“You have zero Privacy. Get over it.”

Scott McNealy, 1999

When to say NO to protect Privacy in the Context of Services.

Prof. Johann-Christoph Freytag, Ph.D.

Datenbanken und Infomationssysteme (DBIS)
Institut für Informatik (CS Department)
Humboldt-Universität zu Berlin
freytag@dbis.informatik.hu-berlin.de

Who am I?? - My Background

Visiting Professor @ Harvard Univ. (2009)

PhD @ Harvard Univ.

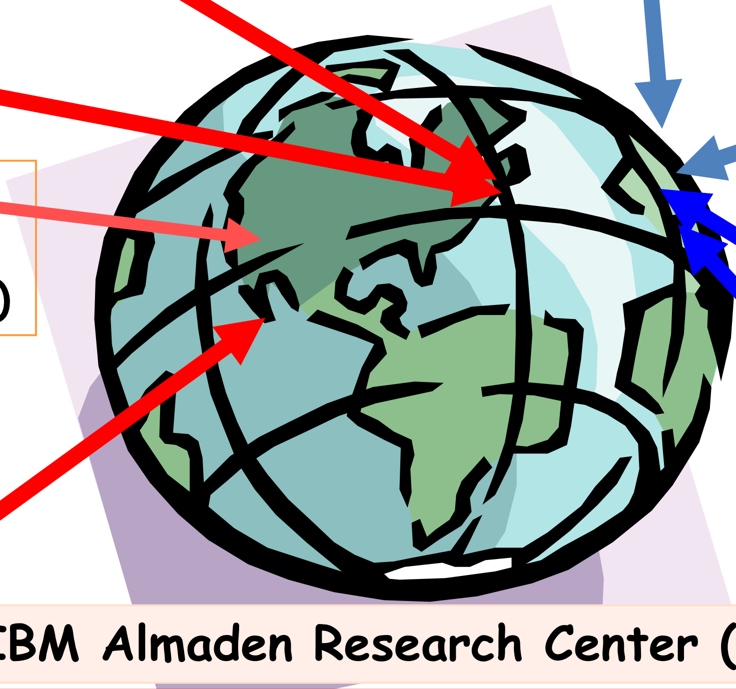
Visiting Scientist, Microsoft Res. (2002, 2005,2007, 2008)

Professor of DBIS @HUB (1994-)

ECRC (European Computer Industry Research Centre), München (87-89)

DEC's Database Technology Center, München (90-93)

Visiting Scientist @IBM Böblingen 2003-2005



Starburst project, IBM Almaden Research Center (85-87)

- Visiting Scientist, Almaden Research Center (97/98)
- Visiting Scientist, IBM SVL (2001)

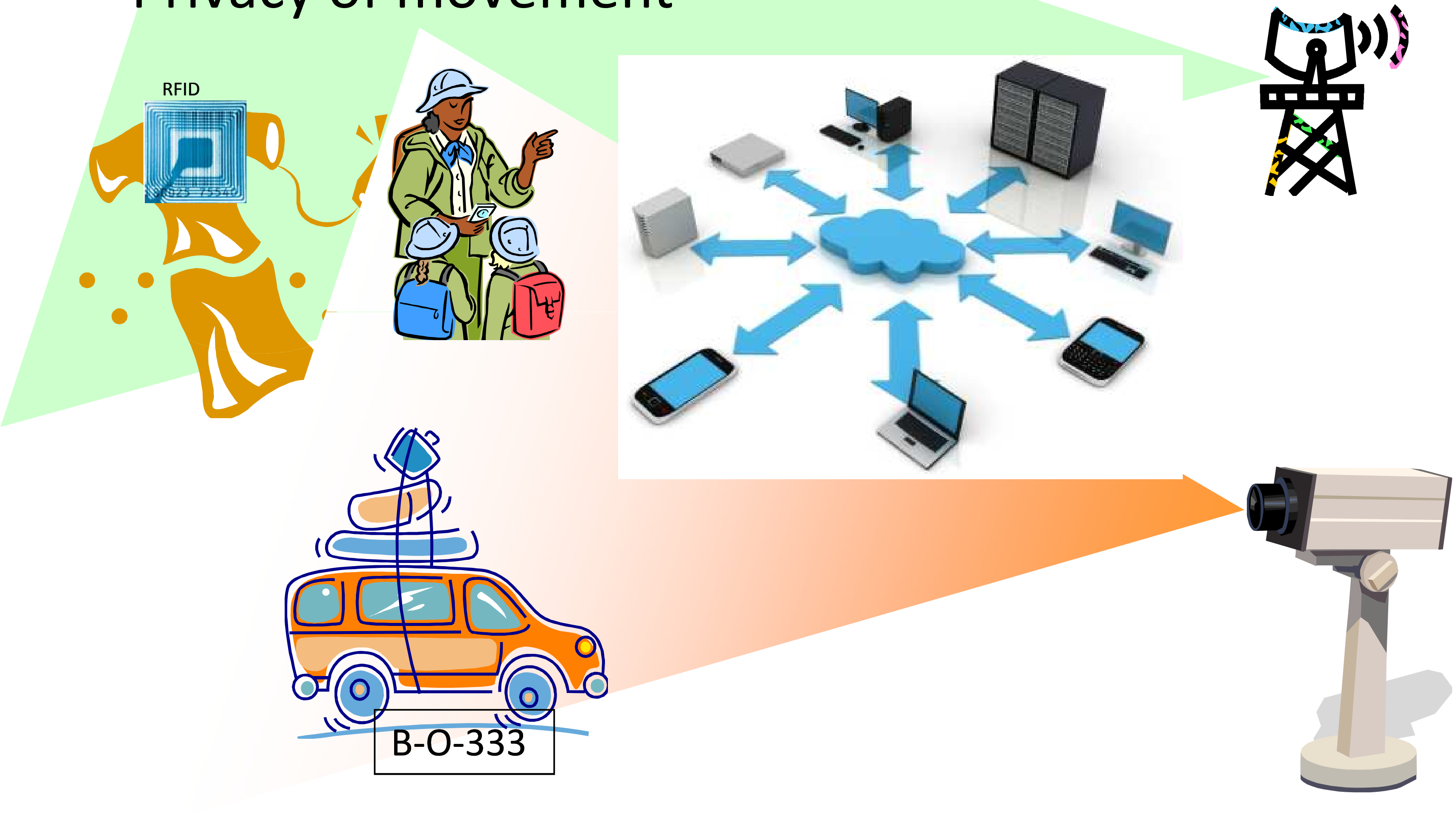
Overview

- What's the problem with Privacy?
- Brief intro to K-anonymity
- Data Requests (Queries) in a (distributed) World
  - Problem: When does the Adversary know too much?
  - Modeling the adversary's knowledge
  - Approaches for saying enough is enough

What‘s the Problem with Privacy??

Privacy violation …

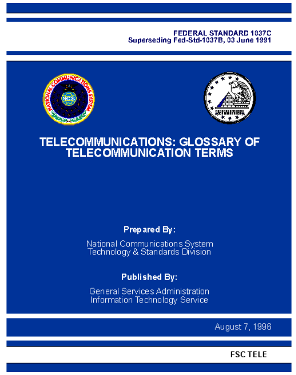
- Privacy of movement



Sensitive and personal Information

- Sensitive Information (slightly changed)

  information which through loss, or misuse, or unauthorized access to, or modification of which could adversely affect the interests of groups, organizations (such as the government or businesses), or the privacy to which individuals are entitled to by national or international law.



FEDSTD-1037C

- Personal (private) data/information

  *shall mean any information relating to an **identified or identifiable natural person**; an **identifiable person** is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural, or social identity*

Directive 95/46/EC of the European Parliament … on the protection of individuals with regard to the processing of personal data and on the free movement of such data

What is Privacy?

- **Definition 1:** [Sweeney, 2002]
  "**Privacy** reflects the ability of a person, organization, government, or entity to control its own space, where the concept of space (or "privacy space") takes on different contexts."
  - Physical space, against invasion
  - Bodily space, medical consent
  - Computer space, spam
  - Web browsing space, Internet privacy

- **Definition 2:** [Agrawal et al., 2002]
  "**Privacy** is the right of individuals to determine for themselves when, how, and to what extent information about them is communicated to others."
  (We shall call this data/information privacy)

Is it always obvious?

- Is it always obvious that privacy is violated or breached?
- Sweeney's Finding

[Sweeney, 2002]

  - In Massachusetts, USA, the *Group Insurance Commission* (GIC) is responsible for purchasing health insurance for state employees
  - GIC has to publish the data:

GIC					
ZIP	Date of birth	Sex	Diagnostic	Medication	…

http://lab.privacy.cs.cmu.edu/people/sweeney/

Sweeney's Finding (1)

- Sweeney paid $20 to buy the voter registration list for Cambridge, MA:

Voter					
Name	Address	…	**ZIP**	Date of birth	**Sex**

GIC					
ZIP	Date of birth	**Sex**	Diagnostic	Medication	…

- William Weld (former governor) lives in Cambridge, hence is in VOTER
- 6 people in VOTER share his **date of birth**
- only 3 of them were man (same **sex**)
- Weld was the only one in that **zip**
- Sweeney learned Weld's medical records!
- 87 % of population in U. S. can be identified by ZIP, dob, sex

Sweeney's Finding (2)

- **Observation:** *All systems worked as specified, yet an important data has leaked*
  - "Information leakage" occurred
  - Despite the observation that all "participating sites" worked as specified
  - Beyond correctness!
  - What's missing/causing the problem?
- How do we protect against this kind of "lack (leakage) of privacy"?

Privacy-Preserving Data Publishing
Challenge

- Objective
  - Publish privacy-relevant data
    - e.g., personal data
  - Preserve privacy of data subjects
    - e.g., individuals
- Purpose
  - e.g., statistic analyzes, legal regulations
- Challenge
  - **Given**
    - privacy-relevant data in microdata table *T*
      - attribute types: **identifying**, **sensitive**, **other**
  - **Goal**
    - generate privacy-preserving public release table *T**
      - information should remain practically useful

Name	Zipcode	Age	Sex	Disease
Alison	10000	18	F	Asthma
Ben	11000	19	M	Bronchitis
Clark	12000	20	M	Cold
Debra	12000	21	F	Diabetes
Elaine	12000	22	F	Earache
Fiona	12000	23	F	Flu
Gary	14000	24	M	Earache

Microdata table *T*

Privacy-Preserving Data Publishing
Insufficient Approach

- Insufficient approach
  - remove only identifying attributes
- Problem
  - set of other attributes could be used to identify individuals
    - call these attributes quasi-identifier
- Example
  - combination of Zipcode, Age, Sex is unique
  - with help of external data (e.g., voter list) identify individuals

Name	Zipcode	Age	Sex	Disease
Alison	10000	18	F	Asthma
Ben	11000	19	M	Bronchitis
Clark	12000	20	M	Cold
Debra	12000	21	F	Diabetes
Elaine	12000	22	F	Earache
Fiona	12000	23	F	Flu
Gary	14000	24	M	Earache

Microdata table *T*

Zipcode	Age	Sex	Disease
10000	18	F	Asthma
11000	19	M	Bronchitis
12000	20	M	Cold
12000	21	F	Diabetes
12000	22	F	Earache
12000	23	F	Flu
14000	24	M	Earache

Insufficient release table *T**

Name	Zipcode	Age	Sex
Alison	10000	18	F
Ben	11000	19	M

external data

Name	Disease
Alison	Asthma
Ben	Bronchitis

ϟ

Privacy-Preserving Data Publishing
Improved Approach

- Improved Approach
  – remove identifying attributes
  + generalize quasi-identifier
    - replace value with a less specific but semantically consistent value
- *k*-anonymity
  – for each tuple there exist $k-1$ other tuples which share the same values for all quasi-identifiers

Name	Zipcode	Age	Sex	Disease
Alison	10000	18	F	Asthma
Ben	11000	19	M	Bronchitis
Clark	12000	20	M	Cold
Debra	12000	21	F	Diabetes
Elaine	12000	22	F	Earache
Fiona	12000	23	F	Flu
Gary	14000	24	M	Earache

Microdata table *T*

Zipcode	Age	Sex	Disease
10–12000	18–20	*	Asthma
10–12000	18–20	*	Bronchitis
10–12000	18–20	*	Cold
12–14000	21–24	*	Diabetes
12–14000	21–24	*	Earache
12–14000	21–24	*	Flu
12–14000	21–24	*	Earache

3-anonymous release table *T**

Name	Zipcode	Age	Sex
Alison	10000	18	F

external data

Name	Disease
Alison	Asthma
Alison	Bronchitis
Alison	Cold

Privacy-Preserving Data Publishing
Better Approach

- Problem
  - tuples in QI-group with same sensitive value
    - QI-group: set of tuples with *same values for all quasi-identifiers*
- Better Approach
  - Restrict sensitive values in each QI-group
    - e.g., *distinct l-diversity*: ≥ *l* distinct sensitive values
    - many other approaches

Name	Zipcode	Age	Sex	Disease
Alison	10000	18	F	Asthma
Ben	11000	19	M	Bronchitis
Elaine	12000	22	F	Earache
Gary	14000	24	M	Earache

Microdata table *T*

Zipcode	Age	Sex	Disease	QI-groups
10–12000	18–20	*	Asthma	2-anonymous ✓
10–12000	18–20	*	Bronchitis	distinct 2-divers ✓
12–14000	21–24	*	Earache	2-anonymous ✓
				distinct 2-divers ϟ
12–14000	21–24	*	Earache	

Release table *T**

Privacy-Preserving Request (Query) Processing

Scenario

User
(potential adversary)

```
SELECT … FROM T …
```

Requests Q_1, Q_2, …

(anonymous) Results R_1, R_2, …

Database

Name	Zipcode	Age	Sex	Disease
Alison	10000	18	F	Asthma
Ben	11000	19	M	Bronchitis
Clark	12000	20	M	Cold
Debra	12000	21	F	Diabetes
Elaine	12000	22	F	Earache
Fiona	12000	23	F	Flu
Gary	14000	24	M	Earache

Zipcode	Age	Sex	Disease
10–12000	18–20	*	Asthma
10–12000	18–20	*	Bronchitis
10–12000	18–20	*	Cold

His knowledge: R_1, R_2, …

Goal: Combination of user knowledge (R_1, R_2, …) **comply with privacy criteria** (e.g., distinct *l*-diversity)

Example

Name	Age	Disease
Alison	18	Asthma
Ben	19	Bronchitis
Clark	20	Cold
Debra	21	Diabetes
Elaine	22	Earache
Fiona	23	Flu
Gary	24	Earache

Microdata table *T*

```
SELECT Age, Disease
FROM T …
```

Q_1: `… WHERE Age BETWEEN 18 AND 20` →

Age	Disease
18–20	Asthma
18–20	Bronchitis
18–20	Cold

R_1: distinct 3-divers

Q_2: `… WHERE Age BETWEEN 20 AND 23` →

Age	Disease
20–23	Cold
20–23	Diabetes
20–23	Earache
20–23	Flu

R_2: distinct 4-divers

Q_3: `… WHERE Age BETWEEN 22 AND 24` →

Age	Disease
22–24	Earache
22–24	Flu
22–24	Earache

R_3: distinct 2-divers

Example
Reasoning

Name	Age	Disease
Alison	18	Asthma
Ben	19	Bronchitis
Clark	20	Cold
Debra	21	Diabetes
Elaine	22	Earache
Fiona	23	Flu
Gary	24	Earache

Microdata table *T*

Q_1

Age	Disease
18–20	Asthma
18–20	Bronchitis
18–20	Cold

R_1: distinct 3-divers

Q_2

Age	Disease
20–23	Cold
20–23	Diabetes
20–23	Earache
20–23	Flu

R_2: distinct 4-divers

Q_3

Age	Disease
22–24	Earache
22–24	Flu
22–24	Earache

R_3: distinct 2-divers

Conclusion 1: Clark – Cold

Conclusion 2: Gary – Earache

Knowledge of adversary

- Anonymous results of queries (R_i)
- Quasi-identifier values of all tuples in *T*

Adversary wants to link individuals to sensitive attribute values

Clark

If an adversary knows that Clark is 20 years old, then he concludes:

- tuple for Clark in R_1
- tuple for Clark in R_2
- only one sensitive value in R_1 and R_2: Cold

Gary

If an adversary knows that Gary is 24 years old, then he concludes:

- tuple for Gary in R_3
- sensitive values in R_3: Earache, Flu
- assume Gary-Flu
- $\rightarrow$ in R_3: Elaine-Earache + Fiona-Earache
- $\rightarrow$ in R_2: 2 × Earache ϟ

Modeling the Request Results

Modeling – including Alternatives

- Simplification for presentation
  - identifiers (ID) are numbers: 1, 2, 3, …
  - sensitive attribute (SA) values are letters: A, B, C, …
- Reasoning of adversary after Query Q_1
  - tuples for 1, 2, 3
  - sensitive values A, B, C
  - $\rightarrow$ 6 possible permutations (= value assignments A_i) of these values
    - e.g., A_4: 1 has B, 2 has C, 3 has A

ID	Name	Age	Disease	SA
1	Alison	18	Asthma	A
2	Ben	19	Bronchitis	B
3	Clark	20	Cold	C
4	Debra	21	Diabetes	D
5	Elaine	22	Earache	E
6	Fiona	23	Flu	F
7	Gary	24	Earache	E

Microdata table *T*

Age	SA
18–20	A
18–20	B
18–20	C

Q_1
output to user

ID	SA
1, 2, 3	A, B, C

Q_1
for reasoning

ID	SA	SA	SA	SA	SA	SA
1	A	A	B	B	C	C
2	B	C	A	C	A	B
3	C	B	C	A	B	A
	A_1	A_2	A_3	A_4	A_5	A_6

possible value assignments

Query Graph
1^{st} Query

ID	SA
1	A
2	B
3	C
4	D
5	E
6	F
7	E

Data

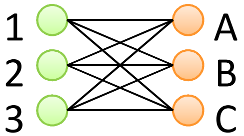
Query

Q_1

ID	SA
1, 2, 3	A, B, C

Graph

G_1

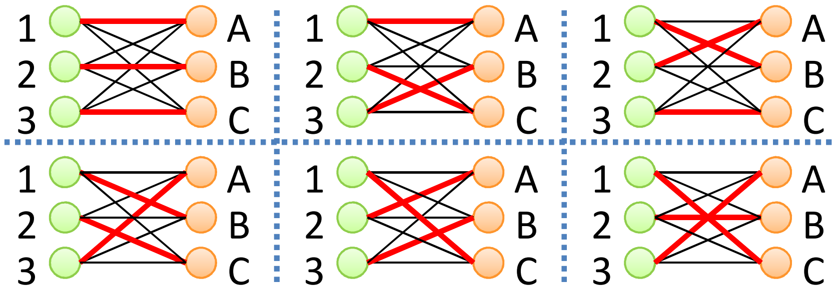


- Query graph $G_1 = (V_1, E_1)$
  - model query/result as graph
    - V_1: vertex for each tupel (ID) and each SA value
    - E_1: edges between tuple and SA vertices
- G_1 is bipartite

Assignments

ID	1	2	3	4	5	6
1	A	A	B	B	C	C
2	B	C	A	C	A	B
3	C	B	C	A	B	A

Perfect Matchings



- Each value assignment
  - = one perfect matching in G_1
- matching := set of edges without common vertices
- **perfect** := each vertex in one edge

List of Query Graphs
2nd Query

ID	SA
1	A
2	B
3	C
4	D
5	E
6	F
7	E

Data

Queries

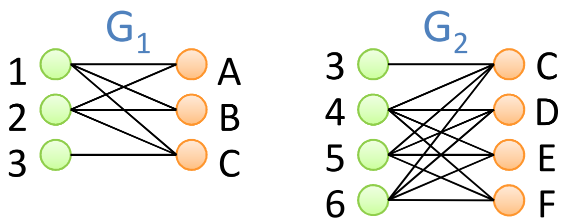
Q_1

ID	SA
1, 2, 3	A, B, C

Q_2

ID	SA
3, 4, 5, 6	C, D, E, F

Graphs

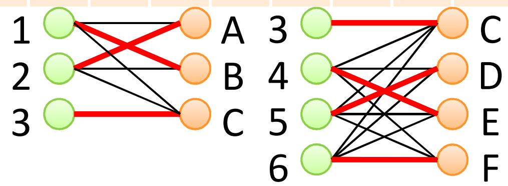


- List of query graphs $G^{(2)} = (G_1, G_2)$
  - remove unnecessary edges
    - In each graph: **each tuple vertex must match the same SA vertex (→ "valid")**
- No A, B in G_2
  - 3 cannot match A or B

Assignments

ID	1.1	1.2	1.3	1.4	1.5	1.6	2	3.1	…	3.6	4	5	6
1	A	A	A	A	A	A	A	B	…	B	B	C	C
2	B	B	B	B	B	B	C	A	…	A	C	A	B
3	C	C	C	C	C	C	B	C	…	C	A	B	A
4	D	D	E	E	F	F	↯	D	…	F	↯	↯	↯
5	E	F	D	F	E	D		E	…	D			
6	F	E	F	D	D	E		F	…	E			

PM 1.3



- Delete assignments/matchings: 2, 4, 5, 6
- Extend matchings
  - 1 → 1.1, …, 1.6, 3 → 3.1, …, 3.6
- In all 12 matchings
  - 3 matches C → 3 must have C
  - → ***privacy violation (identified!)***

List of Query Graphs

3rd Query

ID	SA
1	A
2	B
3	C
4	D
5	E
6	F
7	E

Data

Queries

Q_1

ID	SA
1, 2, 3	A, B, C

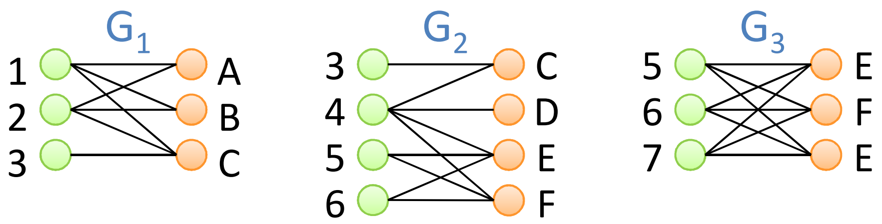
Q_2

ID	SA
3, 4, 5, 6	C, D, E, F

Q_3

ID	SA
5, 6, 7	E, F, E

Graphs



- List of query graphs $G^{(3)} = (G_1, G_2, G_3)$
- No C, D in G_3
  - 5 and 6 cannot match C or D

Assignments

ID	1.1	1.2	1.3	1.4	1.5	1.6	3.1	3.2	…
1	A	A	A	A	A	A	B	B	…
2	B	B	B	B	B	B	A	A	…
3	C	C	C	C	C	C	C	C	…
4	D	D	E	E	F	F	D	D	…
5	E	F	D	F	E	D	E	F	…
6	F	E	F	D	D	E	F	E	…
7	E	E	↯	↯	↯	↯	E	E	↯

- 4 assignments/matchings
  - → 3 must have C
    - = Clark has a Cold
  - → 4 must have D
    - = Debra has Diabetes
  - → 7 must have E
    - = Gary has Earache

Query Graph

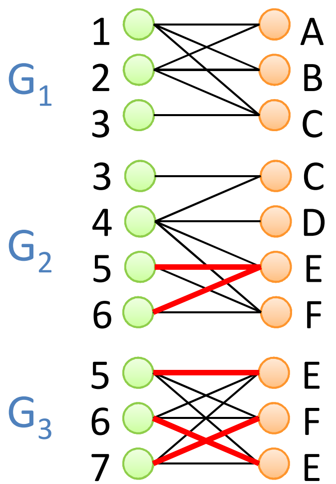
Merging of Graphs → Not Correct

- List of query graphs: one graph for each query
- Idea
  - Merge query graphs → only one graph for all queries
- Result
  - Modeling is not correct
  - There is no assignment 7-F but a matching with 7-F
    - remember example: Gary (7) has Cold (E) because he cannot have Cancer (F)

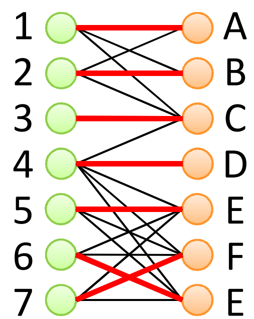
ID	SA
1	A
2	B
3	C
4	D
5	E
6	F
7	E

Data

List of query graphs



Merged graph



Assignments
- Assignment with 7-**F**: ↯

List of query graphs
- Matching with 7-**F**: ↯

Merged graph
- Only one tuple/SA vertex for each tuple/SA value
- Merge edges (union)
- Matching with 7-**F**: ✓

Privacy-Preserving Request (Query) Processing

Privacy Criterion

- **Goal**
  - Prevent linkage between individuals and sensitive values
    - Here: linkage between tuples and **SA** (**S**ensitive **A**ttribute) values
- **Desirable**
  - For each individual/tuple
    - adversary cannot distinguish between k different SA values
- **Privacy criterion**
  - For each individual I there are at least k different SA values s with probability P(s is SA value of I) > 0

    Example

    P(s is SA value of I) > 0 iff there is an assignment A with (I, s) ∈ A
  - We call this property ***k*-assign anonymity**

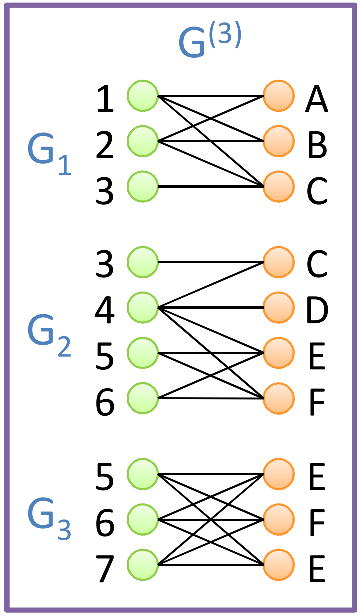
When to say enough is enough!

Privacy-Preserving Request (Query) Processing
Approach

- Transform problem of *privacy-preserving query processing* into a ***graph matching problem***
  - List of 1, …, n queries and results $\rightarrow$ List of query graphs $G^{(n)}$
  - P(s is SA value of I) > 0 $\Leftrightarrow$ perfect matching in $G^{(n)}$ with edge (I, s)
- k-assign anonymity
  - For each tuple t
    - There are at least k different SA values s and matchings M with $(t, s) \in M$
- Privacy violation if not k-assign anonymous for given k
- Approaches
  - **Approach 1:** Store all graphs, calculate all matchings
  - **Approach 2:** Store all perfect matchings
  - Unfortunately, both approaches are not trivial
    - Number of perfect matchings exponential in number of tuples
- Approaches
  1. **Approximation**: reduced number of modeled matchings
  2. **Heuristics**: calculation of matchings

Approach 1

- Idea
  - Store List of graphs
    - For each query/result one graph
  - After each query: calculate matchings
- Challenge
  - Given
    - list of query graphs $G^{(n)} = (G_1, \ldots, G_n)$
  - Wanted
    - For each tuple vertex *t* and SA vertex *s*
      - Calculate perfect matching including edge (*t*, *s*)
    - If there are ≥ *k* different sensitive values *s* → no privacy violation for *t*
- Approach
  - Reduce complexity of graphs
    - Delete as many unnecessary edges as possible
  - Use (integer) linear programming to solve problem
    - Unfortunately: exponential runtime

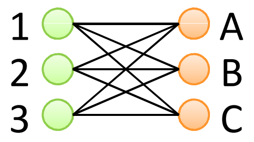


ID	SA	k
1	AB	2
2	AB	2
3	C	1
4	D	1
5	EF	2
6	EF	2
7	E	1

Approach 1
Integer Linear Programming

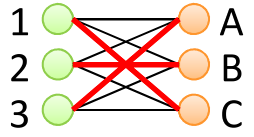
- ILP (Integer Linear Programming)
  - Variables for all edges *e*: $x_e \in \{0, 1\}$
    - $x_e = 1 \rightarrow x_e$ is matching edge
  - Equation for all vertices: $x(\delta(v)) = 1$
    - Otherwise not a matching
  - Maximize size of matching = maximize sum of all x_e = max $\sum_{e \in E} x_e$
- Example
  - $x_e \in \{0, 1\}$
  - Vertex 1: $x_{1A} + x_{1B} + x_{1C} = 1$
  - Vertex 2: $x_{2A} + x_{2B} + x_{2C} = 1$
  - Vertex 3: $x_{3A} + x_{3B} + x_{3C} = 1$
  - Vertex A: $x_{1A} + x_{2A} + x_{3A} = 1$
  - Vertex B: $x_{1B} + x_{2B} + x_{3B} = 1$
  - Vertex C: $x_{1C} + x_{2C} + x_{3C} = 1$
  - max: $x_{1A} + x_{1B} + x_{1C} + x_{2A} + x_{2B} + x_{2C} + x_{3A} + x_{3B} + x_{3C}$

G_1



- Solution (there are other solutions)
  - $x_{1C} = x_{2B} = x_{3A} = 1$
  - $x_{1A} = x_{1B} = x_{2A} = x_{2C} = x_{3B} = x_{3C} = 0$

G_1



Approach 2

- Idea for Approach 2
  - Store (perfect) matchings (PMs)
  - Compute matchings from stored matchings (= “extension”)
- Challenge
  - Given
    - set of perfect matchings $M_i^{(n)}$ for list of query graphs $G^{(n)} = (G_1, \ldots, G_n)$
    - new query graph G_{n+1}
  - Wanted
    - set of perfect matchings $M_i^{(n+1)}$ for $G^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$
- Approach
  - Reduce number of perfect matchings (PMs)
    - from exponential to polynomial
    - e.g., we only need to model a special type of „minimal“ matchings
  - Consider differences of PMs
    - reduces number of stored edges/complexity of algorithm
  - Extend existing PMs for $M_i^{(n)}$ to PMs for $M_i^{(n+1)}$

Approach 2
Reduce Complexity

- Main task
  - Reduce number of stored PMs
    - from exponential to polynomial
- Idea
  - Store matchings

    → model only a subset of all matchings
  - Compute matchings from stored PMs

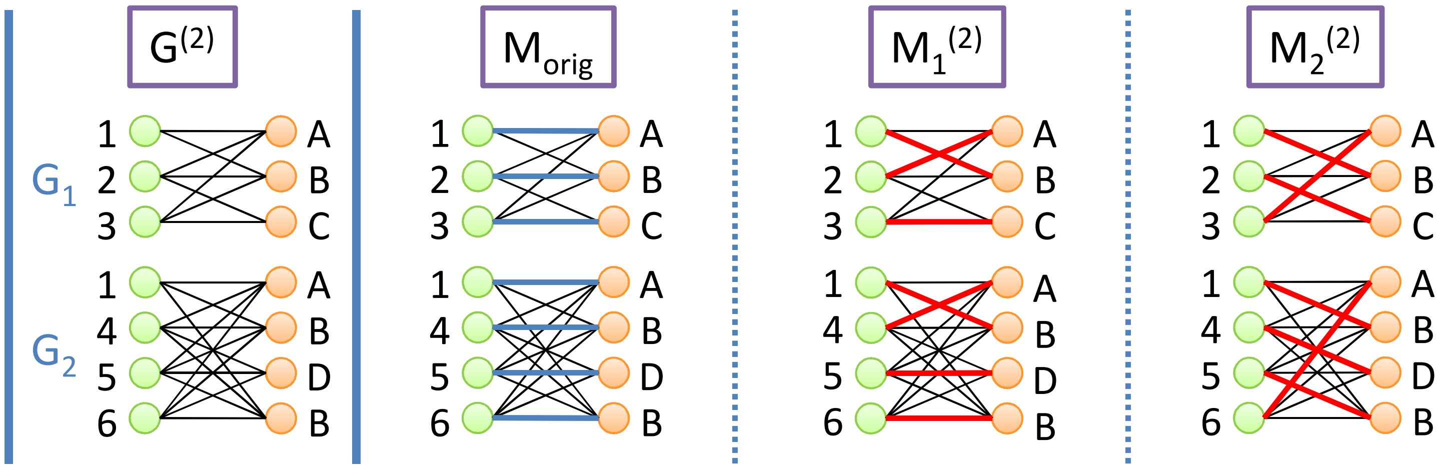
    → compute only a *subset* of all possible PMs
- Problem
  - Loss of matchings and assignments
  - Model is only an *approximation*
    - There are false positives
      - model says “no assignment” but there is one

Approach 2
Original (Perfect) Matching

- Original perfect matching M_{orig}
  - Each tuple vertex matches "correct" SA vertex
- Properties
  - Original PM can be directly derived from data
    - No need to be stored
  - Always exists!

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data

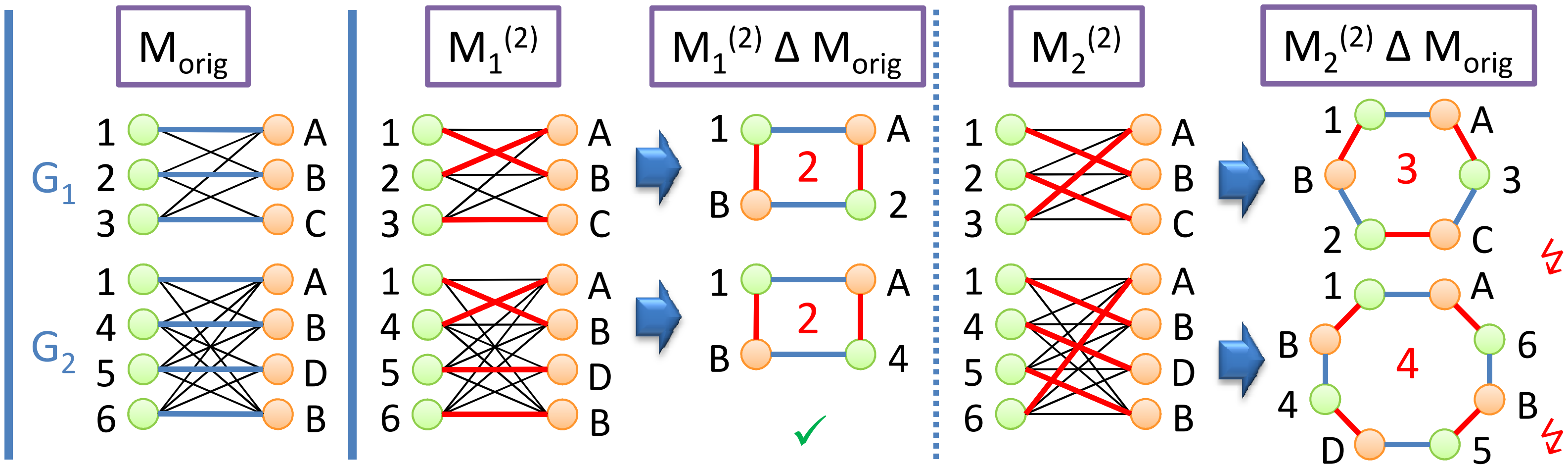


Approach 2
Symmetric Difference of Perfect Matchings

- Goal
  - Reduce number of modeled PMs (exponential → polynomial)
- Idea
  - Consider only PMs with small differences to original matching
- Approach
  - Difference to original PM
    - = symmetric difference $M_i \Delta M_{orig}$
  - = Circles with certain length
  - Model only PMs with length = 4 (2 edges of each of both matchings)
    - → Δ2-matchings

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data

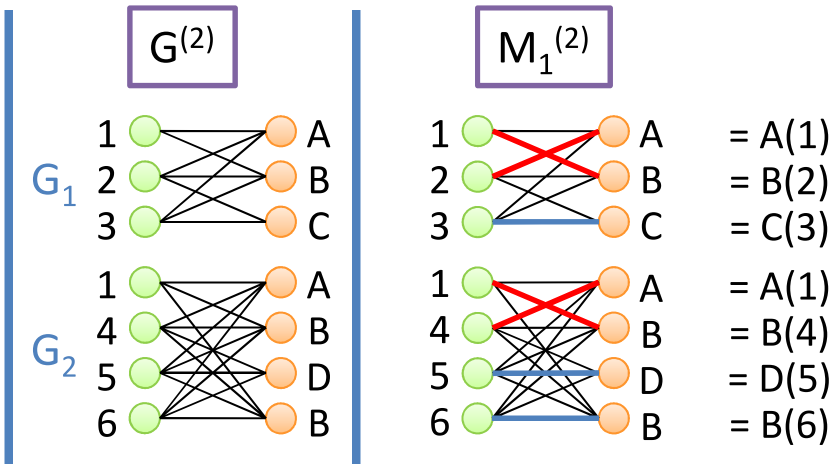


Approach 2
Representation of Perfect Matchings (PMs)

- Idea
  - Do not store complete PMs M (i.e., set of all edges)
    - store only difference $M - M_{orig}$
  - → Reduce storage complexity/decrease algorithm runtime
- Identify vertices
  - Tuple vertices: tuple ID
  - SA vertices: SA value + tuple ID (of tuple in M_{orig})
- Matching table T_{Δ}
  - Columns for tuple (ID), SA value, matching edges in $G^{(n)}$
  - Rows for each tuple

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data



$M_1^{(2)} - M_{orig}$

edges
(1, B(2))
(2, A(1))
(1, B(4))
(4, A(1))

ID	SA	$G^{(2)}$
1	A	B(2, 4)
2	B	A(1)
3	C	
4	B	A(1)
5	D	
6	B	

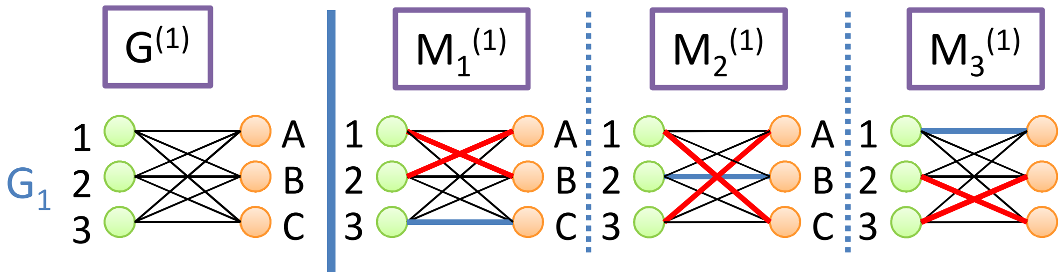
T_{Δ}

Approach 2
1st Query

- All combinations of tuples and SA values
  - Difference to original PM = 2 edges
    - Δ2-Matchings
  - Number of stored PMs with *n* tuples: $O(n^2)$
- Example
  - → 3-assign anonymous after G_1

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data



ID	SA	$G^{(1)}$	# SA
1	A	B(2) C(3)	3
2	B	A(1) C(3)	3
3	C	A(1) B(2)	3

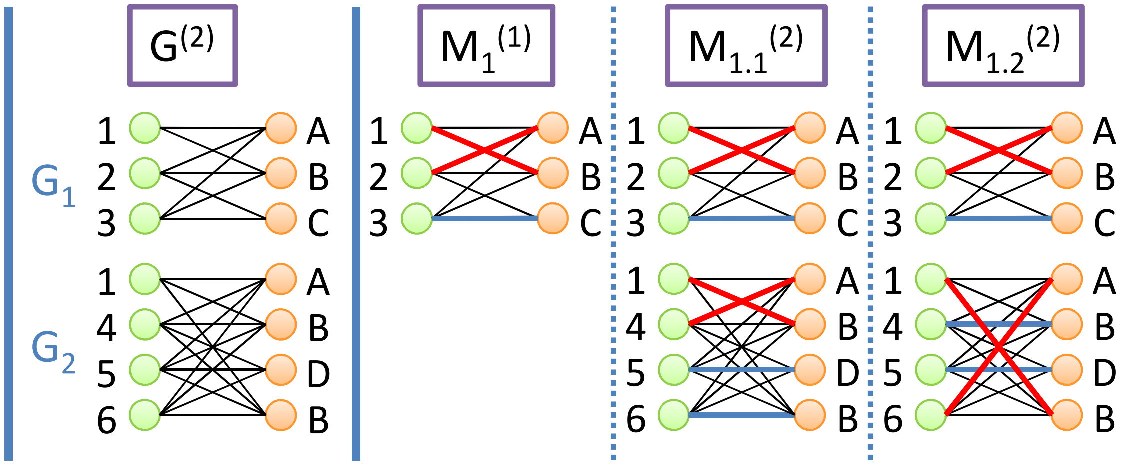
T_Δ

Approach 2
Extension of Perfect Matchings (PMs)

- Given
  - set of PMs $M_i^{(n)}$ for list of query graphs $G^{(n)} = (G_1, \ldots, G_n)$
  - new query graph G_{n+1}
- Wanted
  - set of PMs $M_i^{(n+1)}$ for $G^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$
  - Δ2-matchings + Δ-minimal (= "as few edges as possible")
- Example
  - Extending PM M_1 (here 2 opportunities)

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data

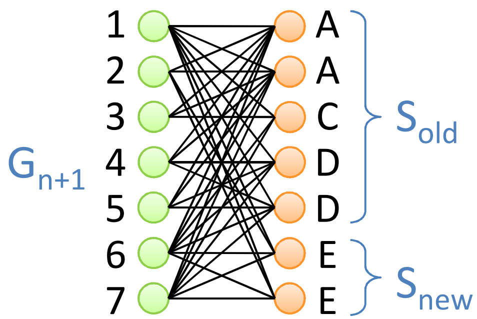


ID	SA	$G^{(1)}$	$G^{(2)}$
1	A	B(2)	B(2,4) B(2,6)
2	B	A(1)	A(1)
3	C		
4	B		A(1)
5	D		
6	B		A(1)

T_Δ

Approach 2
Algorithm

- S_{old} = set of old tuples (tuples in G_{n+1} and in $G^{(n)}$)
- S_{new} = set of new tuples (tuples in G_{n+1}, but not in $G^{(n)}$)
- **forall** old tuples $t \in S_{old}$ **do**
  - **forall** matching edge $e_\Delta = (t, s'(S_T))$ **do**
    - Case 0: s' does not appear in G_{n+1}
      - Delete matching with e_Δ
    - Case 1: exactly 1 tuple of S_T in G_{n+1} → /* okay */
    - Case 2: at least 2 tuples of S_T in G_{n+1}
      - Delete matching with e_Δ
    - Fall 3: no tuple of S_T in G_{n+1}
      - Save t and e_Δ for extension
- **forall** new tuples $t, t' \in S_{neu}$ with different SA values **do**
  - Generate new PM with edges $(t, s'(\{t'\}))$ and $(t', s(\{t\}))$
- **forall** SA values s of saved tuples t **do**
  - Extend all tuples with SA value s, so that no 2 tuples are extended with the same new tuple (→ extension algo)
  - Delete PM with e_Δ, which are not extended



ID	SA	$G^{(n)}$	$G^{(n+1)}$
1	A	B(8) C(3) D(4, 5) E(9)	– C(3) – E(9, 6)
2	A	E(10)	E(10, 7)
3	C	A(1)	A(1)
4	D	A(1)	–
5	D	A(1)	–
6	E		A(1)
7	E		A(2)
8	B	A(1)	–
9	E	A(1)	A(1)
10	E	A(2)	A(2)

T_Δ

Approach 2
Extension of Perfect Matchings (2)

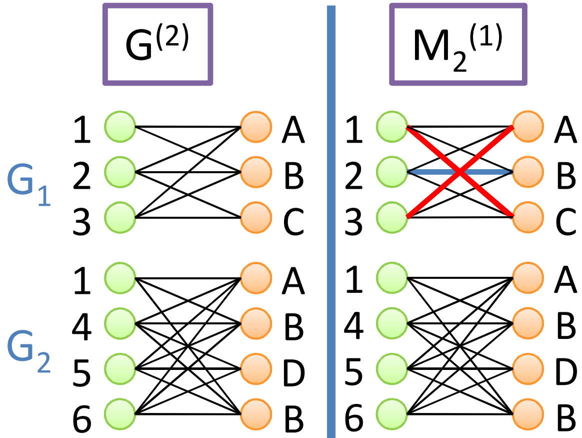
- Complete example
  - Extension of all PMs
  - Delete M_2 because no C in G_2
    - $M_2 = \{(1, C(3)), (3, A(1))\}$
  - $\rightarrow$ 2-assign anonymous after G_2

ID	SA	$G^{(1)}$	$G^{(2)}$	SA	# SA
1	A	B(2) C(3)	B(2,4) B(2,6) –	A, B	2
2	B	A(1) C(3)	A(1) C(3)	A, B, C	3
3	C	A(1) B(2)	– B(2)	B, C	2
4	B		A(1) D(5)	A, B, D	3
5	D		B(4) B(6)	B, D	2
6	B		A(1) D(5)	A, B, D	3

T_Δ

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data

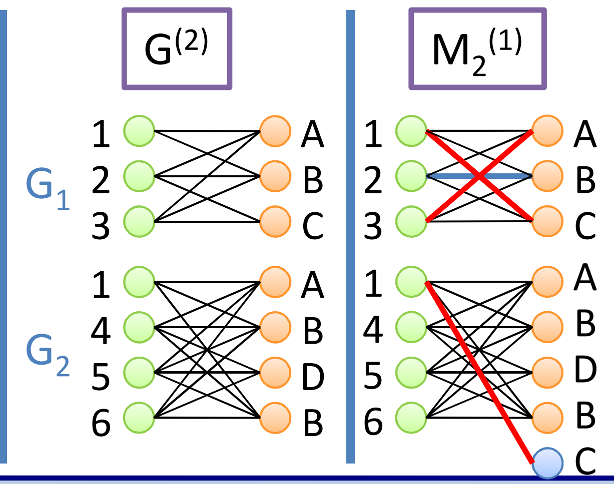


Approach 2
Extension of Perfect matching (PMs) (3)

- Violation of privacy criterion
  - e.g., 3-assign anonymity
- Add additional SA value
  - "counterfeit tuple"
- Example
  - Add C to G_2
  - PM $M_2^{(1)}$ remains
  - $\rightarrow$ 3-assign anonymous after G_2

ID	SA
1	A
2	B
3	C
4	B
5	D
6	B

Data



ID	SA	$G^{(1)}$	$G^{(2)}$	SA	# SA
1	A	B(2) C(3)	B(2,4) B(2,6) C(3)	A, B, C	3
2	B	A(1) C(3)	A(1) C(3)	A, B, C	3
3	C	A(1) B(2)	A(1) B(2)	A, B, C	3
4	B		A(1) C() D(5)	A, B, C, D	4
5	D		B(4) B(6) C()	B, C, D	3
6	B		A(1) C() D(5)	A, B, C, D	4

T_Δ

Summary

- Definition of Privacy
- Violation of Privacy
  - Several Approaches. k-anonymity, …
- Using the results of a series/sequence of queries
  - Modeling by Graphs
  - Algorithms run on Graphs
    - Complete set of perfect matchings: exponential number
    - Reduce set: polynomial number

Questions???

Thank you!!