Protecting Data Privacy in Outsourcing Scenarios

Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
pierangela.samarati@unimi.it

3rd International Workshop on Privacy and Anonymity in the Information Society (PAIS 2010)

Motivation (1)

- The management of large amount of sensitive information is quite expensive
- Database outsourcing is becoming increasingly popular (Database As a Service) [Hacigümüs et al., SIGMOD'02]
  - + significant cost savings and service benefits
  - + promises higher availability and more effective disaster protection than in-house operations
  - – sensitive data are not under the data owner's control

⟹ sensitive data have to be encrypted or kept separate from other PII

Motivation (2)

- Encryption proposed in DAS makes query evaluation more expensive or not always possible
- Often what is sensitive is the association between values of different attributes, rather than the values themselves
  - e.g., association between employee's names and salaries

$\Longrightarrow$ protect associations by breaking them, rather than encrypting

Fragmentation and encryption

- Recent solutions for enforcing privacy requirements couple:
  - encryption together with
  - data fragmentation
- Privacy requirements are represented as a set of confidentiality constraints that capture sensitivity of attributes and associations

Confidentiality constraints

- Sets of attributes such that the (joint) visibility of values of the attributes in the sets should be protected
- Sensitive attributes: the values assumed by some attributes are considered sensitive and cannot be stored in the clear
  $\Longrightarrow$ singleton constraints
- Sensitive associations: the association between values of given attributes is sensitive and should not be released
  $\Longrightarrow$ non-singleton constraints

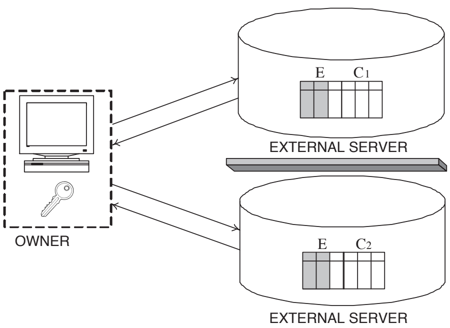
Outline

- Non-communicating pair of servers [Aggarwal et al., CIDR'05]
- Multiple fragments [ESORICS'07, ACM TISSEC'10]
- Departing from encryption: Keep a few [ESORICS'09]
- Fragments and loose associations (ongoing)

Non-Communicating Pair of Servers

Non-communicating pair of servers

- Confidentiality constraints are enforced by splitting information over two independent servers that cannot communicate (need to be completely unaware of each other)
  - Sensitive associations are protected by distributing the involved attributes among the two servers
  - Encryption is applied only when explicitly demanded by the confidentiality constraints or when storing the attribute in any of the servers would expose at least a sensitive association



- $E \cup C_1 \cup C_2 = R$
- $C_1 \cup C_2 \subseteq R$

Enforcing confidentiality constraints

- Confidentiality constraints $\mathscr{C}$ defined over a relation R are enforced by decomposing R as $\langle R_1, R_2, E \rangle$ where:
  - R_1 and R_2 include a unique tuple ID needed to ensure lossless decomposition
  - $R_1 \cup R_2 = R$
  - E is the set of encrypted attributes and $E \subseteq R_1$, $E \subseteq R_2$
  - for each $c \in \mathscr{C}$, $c \not\subseteq (R_1 - E)$ and $c \not\subseteq (R_2 - E)$

Confidentiality constraints – Example (1)

R = (Name,DoB,Gender,Zip,Position,Salary,Email,Telephone)

- ‘Telephone’ and ‘Email’ are sensitive (cannot be stored in the clear)
  - {Telephone}, {Email}
- ‘Salary’, ‘Position’, and ‘DoB’ are private of an individual and cannot be stored in the clear in association with the name
  - {Name,Salary}, {Name,Position}, {Name,DoB}
- {DoB,Gender,Zip} can work as quasi-identifier
  - {DoB,Gender,Zip,Salary}, {DoB,Gender,Zip,Position}
- Prevent an adversary from knowing association rules (e.g., between Position and Salary or between Salary and DoB)
  - {Position,Salary}, {Salary,DoB}

Enforcing confidentiality constraints – Example (2)

R = (Name,DoB,Gender,Zip,Position,Salary,Email,Telephone)

{Telephone}
{Email}
{Name,Salary}
{Name,Position}
{Name,DoB}
{DoB,Gender,Zip,Salary}
{DoB,Gender,Zip,Position}
{Position,Salary}
{Salary,DoB}

$\Longrightarrow$ R = (Name,DoB,Gender,Zip,Position,Salary,Email,Telephone)

- R_1: (ID,Name,Gender,Zip,Salary^e,Email^e,Telephone^e)
- R_2: (ID,Position,DoB,Salary^e,Email^e,Telephone^e)

Note that Salary is encrypted even if non sensitive per se since storing it in the clear in any of the two fragments would violate at least a constraint

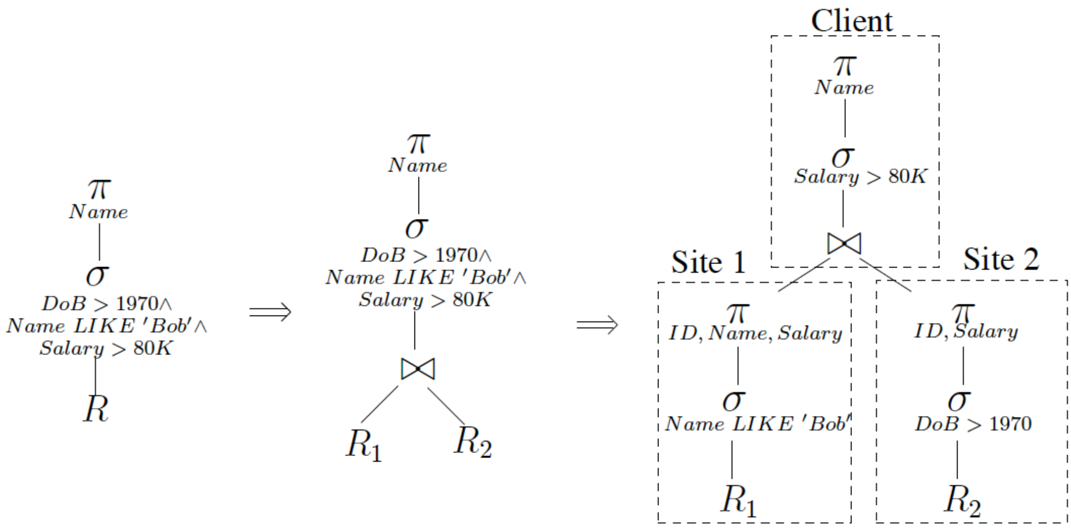
Query execution

At the logical level: replace R with $R_1 \bowtie R_2$
Query plans:

- Fetch R_1 and R_2 from the servers and execute the query locally
  - extremely expensive
- Involve servers S_1 and S_2 in the query evaluation
  - can do the usual optimizations, e.g., push down selections and projections
  - selections on encrypted attributes cannot be pushed down
  - different options for executing queries:
    - send sub-queries to both S_1 and S_2 in parallel, and join the results at the client
    - send only one of the two sub-queries, say to S_1; the tuple IDs of the result from S_1 are then used to perform a semi-join with the result of the sub-query of S_2 to filter R_2

Query execution – Example

- R_1: (ID,Name,Gender,Zip,Salary^e,Email^e,Telephone^e)
- R_2: (ID,Position,DoB,Salary^e,Email^e,Telephone^e)



Identifying the optimal decomposition (1)

Brute force approach for optimizing wrt workload W:

- For each possible safe decomposition of R:
  - optimize each query in W for the decomposition
  - estimate the total cost for executing the queries in W using the optimized query plans
- Select the decomposition that has the lowest overall query cost

Too expensive! $\Longrightarrow$ Exploit affinity matrix

Identifying the optimal decomposition (2)

Adapted affinity matrix M:

- $M_{i,j}$: 'cost' of placing cleartext attributes i and j in different fragments
- $M_{i,i}$: 'cost' of placing encrypted attribute i (across both fragments)

Goal: Minimize

$$\sum_{i,j:i\in(R_1-E),j\in(R_2-E)} M_{i,j} + \sum_{i\in E} M_{i,i}$$

Identifying the optimal decomposition (3)

Optimization problem equivalent to hypergraph coloring problem
Given relation R, define graph $G(R)$:

- attributes are vertices
- affinity value $M_{i,j} \Longrightarrow$ weight of arc (i,j)
- affinity value $M_{i,i} \Longrightarrow$ weight of vertex i
- confidentiality constraints $\mathscr{C}$ represent a hypergraph $H(R,\mathscr{C})$ on the same vertices

Identifying the optimal decomposition (4)

Find a 2-coloring of the vertices such that:

- no hypergraph edge is monochromatic
- the weight of bichromatic edges is minimized
- a vertex can be deleted (i.e., encrypted) by paying the price equal to the vertex weight

Coloring a vertex is equivalent to place it in one of the two fragments.
The 2-coloring problem is NP-hard.
Different heuristics, all exploiting:

- approximate min-cuts
- approximate weighted set cover

Multiple Fragments

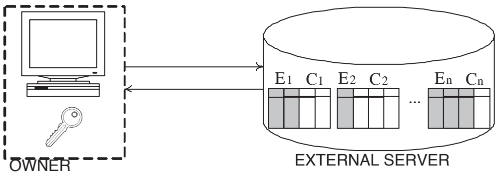
Multiple fragments (1)

Coupling fragmentation and encryption interesting and promising, but, limitation to two servers:

– too strong and difficult to enforce in real environments

– limits the number of associations that can be solved by fragmenting data, often forcing the use of encryption

$\Longrightarrow$ allow for more than two non-linkable fragments



- $E_1 \cup C_1 = \ldots = E_n \cup C_n = R$
- $C_1 \cup \ldots \cup C_n \subseteq R$

Multiple fragments (2)

- A fragmentation of R is a set of fragments $\mathscr{F} = \{F_1, \ldots, F_m\}$, where $F_i \subseteq R$, for $i = 1, \ldots, m$

- A fragmentation $\mathscr{F}$ of R correctly enforces a set $\mathscr{C}$ of confidentiality constraints iff the following conditions are satisfied:
  - $\forall F \in \mathscr{F}, \forall c \in \mathscr{C} : c \not\subseteq F$ (each individual fragment satisfies the constraints)
  - $\forall F_i, F_j \in \mathscr{F}, i \neq j : F_i \cap F_j = \emptyset$ (fragments do not have attributes in common)

Multiple fragments (3)

- Each fragment F is mapped to a physical fragment containing:
  - all the attributes in F in the clear
  - all the other attributes of R encrypted (a salt is applied on each encryption)
- Fragment $F_i = \{A_{i_1}, \ldots, A_{i_n}\}$ of R mapped to physical fragment F_i^e(<u>salt</u>,enc,$A_{i_1}, \ldots, A_{i_n}$):
  - each $t \in r$ over R is mapped to a tuple $t^e \in f_i^e$ with f_i^e a relation over F_i^e and:
    - t^e[*enc*] = $E_k(t[R - F_i] \otimes t^e[\textit{salt}])$
    - $t^e[A_{i_j}] = t[A_{i_j}]$, for $j = 1, \ldots, n$

Multiple fragments – Example (1)

MEDICALDATA

<u>SSN</u>	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

Multiple fragments – Example (2)

R=(SSN,Name,DoB,Zip,Illness,Physician)

{SSN} {Name, DoB} {Name, Illness} {DoB, Zip, Illness}
{Name, Zip} {Name, Physician} {DoB, Zip, Physician}

$\Longrightarrow$ R=(SSN,Name,DoB,Zip,Illness,Physician)

F_1

salt	enc	Name
s_1	α	Nancy
s_2	β	Ned
s_3	γ	Nell
s_4	δ	Nick

F_2

salt	enc	DoB	Zip
s_5	ε	65/12/07	94142
s_6	ζ	73/01/05	94141
s_7	η	86/03/31	94139
s_8	θ	90/07/19	94139

F_3

salt	enc	Illness	Physician
s_9	ι	hypertension	M. White
s_{10}	κ	gastritis	D. Warren
s_{11}	λ	flu	M. White
s_{12}	μ	asthma	D. Warren

Executing queries on fragments

- Every physical fragment of R contains all the attributes of R $\Longrightarrow$ no more than one fragment needs to be accessed to respond to a query
- If the query involves an encrypted attribute, an additional query may need to be executed by the client

Original query on R	**Translation over fragment F_3^e**
Q :=SELECT SSN, Name FROM MedicalData WHERE (Illness=‘gastritis’ OR Illness=‘asthma’) AND Physician=‘D. Warren’ AND Zip=‘94141’	Q^3 :=SELECT salt, enc FROM F_3^e WHERE (Illness=‘gastritis’ OR Illness=‘asthma’) AND Physician=‘D. Warren’ Q' := SELECT SSN, Name FROM *Decrypt*(Q^3, *Key*) WHERE Zip=‘94141’

Optimization criteria

- Goal: find a fragmentation that makes query execution efficient
- The fragmentation process can then take into consideration different optimization criteria:
    - number of fragments [ESORICS'07]
    - affinity among attributes [ACM TISSEC'10]
    - query workload [ICDCS'09]
- All criteria obey maximal visibility
    - only attributes that appear in singleton constraints (sensitive attributes) are encrypted
    - all attributes that are not sensitive appear in the clear in one fragment

Minimal number of fragments

Basic principles:

- avoid excessive fragmentation $\Longrightarrow$ minimal number of fragments

Goal:

- determine a correct fragmentation with the minimal number of fragments
  $\Longrightarrow$ NP-hard problem (minimum hyper-graph coloring problem)

Basic idea of the heuristic:

- define a notion of minimality that can be used for efficiently computing a fragmentation
    - $\mathscr{F}$ is minimal if all the fragmentations that can be obtained from $\mathscr{F}$ by merging any two fragments in $\mathscr{F}$ violate at least one constraint
- iteratively select an attribute with the highest number of non-solved constraints and insert it in an existing fragment if no constraint is violated; create a new fragment otherwise

Minimal number of fragments – Example

MEDICALDATA

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

Minimal fragmentation $\mathcal{F}$

- F_1 = {Name}
- F_2 = {DoB,Zip}
- F_3 = {Illness,Physician}

Merging any two fragments would violate at least a constraint

Maximum affinity

Basic principles:

- preserve the associations among some attributes
  - e.g., association (Illness,DoB) should be preserved to explore the link between a specific illness and the age of patients
- affinity matrix for representing the advantage of having pairs of attributes in the same fragment

Goal:

- determine a correct fragmentation with maximum affinity (sum of fragments affinity computed as the sum of the affinity of the different pairs of attributes in the fragment)
  $\Longrightarrow$ NP-hard problem (minimum hitting set problem)

Basic idea of the heuristic:

- iteratively combine fragments that have the highest affinity and do not violate any confidentiality constraint

Maximum affinity – Example

MedicalData

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d}
F_3={z}
F_4={i}
F_5={p}

	F_1	F_2	F_3	F_4	F_5
F_1		10	5	25	15
F_2			5	20	30
F_3				10	5
F_4					15
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	×	×	×	×		
d	×				×	×
z		×			×	×
i			×		×	
p				×		×

Maximum affinity – Example

MedicalData

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d}
F_3={z}
F_4={i}
F_5={p}

	F_1	F_2	F_3	F_4	F_5
F_1		-1	-1	-1	-1
F_2			5	20	30
F_3				10	5
F_4					15
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	✓	✓	✓	✓		
d	✓				×	×
z		✓			×	×
i			✓		×	
p				✓		×

Maximum affinity – Example

MEDICALDATA

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d}
F_3={z}
F_4={i}
F_5={p}

	F_1	F_2	F_3	F_4	F_5
F_1		-1	-1	-1	-1
F_2			5	20	**30**
F_3				10	5
F_4					15
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	✓	✓	✓	✓		
d	✓				×	×
z		✓			×	×
i			✓		×	
p				✓		×

Maximum affinity – Example

MEDICALDATA

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d,p}
F_3={z}
F_4={i}

	F_1	F_2	F_3	F_4	F_5
F_1		-1	-1	-1	
F_2			-1	**35**	
F_3				10	
F_4					
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	✓	✓	✓	✓		
d	✓				×	✓
z		✓			×	✓
i			✓		×	
p				✓		✓

Maximum affinity – Example

MEDICALDATA

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d,p,i}
F_3={z}

	F_1	F_2	F_3	F_4	F_5
F_1		-1	-1		
F_2			-1		
F_3					
F_4					
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	✓	✓	✓	✓		
d	✓				✓	✓
z		✓			✓	✓
i			✓		✓	
p				✓		✓

Maximum affinity – Example

MEDICALDATA

SSN	Name	DoB	Zip	Illness	Physician
123-45-6789	Nancy	65/12/07	94142	hypertension	M. White
987-65-4321	Ned	73/01/05	94141	gastritis	D. Warren
963-85-2741	Nell	86/03/31	94139	flu	M. White
147-85-2369	Nick	90/07/19	94139	asthma	D. Warren

Confidentiality constraints

c_0= {SSN}
c_1= {Name, DoB}
c_2= {Name, Zip}
c_3= {Name, Illness}
c_4= {Name, Physician}
c_5= {DoB, Zip, Illness}
c_6= {DoB, Zip, Physician}

F_1={n}
F_2={d,p,i}
F_3={z}

	F_1	F_2	F_3	F_4	F_5
F_1		-1	-1		
F_2			-1		
F_3					
F_4					
F_5					

	c_1	c_2	c_3	c_4	c_5	c_6
n	✓	✓	✓	✓		
d	✓				✓	✓
z		✓			✓	✓
i			✓		✓	
p				✓		✓

Maximum affinity fragmentation $\mathscr{F}$ (fragmentation affinity = 65)

Merging any two fragments would violate at least a constraint

Query workload

Basic principles:

- minimize the execution cost of queries
- representative queries (query workload) used as starting point
- query cost model: based on the selectivity of the conditions in queries and queries' frequencies

Goal:

- determine a fragmentation that minimizes the query workload cost $\Longrightarrow$ NP-hard problem (minimum hitting set problem)

Basic idea of the heuristic:

- exploit monotonicity of the query cost function with respect to a dominance relationship among fragmentations
- traversal (checking *ps* solutions at levels multiple of *d*) over a spanning tree of the fragmentation lattice

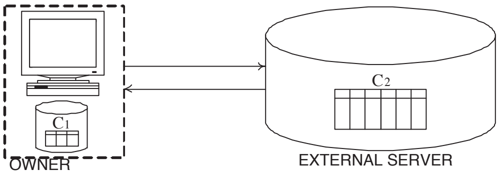
Departing from Encryption: Keep a Few

Keep a few

Basic idea:

– encryption makes query execution more expensive and not always possible
– encryption brings overhead of key management

$\Longrightarrow$ Depart from encryption by involving the owner as a trusted party to maintain a limited amount of data



- $C_1 \cup C_2 = R$

Fragmentation

Given:

- $R(A_1, \ldots, A_n)$: relation schema
- $\mathscr{C} = \{c_1, \ldots, c_m\}$: confidentiality constraints over R

Determine a fragmentation $\mathscr{F} = \langle F_o, F_s \rangle$ for R, where F_o is stored at the owner and F_s is stored at a storage server, and

- $F_o \cup F_s = R$ (completeness)
- $\forall c \in \mathscr{C}, c \not\subseteq F_s$ (confidentiality)
- $F_o \cap F_s = \emptyset$ (non-redundancy) /* can be relaxed */

At the physical level F_o and F_s have a common attribute (additional tid or non-sensitive key attribute) to guarantee lossless join

Fragmentation – Example

PATIENT

SSN	Name	DoB	Race	Job	Illness	Treatment	HDate
123-45-6789	Nancy	65/12/07	white	waiter	hypertension	ace	09/01/02
987-65-4321	Ned	73/01/05	black	nurse	gastritis	antibiotics	09/01/06
963-85-2741	Nell	86/03/31	asian	banker	flu	aspirin	09/01/08
147-85-2369	Nick	90/07/19	asian	waiter	asthma	anti-inflammatory	09/01/10

c_0 ={SSN}
c_1 ={Name,Illness}
c_2 ={Name,Treatment}
c_3 ={DoB,Race,Illness}
c_4 ={DoB,Race,Treatment}
c_5 ={Job,Illness}

F_o

tid	SSN	Illness	Treatment
1	123-45-6789	hypertension	ace
2	987-65-4321	gastritis	antibiotics
3	963-85-2741	flu	aspirin
4	147-85-2369	asthma	anti-inflammatory

F_s

tid	Name	DoB	Race	Job	HDate
1	Nancy	65/12/07	white	waiter	09/01/02
2	Ned	73/01/05	black	nurse	09/01/06
3	Nell	86/03/31	asian	banker	09/01/08
4	Nick	90/07/19	asian	waiter	09/01/10

Query evaluation

- Queries formulated on R need to be translated into equivalent queries on F_o and/or F_s
- Queries of the form: SELECT A FROM R WHERE C where C is a conjunction of basic conditions
  - C_o: conditions that involve only attributes stored at the client
  - C_s: conditions that involve only attributes stored at the sever
  - C_{so}: conditions that involve attributes stored at the client and attributes stored at the server

Query evaluation – Example

- F_o={SSN,Illness,Treatment}, F_s={Name,DoB,Race,Job,HDate}
- q = SELECT SSN, DoB
  FROM Patient
  WHERE (Treatment="antibiotic")
  AND (Job="nurse")
  AND (Name=Illness)
- The conditions in the WHERE clause are split as follows
  - C_o = {Treatment = "antibiotic"}
  - C_s = {Job = "nurse"}
  - C_{so} = {Name = Illness}

Query evaluation strategies

Server-Client strategy

- server: evaluate C_s and return result to client
- client: receive result from server and join it with F_o
- client: evaluate C_o and C_{so} on the joined relation

Client-Server strategy

- client: evaluate C_o and send tid of tuples in result to server
- server: join input with F_s, evaluate C_s, and return result to client
- client: join result from server with F_o and evaluate C_{so}

Server-client strategy – Example

q = SELECT SSN, DoB
FROM Patient
WHERE (Treatment = "antibiotic")
AND (Job = "nurse")
AND (Name = Illness)

C_o={Treatment = "antibiotic"}; C_s={Job = "nurse"}; C_{so}={Name = Illness}

q_s = SELECT tid,Name,DoB
FROM F_s
WHERE Job = "nurse"

q_{so} = SELECT SSN, DoB
FROM F_o JOIN r_s
ON F_o.tid=r_s.tid
WHERE (Treatment = "antibiotic")AND (Name = Illness)

Client-server strategy – Example

q = SELECT SSN, DoB
FROM Patient
WHERE (Treatment = "antibiotic")
AND (Job = "nurse")
AND (Name = Illness)

C_o={Treatment = "antibiotic"}; C_s={Job = "nurse"}; C_{so}={Name = Illness}

q_o = SELECT tid
FROM F_o
WHERE Treatment = "antibiotic"

q_s = SELECT tid,Name,DoB
FROM F_s JOIN r_o ON F_s.tid=r_o.tid
WHERE Job = "nurse"

q_{so} = SELECT SSN, DoB
FROM F_o JOIN r_s ON F_o.tid=r_s.tid
WHERE Name = Illness

Server-client vs client-server strategies

- If the storage server knows or can infer the query
  - Client-Server leaks information: the server infers that some tuples are associated with values that satisfy C_o
- If the storage server does not know and cannot infer the query
  - Server-Client and Client-Server strategies can be adopted without privacy violations
  - possible strategy based on performances: evaluate most selective conditions first

Minimal fragmentation

- The goal is to minimize the owner's workload due to the management of F_o
- Weight function w takes a pair $\langle F_o, F_s \rangle$ as input and returns the owner's workload (i.e., storage and/or computational load)
- A fragmentation $\mathscr{F} = \langle F_o, F_s \rangle$ is minimal iff:
  1. $\mathscr{F}$ is correct (i.e., it satisfies the completeness, confidentiality, and non-redundancy properties)
  2. $\nexists \mathscr{F}'$ such that $w(\mathscr{F}') < w(\mathscr{F})$ and $\mathscr{F}'$ is correct

Fragmentation metrics

Different metrics could be applied splitting the attributes between F_o and F_s, such as minimizing:

- storage
  - number of attributes in F_o (*Min-Attr*)
  - size of attributes in F_o (*Min-Size*)
- computation/traffic
  - number of queries in which the owner needs to be involved (*Min-Query*)
  - number of conditions within queries in which the owner needs to be involved (*Min-Cond*)

The metrics to be applied may depend on the information available

Data and workload information – Example

PATIENT(SSN,Name,DoB,Race,Job,Illness,Treatment,HDate)

A	*size*(*A*)
SSN	9
Name	20
DoB	8
Race	5
Job	18
Illness	15
Treatment	40
HDate	8

q	*freq*(*q*)	*Attr*(*q*)	*Cond*(*q*)
q_1	5	DoB, Illness	⟨Dob⟩, ⟨Illness⟩
q_2	4	Race, Illness	⟨Race⟩, ⟨Illness⟩
q_3	10	Job, Illness	⟨Job⟩, ⟨Illness⟩
q_4	1	Illness, Treatment	⟨Illness⟩, ⟨Treatment⟩
q_5	7	Illness	⟨Illness⟩
q_6	7	DoB, HDate, Treatment	⟨DoB,HDate⟩, ⟨Treatment⟩
q_7	1	SSN, Name	⟨SSN⟩, ⟨Name⟩

Weight metrics and minimization problems (1)

- Min-Attr. Only the relation schema (set of attributes) and the confidentiality constraints are known
  $\Longrightarrow$ minimize the number of attributes in F_o
  - $w_a(\mathscr{F})$=$card(F_o)$
- Min-Size. The relation schema (set of attributes), the confidentiality constraints, and the size of each attribute are known
  $\Longrightarrow$ minimize the physical size of F_o
  - $w_s(\mathscr{F})$=$\sum_{A \in F_o} size(A)$

Weight metrics and minimization problems (2)

- Min-Query. The relation schema (set of attributes), the confidentiality constraints, and a representative profile of the expected query workload are known

  Query workload profile:
  $\mathscr{Q}$=$\{(q_1, freq(q_1), Attr(q_1)), \ldots, (q_l, freq(q_l) Attr(q_l))\}$
  - $q_1, \ldots, q_l$ queries to be executed
  - $freq(q_i)$ expected execution frequency of q_i
  - $Attr(q_i)$ attributes appearing in the WHERE clause of q_i

  $\Longrightarrow$ minimize the number of query executions that require processing at the owner
  - $w_q(\mathscr{F})$=$\sum_{q \in \mathscr{Q}} freq(q)\ s.t.\ Attr(q) \cap F_o \neq \emptyset$

Weight metrics and minimization problems (3)

- Min-Cond. The relation schema (set of attributes), the confidentiality constraints, and a complete profile (conditions in each query of the form a_i op v or a_i op a_j) of the expected query workload are known

  Query workload profile:
  $\mathcal{Q}$={$(q_1, freq(q_1), Cond(q_1)), \ldots, (q_l, freq(q_l) Cond(q_l))$}
  - $q_1, \ldots, q_l$ queries to be executed
  - $freq(q_i)$ expected execution frequency of q_i
  - $Cond(q_i)$ set of conditions in the WHERE clause of query q_i; each condition is represented as a single attribute or a pair of attributes

  $\Longrightarrow$ minimize the number of conditions that require processing at the owner
  - $w_c(\mathcal{F}) = \sum_{q \in \mathcal{Q}} \sum_{cnd \in Cond(q)} freq(q)$ s.t. $Attr(cnd) \cap F_o \neq \emptyset$

Modeling of the minimization problems (1)

- All the problems of minimizing storage or computation/traffic aim at identifying a hitting set
  - F_o must contain at least an attribute for each constraint
- Different metrics correspond to different criteria according to which the hitting set should be minimized
- We represent all the criteria with a uniform model based on:
  - target set: elements (i.e., attributes, queries, or conditions) with respect to which the minimization problem is defined
  - weight function: function that associates a weight with each target element
  - weight of a set of attributes: sum of the weights of the targets intersecting with the set

$\Longrightarrow$ compute the hitting set of attributes with minimum weight

Modeling of the minimization problems (2)

Problem	**Target** $\mathcal{T}$	$w(t)\ \forall t \in \mathcal{T}$
Min-Attr	$\{\{A\} \mid A \in R\}$	1
Min-Size	$\{\{A\} \mid A \in R\}$	$size(A)$ s.t. $\{A\}=t$
Min-Query	$\{attr \mid \exists q \in \mathcal{Q},\ Attr(q)=attr\}$	$\sum_{q\in\mathcal{Q}} freq(q)$ s.t. $Attr(q)=t$
Min-Cond	$\{Attr(cnd) \mid \exists q \in \mathcal{Q},\ cnd \in Cond(q)\}$	$\sum_{q\in\mathcal{Q}} \sum_{cnd \in Cond(q)} freq(q)$ s.t. $Attr(cnd)=t$

Weighted Minimum Target Hitting Set Problem (WMTHSP). Given a finite set A, a set C of subsets of A, a set $\mathcal{T}$ (target) of subsets of A, and a weight function $w{:}\mathcal{T}\rightarrow\mathbb{R}^+$, determine a subset S of A such that:

1. S is a hitting set of A
2. $\nexists S'$ such that S' is a hitting set of A and $\sum_{t\in\mathcal{T}, t\cap S' \neq \emptyset} w(t) < \sum_{t\in\mathcal{T}, t\cap S\neq\emptyset} w(t)$

Modeling of the minimization problems (3)

- The Minimum Hitting Set Problem can be reduced to the WMTHSP
  - $\mathcal{T} = \{A_1,\ldots,A_n\}$; $w(\{A_i\}) = 1$, $i=1,\ldots,n$
  - minimizing $\sum_{t\in\mathcal{T}, t\cap S\neq\emptyset} w(t)$ is equivalent to minimizing the cardinality of the hitting set S

  $\Longrightarrow$ WMTHSP is NP-hard

- We propose a heuristic algorithm for solving the WMTHSP that:
  - ensures minimality, that is, moving any attribute from F_o to F_s violates at least a constraint
  - has polynomial time complexity in the number of attributes (efficient execution time)
  - provides solutions close to the optimum (from experiments run: optimum was returned in many cases, 14% maximum error observed)

Heuristic algorithm – Input and output

- Input
  - $\mathscr{A}$: set of attributes not appearing in singleton constraints
  - $\mathscr{C}$: set of well defined constraints
  - $\mathscr{T}$: set of targets
  - w: weight function defined on $\mathscr{T}$
- Output
  - $\mathscr{H}$: set of attributes composing, together with those appearing in singleton constraints, F_o
  - F_s is computed as $R \setminus F_o$, obtaining a correct fragmentation

Heuristic algorithm – Data structure

- Priority-queue *PQ* with an element *E* for each attribute:
  - $E.A$: attribute
  - $E.C$: pointers to non-satisfied constraints that contain $E.A$
  - $E.T$: pointers to the targets non intersecting $\mathscr{H}$ that contain $E.A$
  - $E.n_c$: number of constraints pointed by $E.C$
  - $E.w$: total weight of targets pointed by $E.T$

  Priority dictated by $E.w/E.n_c$: elements with lower ratio have higher priority

Heuristic algorithm – Example of initialization (1)

PATIENT(SSN,Name,DoB,Race,Job,Illness,Treatment,HDate)

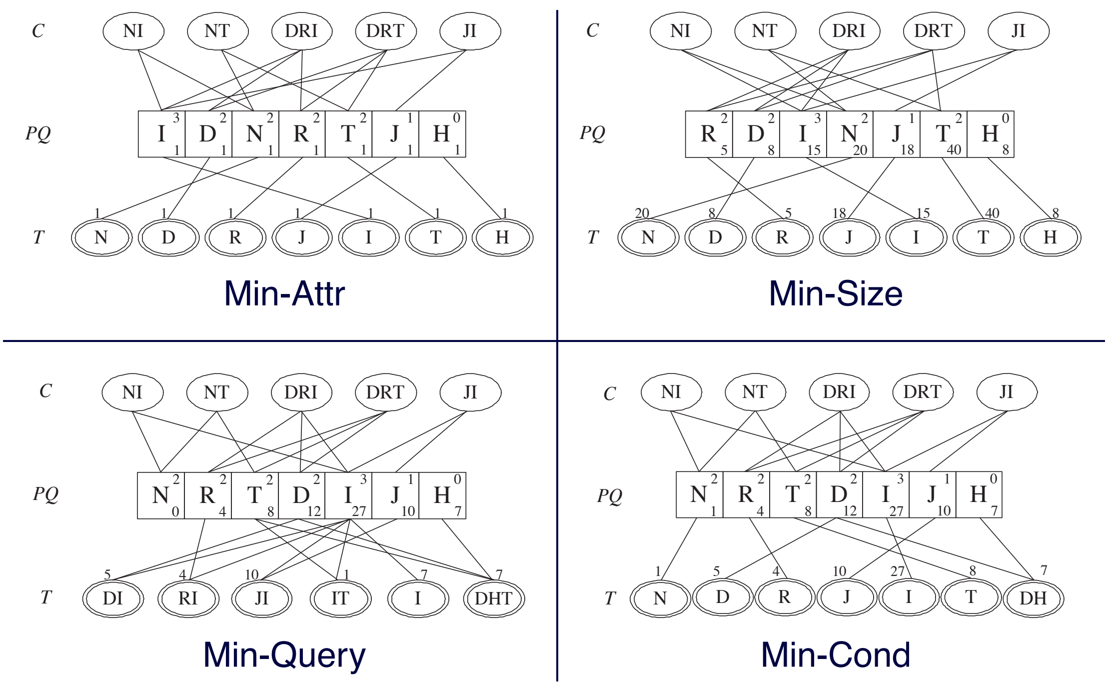
Confidentiality constraints

c_0 ={SSN}
c_1 ={Name,Illness}
c_2 ={Name,Treatment}
c_3 ={DoB,Race,Illness}
c_4 ={DoB,Race,Treatment}
c_5 ={Job,Illness}

A	*size*(*A*)
SSN	9
Name	20
DoB	8
Race	5
Job	18
Illness	15
Treatment	40
HDate	8

q	*freq*(*q*)	*Attr*(*q*)	*Cond*(*q*)
q_1	5	DoB, Illness	⟨Dob⟩, ⟨Illness⟩
q_2	4	Race, Illness	⟨Race⟩, ⟨Illness⟩
q_3	10	Job, Illness	⟨Job⟩, ⟨Illness⟩
q_4	1	Illness, Treatment	⟨Illness⟩, ⟨Treatment⟩
q_5	7	Illness	⟨Illness⟩
q_6	7	DoB, HDate, Treatment	⟨DoB,HDate⟩, ⟨Treatment⟩
q_7	1	SSN, Name	⟨SSN⟩, ⟨Name⟩

Heuristic algorithm – Example of initialization (2)



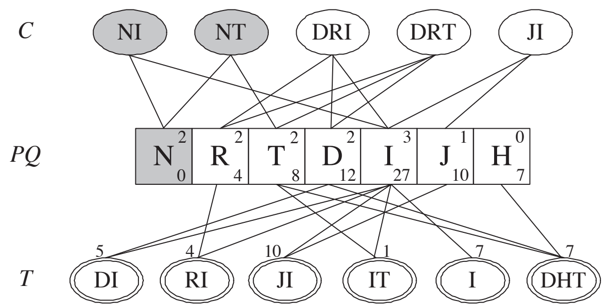
Heuristic algorithm – Working process

- **while** $PQ \neq \emptyset$ and $\exists E \in PQ$, $E.n_c \neq 0$
  - extract the element E with lowest $E.w/E.n_c$ from PQ
  - insert $E.A$ into $\mathscr{H}$
  - $\forall c$ pointed by $E.C$, remove the pointers to c from any element E' in PQ and update $E'.n_c$
  - $\forall t$ pointed by $E.T$, remove the pointers to t from any element E' in PQ and update $E'.w$
  - readjust PQ based on the new values for $E.w/E.n_c$ (*to_be_updated*)

- **for** each $A \in \mathscr{H}$
  - if $\mathscr{H} \setminus \{A\}$ is a hitting set for $\mathscr{C}$, remove A from $\mathscr{H}$

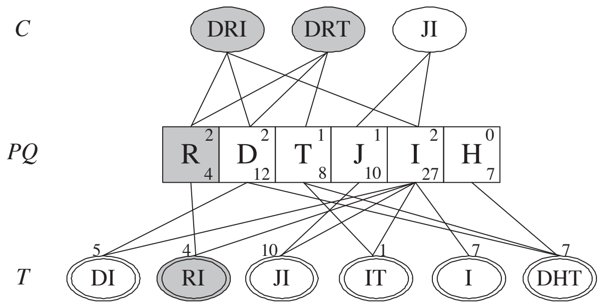
Heuristic algorithm – Example of Min-Query

$\mathscr{H} = \{\}$
$E.A = N$
$E.C = \{NI, NT\}$
$E.T = \{\}$
$to\_be\_updated = \{I, T\}$



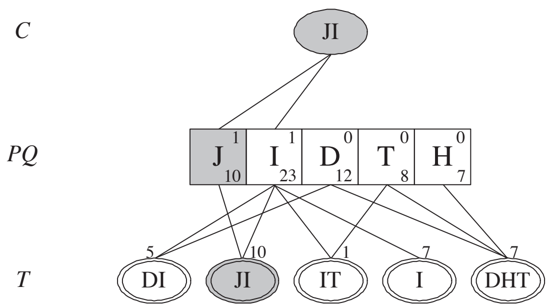
Heuristic algorithm – Example of Min-Query

$\mathscr{H} = \{N\}$
$E.A = R$
$E.C = \{DRI, DRT\}$
$E.T = \{RI\}$
$to\_be\_updated = \{D,I,T\}$



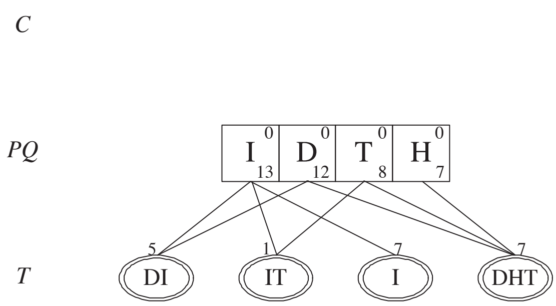
Heuristic algorithm – Example of Min-Query

$\mathscr{H} = \{N,R\}$
$E.A = J$
$E.C = \{JI\}$
$E.T = \{JI\}$
$to\_be\_updated = \{I\}$



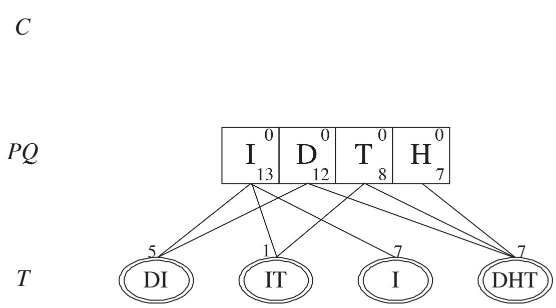
Heuristic algorithm – Example of Min-Query

$\mathscr{H} = \{N,R,J\}$



Heuristic algorithm – Example of Min-Query

$\mathscr{H} = \{N,R,J\}$



F_o={SSN,Name,Race,Job} F_s={Illness,DoB,Treatment,HDate}

Fragments and Loose Associations

Data publication

- Fragmentation can also be used to protect sensitive associations in data publishing
  $\Longrightarrow$ publish/release to external parties only views (fragments) that do not expose sensitive associations

- For increasing utility of published information fragments could be coupled with some associations in sanitized form
  $\Longrightarrow$ loose associations

Loose association

- Publish associations among groups of tuples (in contrast to specific tuples)
- Given two fragments F_l and F_r containing sub-tuples involved in a sensitive association:
  - partition the tuples of F_l and F_r in different groups of size k_l and k_r
  - associations among tuples induce associations among groups

Loose association – Example

SSN	Name	DoB	Race	Illness
123-45-6789	Nancy	65/12/07	white	hypertension
987-65-4321	Ned	73/01/05	black	gastritis
963-85-2741	Nell	86/03/31	asian	flu
147-85-2369	Nick	90/07/19	asian	asthma
782-90-5280	Nancy	55/05/22	white	gastritis
816-52-7272	Noel	32/11/22	black	obesity
872-62-5178	Nora	68/08/14	asian	measles
712-81-7618	Norman	73/01/05	hispanic	hypertension

c_0 ={SSN}
c_1 ={Name,Illness}
c_2 ={Race,DoB,Illness}

F_l

	Name	Race	G
l_1	Nancy	white	nr2
l_2	Ned	black	nr1
l_3	Nell	asian	nr3
l_4	Nick	asian	nr1
l_5	Nancy	white	nr3
l_6	Noel	black	nr2
l_7	Nora	asian	nr4
l_8	Norman	hispanic	nr4

A

G_l	G_r
nr1	id1
nr1	id2
nr2	id1
nr2	id3
nr3	id2
nr3	id4
nr4	id3
nr4	id4

F_r

Illness	DoB	G	
hypertension	65/12/07	id1	r_1
gastritis	73/01/05	id1	r_2
flu	86/03/31	id2	r_3
asthma	90/07/19	id2	r_4
gastritis	55/05/22	id4	r_5
obesity	32/11/22	id3	r_6
measles	68/08/14	id3	r_7
hypertension	73/01/05	id4	r_8

k-loose association (1)

- An association is k-loose if every group association indistinguishably corresponds to at least k distinct associations among tuples
- The degree of looseness characterizes the privacy (and utility) of the associations
  - the probability of an association to exist in the original relation may change from 1/card(relation) to $1/k$
- If grouping satisfies given heterogeneity properties, the group association is guaranteed to be k-loose with $k = k_l \cdot k_r$

k-loose association (2)

- No groups can contain two tuples with the same values for the attributes involved in the sensitive association
  E.g., all groups of the left and right fragment have different values for the attributes appearing in constraints c_2 and c_3
- Groups of the left (right, resp.) fragment have to be associated with different groups of the right (left, resp.) fragment
  E.g., relation A does not contain duplicate tuples
- All groups associated with a same group must have different values for the attributes involved in the sensitive association
  E.g., each group of the left (right, resp.) fragment is associated with groups of the right (left, resp.) fragment that contain tuples with different values for Illness and Race,DoB (Name, resp.)

Research directions

- Balance between encryption and fragmentation
- Schema vs. instance constraints
- Enforcement of different kinds of queries
- Visibility requirements
- Balance privacy and utility
- External knowledge

Conclusions

- The development of the Information technologies presents:
  - new needs and risks for privacy
  - new opportunities for protecting privacy
- Lots of opportunities for new open issues to be addressed

... towards allowing society to fully benefit from information technology while enjoying security and privacy