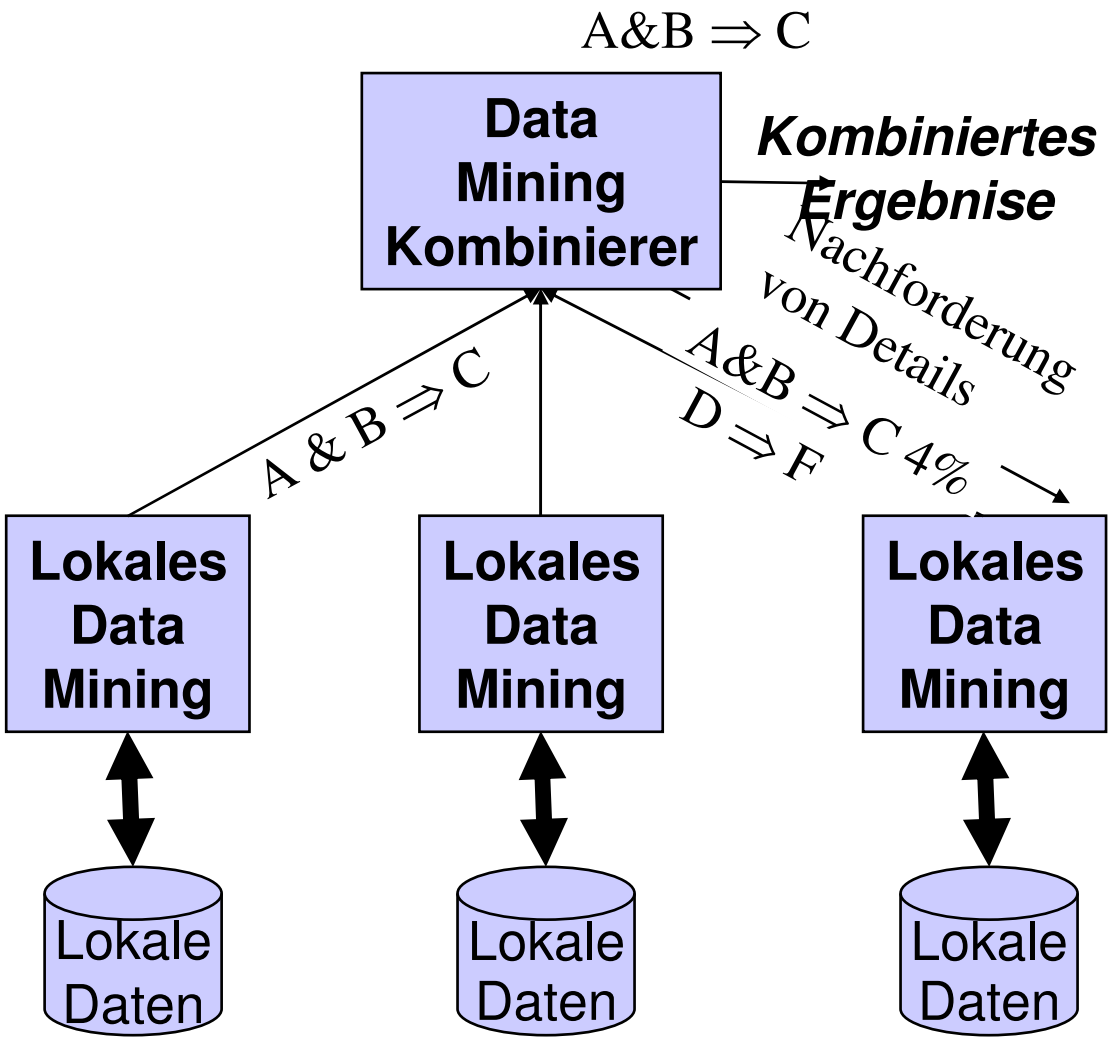


Distributed Association Rule Mining

Distributed A-Priori

- Lemma
  - Hat eine Regel einen globalen Support > k%, dann muss mind. eine Seite einen Support > k% haben.
- Algorithmus:
  - Fordere von jede Seite alle Regeln mit Support > k.
  - Fordere von jeder Seite die Anzahl der Transaktionen an, die die Regel unterstützen und die Anzahl aller Transaktionen der Seite.
  - Berechne den globalen Support jeder Regel. Das Lemma garantiert, dass alle Regeln mit Support k gefunden wurden.

Distributed A-Priori



Distributed A-Priori

- Berechnung der Confidence einer Association Rule AB ⇒ C

$$\mathbf{support}_{AB} = \frac{\sum_{i=1}^{sites} \mathbf{support\_count}_{AB}(i)}{\sum_{i=1}^{sites} \mathbf{database\_size}(i)}$$

$$\mathbf{support}_{AB \Rightarrow C} = \frac{\sum_{i=1}^{sites} \mathbf{support\_count}_{ABC}(i)}{\sum_{i=1}^{sites} \mathbf{database\_size}(i)}$$

$$\mathbf{confidence}_{AB \Rightarrow C} = \frac{\mathbf{support}_{\mathbf{AB \Rightarrow C}}}{\mathbf{support}_{\mathbf{AB}}}$$

  - Kein Austausch der Transaktionen selbst erforderlich
  - Verlustfreies Vorgehen

Frage

  - Sind die Regeln unter Umständen nicht selbst privat?

Privacy Preserving Association Rule Mining

Privacy Preserving Data Mining

- Regeln können private Information beinhalten
- Fortsetzung Bsp. Komplikationen
  - Versicherungen geben Informationen weiter
    - Problem mit Patientenakten
    - Aber auch Probleme bei Behandlungen, die ggf. nur in bestimmten Krankenhäusern auftreten
  - Versicherer kann bei Weitergabe solcher Informationen unter Druck geraten.

Privacy Preserving Data Mining

- Im Folgenden wird eine Lösung für das Problem vorgestellt
  - Cryptographiebasierter Ansatz
  - Seiten lernen nahezu nichts voneinander
  - Ansatz ist effizient
    - Kosten im Vergleich zu einer nicht ‚securen' Lösung O(candidate_itemset * sites) Verschlüsselungen
    - Konstanter Anstieg in der Anzahl der erforderlichen Nachrichten
  - Achtung: Ansatz für drei oder mehr Seiten

Secure Multiparty Computation

- Ziel: Berechnung des Ergebnisses, wenn jede Seite über Teile der erforderlichen Eingabe verfügt
- Yao's Millionaire's problem *(Yao '86)*
  - 'Secure' Berechnung ist möglich, wenn die Funktion als Schaltung modelliert werden kann
  - Idee: 'Secure' Berechnung der einzelnen Gates
  - Fortsetzen, bis Schaltung berechnet ist.
- Funktioniert auch für mehrere Parteien *(Goldreich, Micali, und Wigderson '87)*

- → Secure Multiparty Computation
- Was ist *Secure*?

Secure Multiparty Computation: Definitions

- Unsere Definition von *Secure*
  - Niemand weiß irgend etwas bis auf die eigene Eingabe und das Ergebnis
  - Formal: ∃ polynomial time S genau so, dass {S(x,f(x,y))} ≡ {View(x,y)}

- Semi-Honest model: folgt dem Protokoll, eine Partei darf aber alles verwenden, was sie während des Protokolls lern

Was wäre der Gegensatz
  - Malicious: “cheaten” um etwas herauszufinden

Beispiel: Exklusiv-Oder

A	B	A ⊻ B
T	T	F
T	F	T
F	T	T
F	F	F

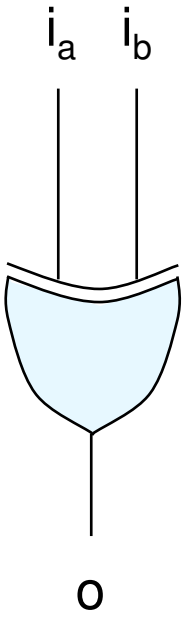
Truth Table XOR?

Person A

- Wähle zufälliges Bit r_a
- Schicke r_a an B
- Ersetze Input i_a durch $(i_a \oplus r_a)$
- Berechne $o_a = (i_a \oplus r_a) \oplus r_b$

Person B

- Wähle zufälliges Bit r_b
- Schicke r_b and A
- Ersetze Input i_b durch $(i_b \oplus r_b)$
- Berechne $o_b = (i_b \oplus r_b) \oplus r_a$



Bisher nichts preisgegeben außer der Zufallszahl

$$o = o_a \oplus o_b = ((i_a \oplus r_a) \oplus r_b) \oplus ((i_b \oplus r_b) \oplus r_a)$$
$$= i_a \oplus i_b \oplus r_a \oplus r_a \oplus r_b \oplus r_b$$
$$= i_a \oplus i_b$$

XOR = (A + B) mod 2
→ Assoziativ & Kommutativ

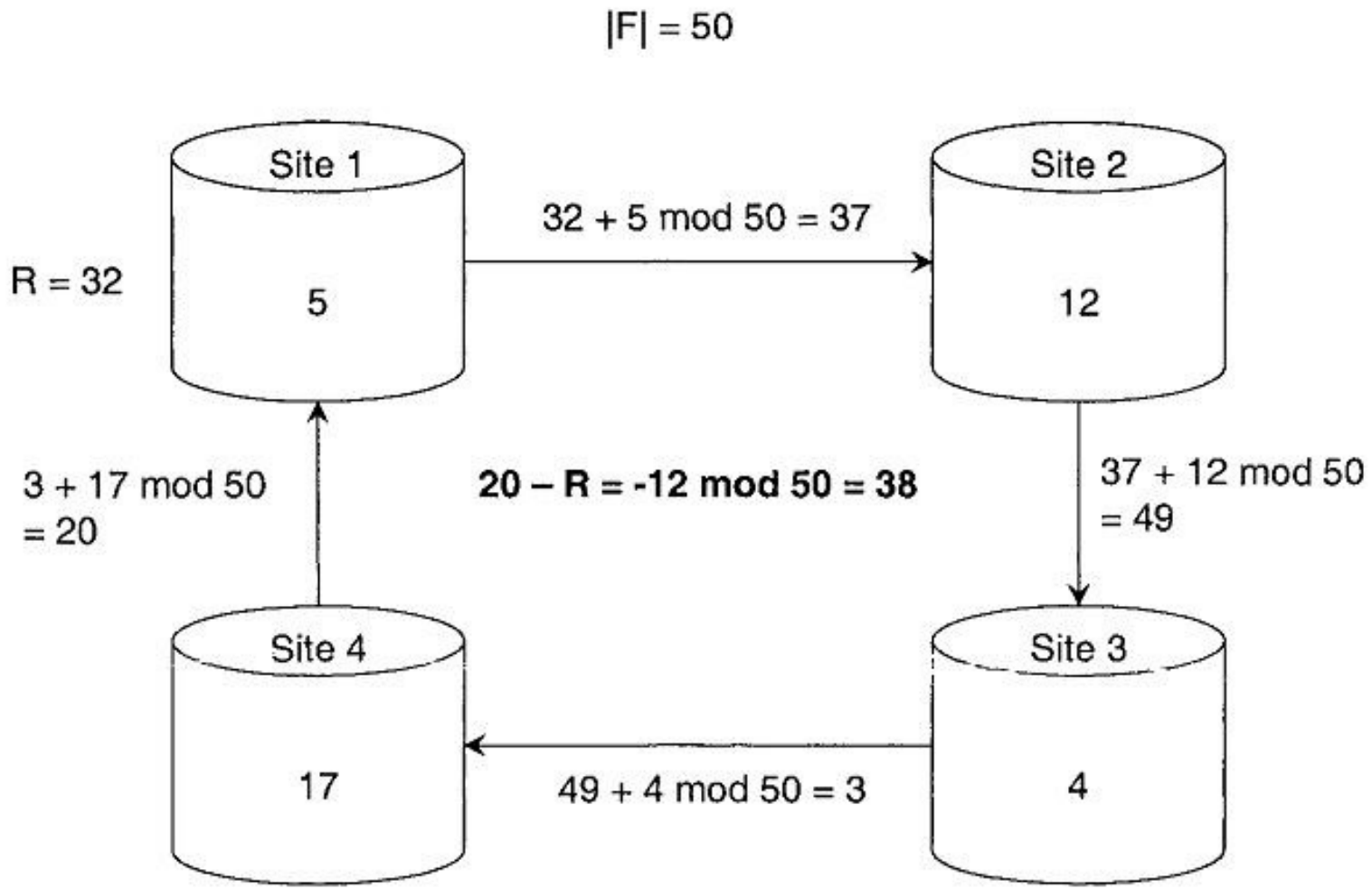
Secure Sum

- Vorbedingung
  - k Parteien
  - k > 2 warum?
  - Obere Schranke für die Summe (F) bekannt

Secure Sum

- Protokoll
  - P_1 erstellt Zufallszahl r von Gleichverteilung über F
  - P_1 berechnet $S_1 = x_1 + r \bmod |F|$ und sendet es an P_2
  - For $P_2 \ldots P_{k-1}$
    - P_i empfängt $S_{i-1} = r + \sum_{j=1}^{i-1} x_j \bmod |F|$
    - P_i berechnet $S_i = S_{i-1} + x_i \bmod |F| = \sum_{j=1}^{i} x_j \bmod |F|$ und sendet es an P_{i+1}
  - P_k empfängt $S_{k-1} = r + \sum_{j=1}^{k-1} x_j \bmod |F|$
  - P_k berechnet $S_k = S_{k-1} + x_i \bmod |F| = \sum_{j=1}^{k} x_j \bmod |F|$ und sendet es an P1
  - P_1 berechnet $S = S_k - r \bmod |F| = \sum_{j=1}^{k} x_j \bmod |F|$ und sendet es an alle anderen Parteien

Beispiel: Secure Sum



Secure Sum: Pro / Kontra

- Vorteile
  - Jeder außer P_1 sieht nur Nachrichten, die von einer Zufallszahl maskiert sind.
  - P_1 sieht dafür nur das finale Ergebnis
  - Einfach
  - Nützlich für viele DM Applikationen
- Nachteile
  - Parteien können zusammenarbeiten
    - Lösung durch Shares von x_i
    - Unterschiedliche Permutationen von Informationsflüssen
  - Anpassung an jede DM Applikation erforderlich

$$\mathbf{support}_{AB} = \frac{\sum_{i=1}^{sites} \mathbf{support\_count}_{AB}(i)}{\sum_{i=1}^{sites} \mathbf{database\_size}(i)}$$

$$\mathbf{support}_{AB \Rightarrow C} = \frac{\sum_{i=1}^{sites} \mathbf{support\_count}_{ABC}(i)}{\sum_{i=1}^{sites} \mathbf{database\_size}(i)}$$

$$\mathbf{confidence}_{AB \Rightarrow C} = \frac{\mathbf{support}_{AB \Rightarrow C}}{\mathbf{support}_{AB}}$$

Methode nach *Kantarcıoğlu and Clifton (DMKD'02) Überblick*

1. Finde die Vereinigung der lokalen large Candidate Itemsets *serurely*
2. Nach dem lokalen Pruning, berechne die large Itemsets mit global support *securely*
3. Zuletzt überprüfe die Konfidenz der potentiellen Assoziationsregeln *securely*

Secure Berechnung der Kandidaten

- Idee: Nutze kommutative Verschlüsselung
- $(E_a(E_b(x)) = E_b(E_a(x)))$
- Protokoll
  1. Berechne das lokale ‘Candidate Set’
  2. Ergebnis verschlüsseln und an die nächste Seite schicken
     - So lange durchführen, bis alle Seiten ihre Regeln verschlüsselt haben
  3. Eliminieren von Duplikaten
     - Die kommutative Verschlüsselung stellt sicher, dass gleiche Regeln auch nach der Verschlüsselung, unabhängig von der Reihenfolge, identisch sind.
     - Jede Seite entschlüsselt die verbleibenden Regeln
     - Anschließend verbleiben die relevanten Regeln

Anmerung zu Schritt 1

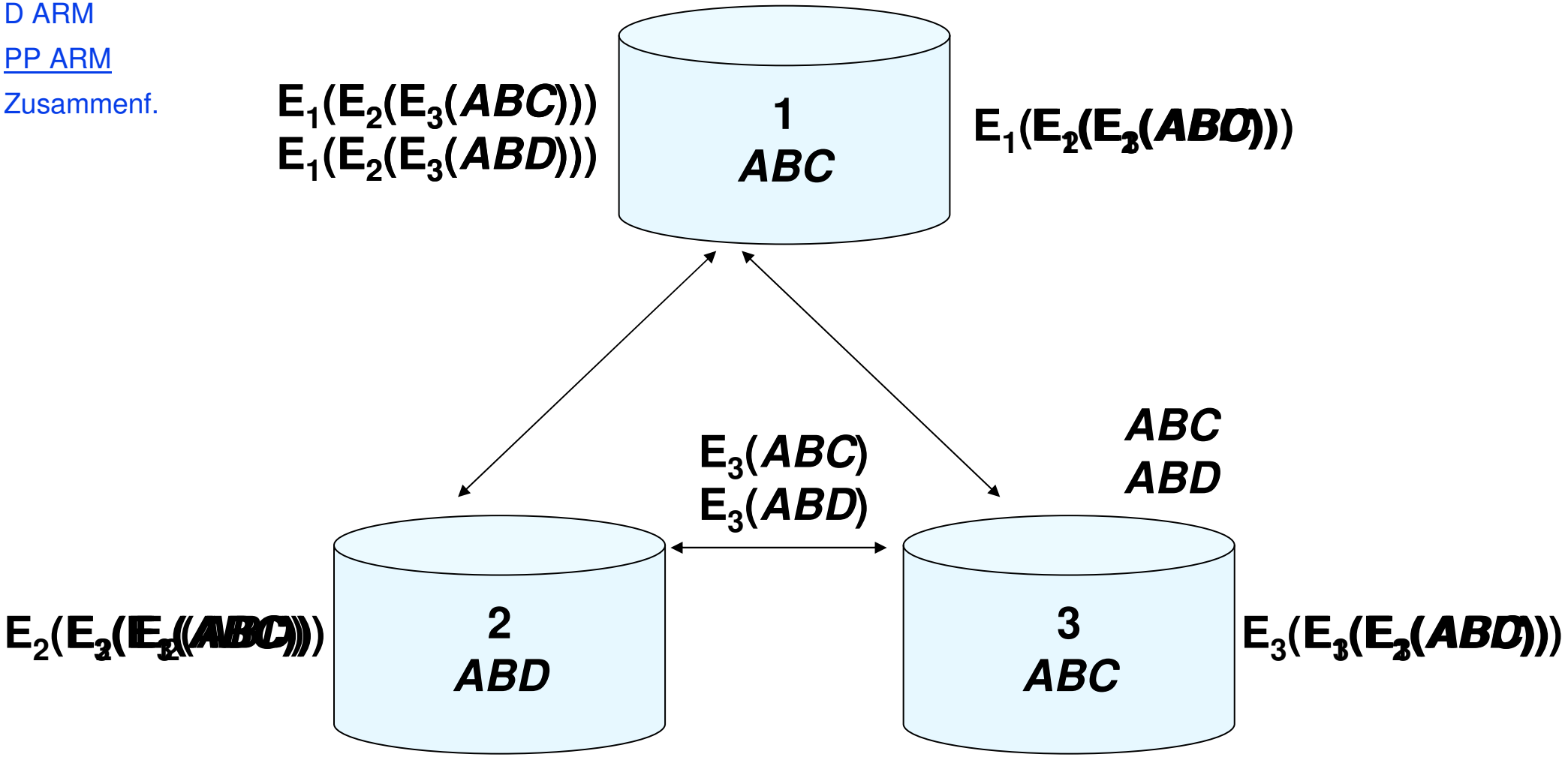
- Anmerkungen
  - Vorsicht geboten, dass durch die Sortierung der Regeln keine Information preisgegeben wird.
  - Um die Sicherheit zu erhöhen, kann Redundanz eingefügt werden.
  - Nicht voll ‘secure’ gemäß der Definition von Secure Multiparty Computation.

tb2

Warum?

burgthor; 24.06.2009

Berechnung des Candidate Sets



Nomenklatur (gem. Paper)

- Variablen
  - DB_i (Transaktions-)Datenbank auf Seite i
  - $|DB_i|$ Größe der Datenbank auf Seite i
  - s Support Schranke (minimaler Support), (prozentual)
  - c Konfidenz (prozentual)
  - X Itemset
  - x.sup Globaler Support von Itemset x
  - $X.Sup_i$ Lokaler Support von Itemset X (absolut) an Seite i

Berechne, welche Kandidaten globalen Support haben

Ziel: Prüfen

$$x.\sup \geq s^* \mid DB \mid = s^* \sum_{i=1}^{n} \left| DB_i \right| \qquad (1)$$

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

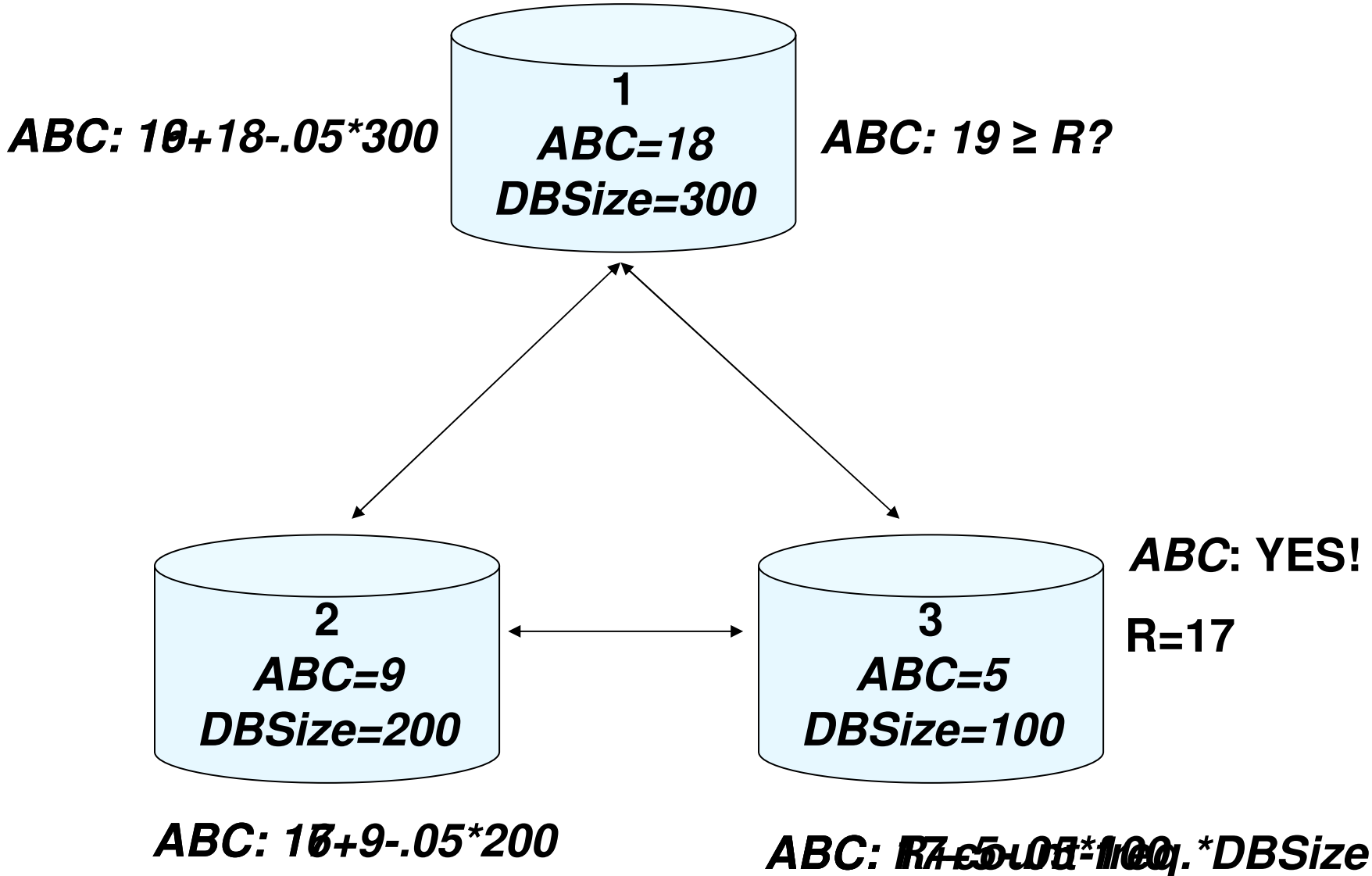
$$\sum_{i=1}^{n} (X.\sup_i - s^* \mid DB_i \mid) \geq 0 \qquad (3)$$

– Hinweis, das Prüfen von (1) ist identisch mit dem Prüfen von (3)

Berechne, welche Kandidaten globalen Support haben (2)

- Protokoll (Fortsetzung)
  - Berechne Sum ≥ 0:
    - $Seite_0$ generiert zufälliges r
      - Sendet $r+count_0 - frequency*dbsize_0$ zur $Seite_1$
    - $Seite_k$ addiert $count_k - frequency*dbsize_k$,
    - Sendet Ergebnis zu $Seite_{k+1}$
  - Endergebnis: Ist die Summe von $Seite_n$ - r ≥ 0?
    - Nutze *secure* two party computation
  - Dieses Protokoll ist *secure* im Sinne des semi-honest modells

Berechne, welche Kandidaten globalen Support haben (3) : Ist der Support von ABC ≥ 5%?



Berechnung der Konfidenz

- Protokoll
  - Gleich, wie für die Berechnung des supports
  - Berechnung der Konfidenz für $X \Rightarrow Y$

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

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

Zusammenfassung

Zusammenfassung

Einblicke in

- Data Mining
- Partitionierungen von Datenbeständen
- Verteiltes Data Mining
- Secure Multiparty computation
  - Yao‘s Millionaires Problem
  - Secure Sum
- Kommutative Verschlüsselung
- Secure Association Rule Mining

Zusammenfassung

Konkreter

- Data Mining wurde oftmals verpönt als privatheitsgefährdent.
- Hier gezeigt, dass Data Mining Ansätze die Privatheit in verteilten Systemen erst ermöglichen.
- Wir haben Muster auf verteilten Datenbeständen identifiziert, ohne dass die Originaldaten dazu preisgegeben werden mussten.
- Verlustfrei, unter Einsatz von Verschlüsselungstechniken.
- Verzicht auf zentrale Instanz.

Mögliche Prüfungsfragen

Mögliche Prüfungsfragen

- Begründen Sie mit dem Datenschutzrecht, warum ein Krankenhaus nicht jede Krankenakte and eine Gesundheitsorganisation geben darf?
- Werden beim verteilten Finden von Association Rules mehr oder weniger Ass. Rules an den Kombinierer gemeldet als tatsächlich vorliegen?
- Beschreiben Sie den Aufbau eines Securen 2-Bit Vergleichers.

Literatur

[1] Kantarcioglu, M. & Clifton, C.
Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data, *IEEE Trans. on Knowl. and Data Eng., IEEE Educational Activities Department,* **2004**, *16*, 1026-103

[2] **Privacy-Preserving Data Mining (Models and Algorithms):** Charu Aggarwal

[3] **Privacy-Preserving Data Mining:** Chris Clifton