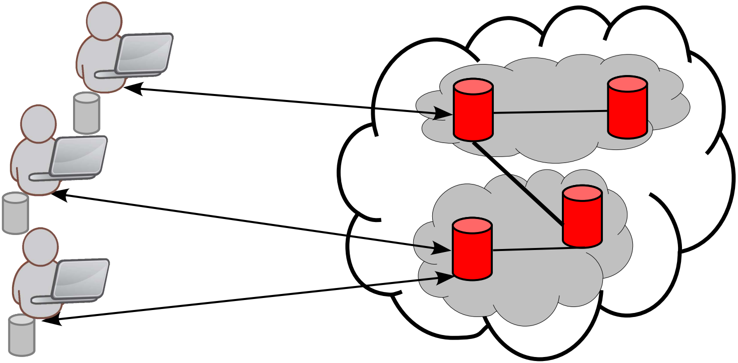
Privacy and Data Protection in Emerging Scenarios

Security, Privacy, and Data Protection Laboratory
Dipartimento di Informatica
Università degli Studi di Milano

Privacy and integrity of data storage



Privacy of users

Privacy and integrity of data storage

Contributions and advancements

The research community has been very active and produced several contributions and advancements. E.g.,:

- Solutions for protecting confidentiality of stored data [ABGGKMSTX-05, CDJJPS-09b, CDFJPS-10, HIML-02]
- Indexes supporting different types of queries [CDDJPS-05, HIML-02, WL-06]
- Inference exposure evaluation [CDDJPS-05]
- Data integrity [S-05, XWYM-07, WYPY-08]
- Selective access to outsourced data [DFJPS-10b]
- ...

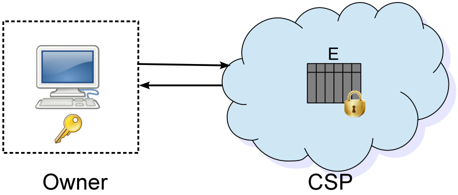
Protecting data confidentiality

- Solutions for protecting data can be based on:
  - encryption
  - encryption and fragmentation
  - fragmentation

Encryption

Encryption

- The server can be honest-but-curious and should not have access to the resource content
- Data confidentiality is provided by wrapping a layer of encryption around sensitive data [HIML-02]
  - for performance reasons, encryption is typically applied at the tuple level

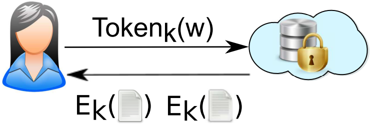


Fine-grained access to data in the cloud

- For confidentiality reasons, CSPs storing data cannot decrypt them for data processing/access
- Need mechanisms to support access to the outsourced data
  - effective and efficient
  - should not open the door to inferences

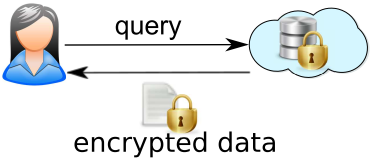
Fine-grained access: Approaches – 1

Keyword-based searches directly on the encrypted data: supported by specific cryptographic techniques (e.g., [CWLRL-11])



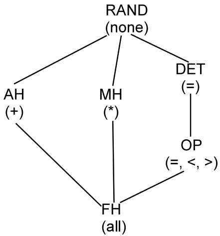
Fine-grained access: Approaches – 2

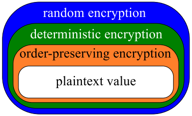
Homomorphic encryption: supports the execution of operations directly on the encrypted data (e.g., [BV11,G-09,GSW13])

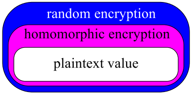


Fine-grained access: Approaches – 3

- Encryption schemas: each column can be encrypted with a different encryption schema, depending on the conditions to be evaluated on it (e.g., Google encrypted BigQuery)
- Onion encryption (CryptDB): different onion layers each of which supports the execution of a specific SQL operation (e.g., HanaDB SEEED framework) [PRZB-11]

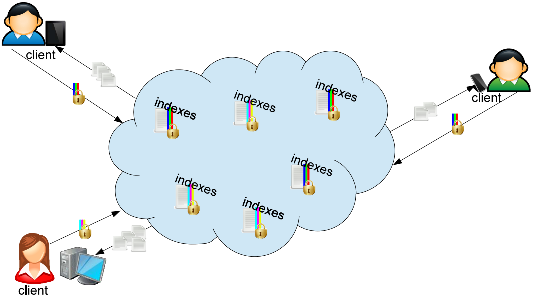






Fine-grained access: Approaches – 4

Indexes: metadata attached to the data and used for fine-grained information retrieval and query execution (e.g., [CDDJPS-05, HIML-02, WL-06])



can also be complementary to encryption (even with encryption users want to have the ability to perform searches based on metadata)

Encryption and indexes – Example

Indexes associated with attributes are used by the server to select data to be returned in response to a query

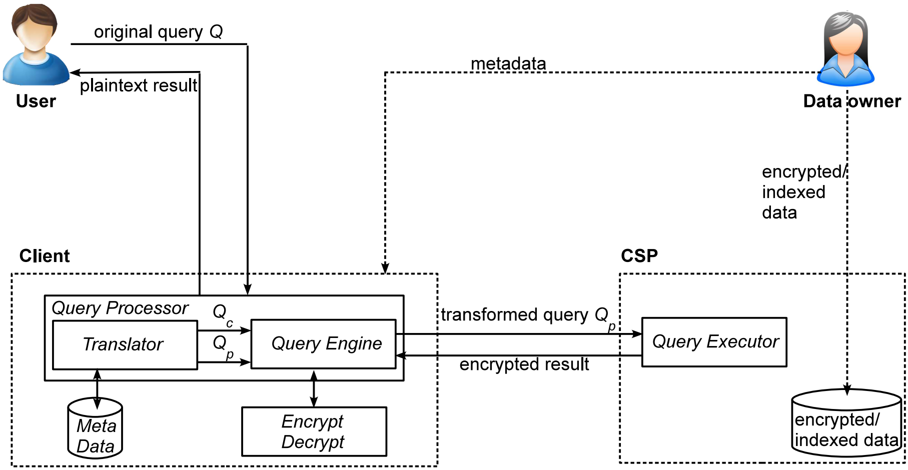
Accounts

Account	**Customer**	**Balance**
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_1^k$

Counter	**Etuple**	$\mathbf{I}_A$	$\mathbf{I}_C$	$\mathbf{I}_B$
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	β	η
4	JpO8eLTVgwV1E	ρ	γ	κ
5	qctG6XnFNDTQc	ς	δ	θ
6	4QbqCeq3hxZHkIU	ι	ε	κ

Query evaluation process



Indexes for queries: Direct (1:1)

Actual value or coding

+ simple and precise for equality queries
– preserves plaintext value distinguishability (inference attacks)

Indexes for queries: Direct (1:1)

Actual value or coding

- \+ simple and precise for equality queries
- – preserves plaintext value distinguishability (inference attacks)

Patients

SSN	Name	Illness	**Doctor**
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients$^{\mathrm{k}}$

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Direct (1:1)

Actual value or coding

- \+ simple and precise for equality queries
- – preserves plaintext value distinguishability (inference attacks)

Patients

SSN	Name	Illness	**Doctor**
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

+ supports for equality queries
+ collisions remove plaintext distinguishability
– result may contain spurious tuples (postprocessing query)
– still vulnerable to inference attacks

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

+ supports for equality queries
+ collisions remove plaintext distinguishability
– result may contain spurious tuples (postprocessing query)
– still vulnerable to inference attacks

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Bucket (n:1)

Partition-based or hash-based

+ supports for equality queries
+ collisions remove plaintext distinguishability
– result may contain spurious tuples (postprocessing query)
– still vulnerable to inference attacks

Patients

SSN	Name	Illness	Doctor
123…89	Alice	Asthma	Angel
234…91	Bob	Asthma	Angel
345…12	Carol	Asthma	Bell
456…23	David	Bronchitis	Clark
567…34	Eva	Gastritis	Dan
232…11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Flattened (1:n)

Flat indexes

+ decreases exposure to inference attacks
– remains vulnerabile to dynamic observations

Indexes for queries: Flattened (1:n)

Flat indexes

+ decreases exposure to inference attacks
– remains vulnerabile to dynamic observations

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Indexes for queries: Flattened (1:n)

Flat indexes

+ decreases exposure to inference attacks
– remains vulnerabile to dynamic observations

Patients

SSN	Name	Illness	Doctor
123...89	Alice	Asthma	Angel
234...91	Bob	Asthma	Angel
345...12	Carol	Asthma	Bell
456...23	David	Bronchitis	Clark
567...34	Eva	Gastritis	Dan
232...11	Eva	Stroke	Ellis

Patients^k

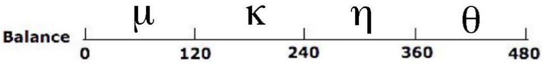
Tid	Etuple	I_S	I_N	I_I	I_D
1	x4Z3tfX2ShOSM	π	κ	α	δ
2	mNHg1oC010p8w	ϖ	ω	α	δ
3	WslaCvfyF1Dxw	ξ	λ	α	ν
4	JpO8eLTVgwV1E	ρ	υ	β	γ
5	qctG6XnFNDTQc	ι	μ	α	σ
6	kotG8XnFNDTaW	χ	o	β	ψ

Partition-based index [HIML-02]

- Consider an arbitrary plaintext attribute A_i in relational schema R, with domain D_i
- D_i is partitioned in a number of non-overlapping subsets of values, called partitions, containing contiguous values
- Given a plaintext tuple t in r, the value of attribute A_i for t belongs to a partition
  - function $\mathit{ident}_{R.A_i}(p_j)$ assigns to each partition p_j of attribute A_i in R an identifier
- The corresponding index value is the unique value associated with the partition to which the plaintext value $t[A_i]$ belongs
  - $\mathit{Map}_{R.A_i}(v)$ = $\mathit{ident}_{R.A_i}(p_j)$, where p_j is the partition containing v
- $\mathit{Map}_{R.A_i}$ can be order-preserving or random

Partition-based index – Example

Random mapping



- $Map_{Balance}(100) = \mu$
- $Map_{Balance}(200) = \kappa$
- $Map_{Balance}(300) = \eta$
- $Map_{Balance}(400) = \theta$

Query conditions supported by the partition-based index

- Support queries where conditions are boolean formulas over terms of the form
  - *Attribute op Value*
  - *Attribute op Attribute*
- Allowed operations for *op* include $\{=, <, >, \leq, \geq\}$

Mapping conditions Map_{cond} – 1

- $A_i = v$. The mapping is defined as:

$$Map_{cond}(A_i = v) \Longrightarrow I_i = Map_{A_i}(v)$$

Example

$$Map_{cond}(\text{Balance} = 100) \Longrightarrow I_{Balance} = Map_{Balance}(100) = \mu$$

- $A_i < v$. The mapping depends on whether or not the mapping function Map_{A_i} is order-preserving or random
  - order-preserving: $Map_{cond}(A_i < v) \Longrightarrow I_i \leq Map_{A_i}(v)$
  - random: check if attribute I_i lies in any of the partitions that may contain a value v' where $v' < v$: $Map_{cond}(A_i < v) \Longrightarrow I_i \in Map^{<}_{A_i}(v)$

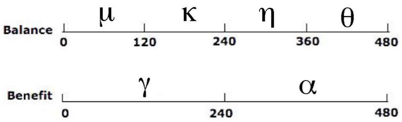
Example

$$Map_{cond}(\text{Balance} < 200) \Longrightarrow I_{Balance} \in \{\mu, \kappa\}$$

- $A_i > v$. Symmetric with respect to $A_i < v$

Mapping conditions Map_{cond} – 2

- $A_i = A_j$. The translation is performed by considering all possible pairs of partitions of A_i and A_j that overlap.
  Example



$$
Map_{cond}(\text{Balance}=\text{Benefit}) \Longrightarrow \quad \begin{aligned} & (I_{Balance}=\mu \wedge I_{Benefit}=\gamma) \\ \vee\ & (I_{Balance}=\kappa \wedge I_{Benefit}=\gamma) \\ \vee\ & (I_{Balance}=\eta \wedge I_{Benefit}=\alpha) \\ \vee\ & (I_{Balance}=\theta \wedge I_{Benefit}=\alpha) \end{aligned}
$$

- $A_i < A_j$. The mapping depends on whether or not the mapping functions Map_{A_i} and Map_{A_j} are order-preserving or random

Query execution

- Each query Q on the plaintext DB is translated into:
  - a query Q_s to be executed at the server
  - a query Q_c to be executed at client on the result
- Query Q_s is defined by exploiting the definition of *Map*$_{cond}(C)$
- Query Q_c is executed on the decrypted result of Q_s to filter out spurious tuples
- The translation should be performed in such a way that the server is responsible for the majority of the work

Query execution – Simple example

Accounts

Account	**Customer**	**Balance**
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_2^k$

Counter	**Etuple**	$\mathbf{I_A}$	$\mathbf{I_C}$	$\mathbf{I_B}$
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

Original query on Accounts	**Translation over Accounts$_2^k$**
Q := SELECT * FROM Accounts WHERE Balance=200	Q$_s$:= SELECT Etuple FROM Accounts$_2^k$ WHERE $I_B = \kappa$ Q$_c$:= SELECT * FROM *Decrypt*(Q$_s$, *Key*) WHERE Balance=200

Query execution – Simple example

Accounts

Account	Customer	Balance
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_2^k$

Counter	Etuple	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

Original query on Accounts	Translation over Accounts$_2^k$
Q := SELECT * FROM Accounts WHERE Balance=200	Q_s := SELECT Etuple FROM Accounts$_2^k$ WHERE $I_B = \kappa$ Q_c := SELECT * FROM *Decrypt*(Q_s, *Key*) WHERE Balance=200

Query execution – Simple example

Accounts

Account	**Customer**	**Balance**
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_2^k$

Counter	**Etuple**	$\mathbf{I_A}$	$\mathbf{I_C}$	$\mathbf{I_B}$
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

Original query on Accounts	**Translation over Accounts$_2^k$**
Q := SELECT * FROM Accounts WHERE Balance$=$200	Q$_s$:= SELECT Etuple FROM Accounts$_2^k$ WHERE $I_B=\kappa$ Q$_c$:= SELECT * FROM *Decrypt*(Q$_s$, *Key*) WHERE Balance$=$200

Query execution – Simple example

Accounts

<u>Account</u>	Customer	Balance
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_2^k$

<u>Counter</u>	<u>Etuple</u>	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

Original query on Accounts	**Translation over Accounts$_2^k$**
Q := SELECT * FROM Accounts WHERE Balance=200	Q_s := SELECT Etuple FROM Accounts$_2^k$ WHERE $I_B=\kappa$ Q_c := SELECT * FROM *Decrypt*(Q_s, *Key*) WHERE Balance=200

Query execution – Simple example

Accounts

Account	Customer	Balance
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts_2^k

Counter	Etuple	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	δ	θ
4	JpO8eLTVgwV1E	ρ	α	κ
5	qctG6XnFNDTQc	ς	β	κ
6	4QbqC3hxZHklU	ι	β	κ

Original query on Accounts	Translation over Accounts_2^k
Q := SELECT * FROM Accounts WHERE Balance=200	Q_s := SELECT Etuple FROM Accounts_2^k WHERE $I_B=\kappa$ Q_c := SELECT * FROM *Decrypt*(Q_s, *Key*) WHERE Balance=200

Hash-based index [CDDJPS-05]

- Based on the concept of one-way hash function
- For each attribute A_i in R with domain D_i, a secure one-way hash function $h : D_i \rightarrow B_i$ is defined, where B_i is the domain of index I_i associated with A_i
- Given a plaintext tuple t in r, the index value corresponding to $t[A_i]$ is $h(t[A_i])$
- Important properties of any secure hash function h are:
  - $\forall x, y \in D_i : \; x = y \implies h(x) = h(y)$ (determinism)
  - given two values $x, y \in D_i$ with $x \neq y$, we may have that $h(x) = h(y)$ (collision)
  - given two distinct but near values x, y ($| x - y | < \varepsilon$) chosen randomly in D_i, the discrete probability distribution of the difference $h(x) - h(y)$ is uniform (strong mixing)

An example of encrypted relation with hashing

Accounts

<u>Account</u>	Customer	Balance
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_2^k$

<u>Enc_tuple</u>	I_A	I_C	I_B
x4Z3tfX2ShOSM	π	α	μ
mNHg1oC010p8w	ϖ	α	κ
WsIaCvfyF1Dxw	ξ	δ	θ
JpO8eLTVgwV1E	ρ	α	κ
qctG6XnFNDTQc	ς	β	κ
4QbqC3hxZHklU	ι	β	κ

- h_c(Alice)=h_c(Chris)=α
- h_c(Donna)=h_c(Elvis)=β
- h_c(Bob)=δ
- h_b(200)=h_b(400)=κ
- h_b(100)=μ
- h_b(300)=θ

Query conditions supported by the hash-based index

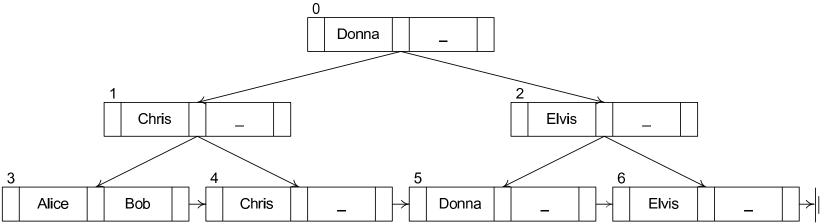
- Support queries where conditions are boolean formulas over terms of the form
  - *Attribute = Value*
  - *Attribute1 = Attribute2*, if *Attribute1* and *Attribute2* are indexed with the same hash function
- It does not support range queries (a solution similar to the one adopted for partition-based methods is not viable)
  - colliding values in general are not contiguous in the plaintext domain
- Query translation works like in the partition-based method

Interval-based queries [CDDJPS-05]

- Order-preserving indexing techniques (e.g., [AKSX-04]): support interval-based queries but expose to inference
  - comparing the ordered sequences of plaintext and indexes would lead to reconstruct the correspondence
- Non order-preserving techniques: data are not exposed to inference but interval-based queries are not supported
- DBMSs support interval-based queries using B+-trees, but the B+-tree defined by the server on indexes is of no use

  Possible solution:
  - Calculate the nodes in the B+-tree at the client and encrypt each node as a whole at the server
  - B+-tree traversal must be performed at the trusted front-end

B+-tree example – 1



B+-tree Table

ID	Node
0	(1,Donna,2,_,_)
1	(3,Chris,4,_,_)
2	(5,Elvis,6,_,_)
3	(Alice,Bob,4)
4	(Chris,_,5)
5	(Donna,_,6)
6	(Elvis,_,_)

Encrypted B+-tree Table

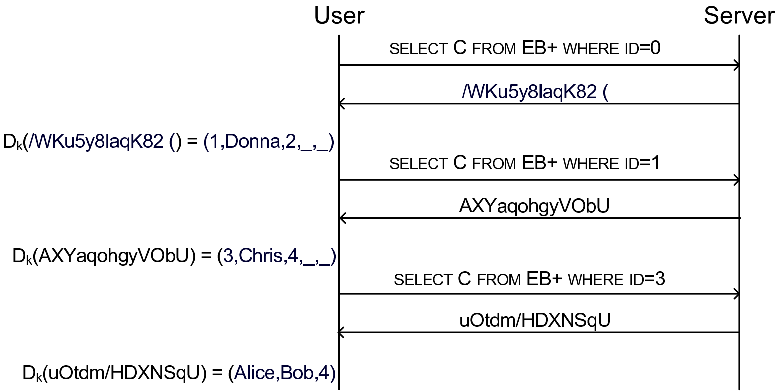
ID	Enc_Node
0	/WKu5y8laqK82(
1	AXYaqohgyVObU
2	IUf7R.PK5h5fU
3	uOtdm/HDXNSqU
4	GLDWRnBGIvYBA
5	a9yl36PA3LeLk
6	H6GwdJpXiU8MY

B+-tree example – 2

Query on the plaintext relation

SELECT * FROM Accounts WHERE Customer = ‘Bob’

Interaction for query evaluation



Searchable encryption

Order preserving encryption

- Order Preserving Encryption Schema (OPES) takes as input a target distribution of index values and applies an order preserving transformation [AKS-04] so that the resulting index values follow the target distribution
  - + comparison can be directly applied on the encrypted data
  - + query evaluation does not produce spurious tuples
  - – vulnerable with respect to inference attacks
- Order Preserving Encryption with Splitting and Scaling (OPESS) schema creates index values so that their frequency distribution is flat [WL-06]

Fully homomorphic encryption [G-09, GKPVZ-13]

Fully homomorphic encryption schema:

- allows performing specific computation on encrypted data
- decryption of the computation result, yields the result of operations performed on the plaintext data

Recent advancement: a functional-encryption schema that fits together several existing schemes (homomorphic encryption, garbled circuit, attribute-based encryption) [GKPVZ-13]

- still too computationally intensive for practical DBMS applications

Inference exposure

A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Modeling and Assessing Inference Exposure in Encrypted Databases," in *ACM TISSEC*, vol. 8, no. 1, February 2005.

Inference exposure

There are two conflicting requirements in indexing data:

- indexes should provide an effective query execution mechanism
- indexes should not open the door to inference and linking attacks

It is important to measure quantitatively the level of exposure due to the publication of indexes:

$$\varepsilon = \text{Exposure Coefficient}$$

Scenarios

The computation of the exposure coefficient ε depends on two factors:

- the indexing method adopted, e.g.,
  - direct encryption
  - hashing
- the a-priori knowledge of the intruder, e.g.,
  - Freq+DB^k:
    - the frequency distribution of plaintext values in the original database (Freq)
    - the encrypted database (DB^k)
  - DB+DB^k:
    - the plaintext database (DB)
    - the encrypted database (DB^k)

Possible inferences

Freq+DB^k

- *plaintext content*: determine the existence of a certain tuple (or *association* of values) in the original database
- *indexing function*: determine the correspondence between plaintext values and indexes

DB+DB^k

- *indexing function*: determine the correspondence between plaintext values and indexes

Exposure coefficient computation [CDDJPS-05]

	Direct Encryption	**Hashing**
Freq+DB^k	Quotient Table	Multiple subset sum problem
DB+DB^k	RCV graph	RCV line graph

Freq+DB^k – Example

Knowledge

Account	Customer	Balance
Acc1	Alice	100
Acc2	Alice	200
Acc3	Bob	300
Acc4	Chris	200
Acc5	Donna	400
Acc6	Elvis	200

Accounts$_1^k$

Counter	Etuple	I_A	I_C	I_B
1	x4Z3tfX2ShOSM	π	α	μ
2	mNHg1oC010p8w	ϖ	α	κ
3	WsIaCvfyF1Dxw	ξ	β	η
4	JpO8eLTVgwV1E	ρ	γ	κ
5	qctG6XnFNDTQc	ς	δ	θ
6	4QbqC3hxZHklU	ι	ε	κ

Inference

- $I_A =$ Account
- $I_C =$ Customer
- $I_B =$ Balance
- $\kappa = 200$ (indexing inference)
- $\alpha =$Alice (indexing inference)
- $\langle$Alice,200$\rangle$ is in the table (association inference)
- Alice is also associated with a value different from 200 (“100,300,400”, all equiprobable)

Direct encryption – Freq+DB^k

- Correspondence between an index and a plaintext value can be determined based on the number of occurrences of the index/value
  - Basic protection: values with the same number of occurrences are indistinguishable to the attacker
- Assessment of index exposure based on equivalence relation where index/plaintext values with same number of occurrences belong to the same class
  - Exposure of values in equivalence class C is $1/|C|$

Freq+DB^k – Example of exposure computation

A.1 = $\{\pi, \varpi, \xi, \rho, \varsigma, \iota\}$ = {Acc1,...,Acc6}

C.1 = $\{\beta, \gamma, \delta, \varepsilon\}$ = {Bob,Chris,Donna,Elvis}

C.2 = $\{\alpha\}$ = {Alice}

B.1 = $\{\mu, \eta, \theta\}$ = {100,300,400}

B.3 = $\{\kappa\}$ = {200}

INDEX_VALUES

I_A	I_C	I_B
π	α	μ
ϖ	α	κ
ξ	β	η
ρ	γ	κ
ς	δ	θ
ι	ε	κ

QUOTIENT

qt_A	qt_C	qt_B
A.1	C.2	B.1
A.1	C.2	B.3
A.1	C.1	B.1
A.1	C.1	B.3
A.1	C.1	B.1
A.1	C.1	B.3

INVERSE CARDINALITY

ic_A	ic_C	ic_B
1/6	1	1/3
1/6	1	1
1/6	1/4	1/3
1/6	1/4	1
1/6	1/4	1/3
1/6	1/4	1

$$\mathscr{E} = \frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{k} \mathrm{IC}_{i,j} = 1/18$$

Direct encryption – DB+DB^k

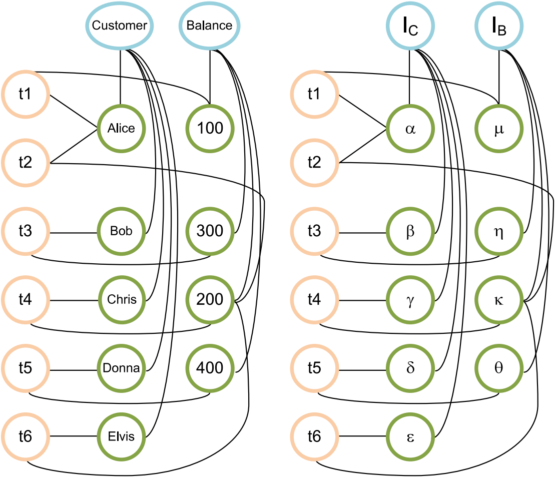
- 3-colored undirected Row-Column-Value graph:
  - one vertex of color “column” for every attribute
  - one vertex of color “row” for every tuple
  - one vertex for every distinct value in a column
  - an arc connects every value to the column and row(s) in which it appears
- RCV on plaintext values is identical to the one on indexes
- Inference exposure can be measured by evaluating the automorphisms of the graph
- Not sufficient to count the number of automorphisms:
  - if there are K automorphisms and in k of them the label assigned to v_i is the same, there is a probability of k/K of identifying the value

DB+DB^k – Example (1)

Customer	Balance
Alice	100
Alice	200
Bob	300
Chris	200
Donna	400
Elvis	200

$\mathbf{I_C}$	$\mathbf{I_B}$
α	μ
α	κ
β	η
γ	κ
δ	θ
ε	κ

DB+DB^k – Example (2)



Inference

- I_C = Customer
- I_B = Balance
- α = Alice
- μ = 100
- κ = 200
- $\{\gamma, \varepsilon\}$ = {Chris,Elvis}
- $\{\langle\beta,\eta\rangle,\langle\delta,\theta\rangle\}$= {⟨Bob,300⟩,⟨Donna,400⟩}

Computing the exposure coefficient

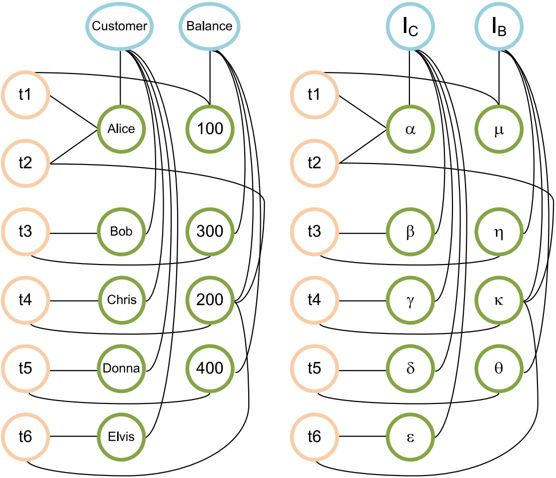
- The set of automorphisms constitutes a group described by the coarsest equitable partition of the vertices:
    - each subset appearing in the partition contains vertices that can be substituted one for the other in an automorphism
- Nauty algorithm: iteratively derives the partition
- Probability of identifying a vertex in partition C: $1/\mid C \mid$

Exposure with equitable partition of n elements over a total number of m: n/m

Example

- β indistinguishable from δ
- η indistinguishable from θ
- γ indistinguishable from ε

Computing the exposure coefficient – Example



Inference

- I_C = Customer
- I_B = Balance
- α = Alice
- μ = 100
- κ = 200
- $\{\gamma, \varepsilon\}$ = {Chris,Elvis}
- $\{\langle\beta,\eta\rangle,\langle\delta,\theta\rangle\}$= $\{\langle$Bob,300$\rangle,\langle$Donna,400$\rangle\}$

Equitable partition: $\{(\alpha),(\beta,\delta),(\gamma,\varepsilon),(\mu),(\eta,\theta),(\kappa)\}$

$\mathscr{E} = 6/9 = 2/3$

Hashing exposure – Freq+DB^k

- The hash function is characterized by a collision factor, denoting the number of attribute values that on average collide on the same index value
- There are different possible mappings of plaintext values in index values, w.r.t. the constraints imposed by frequencies
- Need to enumerate the different mappings by using an adaptation of Pisinger's algorithm for the subset sum problem
- Compute the exposure coefficient for each mapping

Hashing exposure – DB+DB^k

- The RCV-graph built on plaintext and encrypted data are not identical
- Different vertexes of the plaintext RCV-graph may collapse to the same encrypted RCV-graph vertex
- The number of edges connecting row vertexes to value vertexes in the plaintext and encrypted RCV-graph is the same
- The problem becomes finding a correct matching between the edges of the plaintext RCV-graph and the edges of the encrypted RCV-graph

Bloom Filter

Bloom filter [B-70]

A Bloom filter is at the basis of the construction of some indexing techniques. It is an efficient method to encode set membership

- Set of n elements (n is large)
- Vector of l bits (l is small)
- h independent hash functions $H_i : \{0,1\}^* \rightarrow [1,l]$

Insert element x:

- Sets to 1 the bit values at index positions $H_1(x), H_2(x), \ldots, H_h(x)$

Search element x:

- Compute $H_1(x), H_2(x), \ldots, H_h(x)$ and check whether those values are set in the bit vector

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	2	3	4	5	6	7	8	9	10

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

	1			1				1	
1	2	3	4	5	6	7	8	9	10

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	1			1		1		1	
1	2	3	4	5	6	7	8	9	10

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9
- Insert frog: H_1(frog)=1; H_2(frog)=5; H_3(frog)=7

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	1			1		1		1	
1	2	3	4	5	6	7	8	9	10

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9
- Insert frog: H_1(frog)=1; H_2(frog)=5; H_3(frog)=7
- Search dog: H_1(dog)=2; H_2(dog)=5; H_3(dog)=10

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	1			1		1		1	
1	2	3	4	5	6	7	8	9	10

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9
- Insert frog: H_1(frog)=1; H_2(frog)=5; H_3(frog)=7
- Search dog: H_1(dog)=2; H_2(dog)=5; H_3(dog)=10
  $\Longrightarrow$ No

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	1			**1**		1		**1**	
1	2	3	4	5	6	7	8	9	10

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9
- Insert frog: H_1(frog)=1; H_2(frog)=5; H_3(frog)=7
- Search dog: H_1(dog)=2; H_2(dog)=5; H_3(dog)=10
  $\Longrightarrow$ No
- Search car: H_1(car)=1; H_2(car)=5; H_3(car)=9

Bloom filter [B-70] – Example

Let $l = 10$ and $h = 3$

1	2	3	4	5	6	7	8	9	10
1	1			**1**		1		**1**	

- Insert sun: H_1(sun)=2; H_2(sun)=5; H_3(sun)=9
- Insert frog: H_1(frog)=1; H_2(frog)=5; H_3(frog)=7
- Search dog: H_1(dog)=2; H_2(dog)=5; H_3(dog)=10
  $\Longrightarrow$ No
- Search car: H_1(car)=1; H_2(car)=5; H_3(car)=9
  $\Longrightarrow$ Maybe Yes; false positive!

Bloom filter – Properties

- Generalization of hashing (Bloom filter with one hash function is equivalent to ordinary hashing)
  - + space efficient (roughly ten bit for every element in the dictionary with 1% error)
  - – elements cannot be removed
- Yield a constant false positive probability
  - – theoretically considered not acceptable
  - + acceptable in practical applications as fine price to pay for space efficiency

Data Integrity

Integrity of outsourced data

Two aspects:

- Integrity in storage: data must be protected against improper modifications

  $\Longrightarrow$ unauthorized updates to the data must be detected

- Integrity in query computation: query results must be correct and complete

  $\Longrightarrow$ server's misbehavior in query evaluation must be detected

Integrity in storage

- Data integrity in storage relies on digital signatures

- Signatures are usually computed at tuple level
  - table and attribute level signatures can be verified only after downloading the whole table/column
  - cell level signature causes a high verification overhead

- The verification cost grows linearly with the number of tuples in the query result

  $\Longrightarrow$ the signature of a set of tuples can be combined to generate the aggregated signature [MNT-06]

Selective Encryption and Over-Encryption

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Encryption Policies for Regulating Access to Outsourced Data,” in *ACM TODS,* vol. 35, no. 2, April 2010.

Selective information sharing

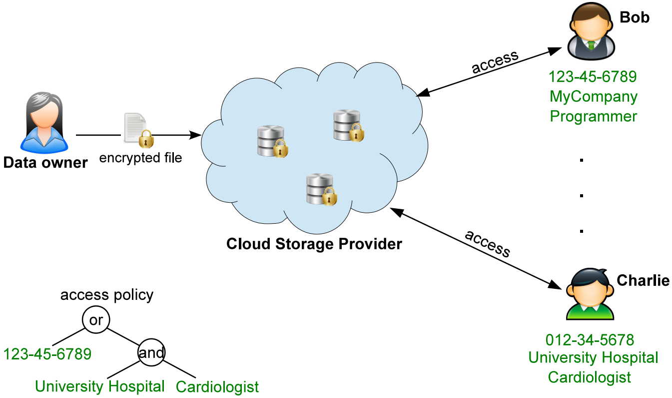
- Different users might need to enjoy different views on the outsourced data
- Enforcement of the access control policy requires the data owner to mediate access requests

  $\Longrightarrow$ impractical (if not inapplicable)
- Authorization enforcement may not be delegated to the provider

  $\Longrightarrow$ data owner should remain in control

Selective information sharing: Approaches – 1

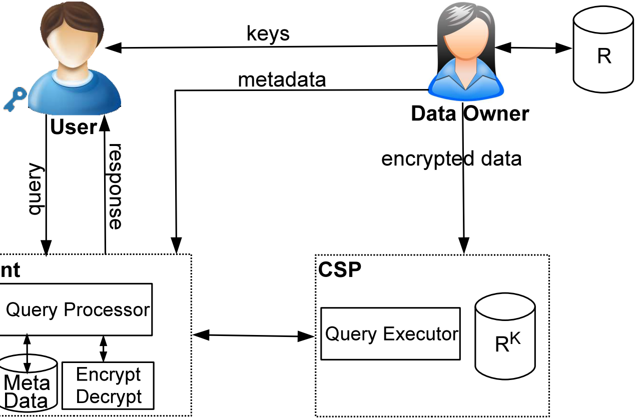
- Attribute-based encryption (ABE): allow derivation of a key only by users who hold certain attributes (based on asymmetric cryptography)



Selective information sharing: Approaches – 2

- Selective encryption: the authorization policy defined by the data owner is translated into an equivalent encryption policy

Selective encryption – Scenario



Selective encryption [DFJPS-10b]

Basic idea/desiderata:

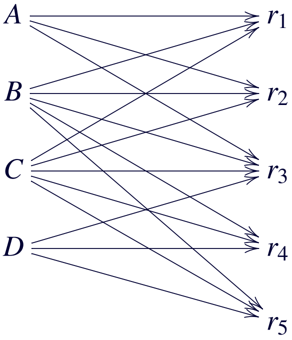
- data themselves need to directly enforce access control
- different keys should be used for encrypting data
- authorization to access a resource translated into knowledge of the key with which the resource is encrypted
- each user is communicated the keys necessary to decrypt the resources she is entailed to access

Authorization policy

- The data owner defines a discretionary access control (authorization) policy to regulate read access to the resources
- An authorization policy $\mathscr{A}$, is a set of permissions of the form $\langle$user,resource$\rangle$.
  It can be represented as:
  - an access matrix
  - a directed and bipartite graph having a vertex for each user u and for each resource r, and an edge from u to r for each permission $\langle u, r\rangle$
- Basic idea:
  - different ACLs implies different encryption keys

Authorization policy – Example

	r_1	r_2	r_3	r_4	r_5
A	1	1	1	0	0
B	1	1	1	1	1
C	1	1	1	1	1
D	0	0	1	1	1



Encryption policy

- The authorization policy defined by the data owner is translated into an equivalent encryption policy

- Possible solutions:
  - encrypt each resource with a different key and give users the keys for the resources they can access
    – requires each user to manage as many keys as the number of resources she is authorized to access
  - use a key derivation method for allowing users to derive from their user keys all the keys that they are entitled to access
    + allows limiting to one the key to be released to each user

Key derivation methods

- Based on a key derivation hierarchy $(\mathscr{K}, \preceq)$
  - $\mathscr{K}$ is the set of keys in the system
  - $\preceq$ partial order relation defined on $\mathscr{K}$
- The knowledge of the key of vertex v_1 and of a piece of information publicly available allows the computation of the key of a lower level vertex v_2 such that $v_2 \preceq v_1$
- $(\mathscr{K}, \preceq)$ can be graphically represented as a graph with a vertex for each $x \in \mathscr{K}$ and a path from x to y iff $y \preceq x$
- Depending on the partial order relation defined on $\mathscr{K}$, the key derivation hierarchy can be:
  - a chain [S-87]
  - a tree [G-80,S-87,S-88]
  - a DAG [AT-83,CMW-06,DFM-04,HL-90,HY-03,LWL-89,M-85,SC-02]

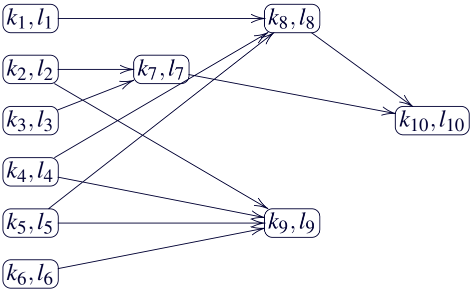
Token-based key derivation methods [AFB-05]

- Keys are arbitrarily assigned to vertices
- A public label l_i is associated with each key k_i
- A piece of public information $t_{i,j}$, called token, is associated with each edge in the hierarchy
- Given an edge (k_i,k_j), token $t_{i,j}$ is computed as $k_j \oplus h(k_i, l_j)$ where
  - $\oplus$ is the n-ary `xor` operator
  - h is a secure hash function
- Advantages of tokens:
  - they are public and allow users to derive multiple encryption keys, while having to worry about a single one
  - they can be stored on the remote server (just like the encrypted data), so any user can access them

Key and token graph

- Relationships between keys through tokens can be represented via a key and token graph
  - a vertex for each pair $\langle k, l \rangle$, where $k \in \mathscr{K}$ is a key and $l \in \mathscr{L}$ the corresponding label
  - an edge from a vertex $\langle k_i, l_i \rangle$ to vertex $\langle k_j, l_j \rangle$ if there exists a token $t_{i,j} \in \mathscr{T}$ allowing the derivation of k_j from k_i

Example



Key assignment and encryption schema

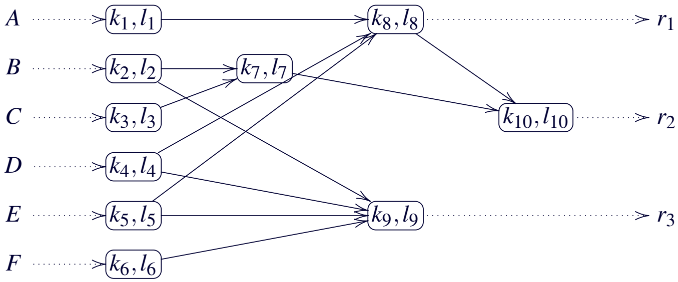
Translation of the authorization policy into an encryption policy:

- Starting assumptions (desiderata):
  - each user can be released only a single key
  - each resource is encrypted only once (with a single key)
- Function ϕ:$\mathscr{U} \cup \mathscr{R} \rightarrow \mathscr{L}$ describes:
  - the association between a user and (the label of) her key
  - the association between a resource and (the label of) the key used for encrypting it

Formal definition of encryption policy

- An encryption policy over users $\mathscr{U}$ and resources $\mathscr{R}$, denoted $\mathscr{E}$, is a 6-tuple $\langle \mathscr{U}, \mathscr{R}, \mathscr{K}, \mathscr{L}, \phi, \mathscr{T} \rangle$, where:
  - $\mathscr{K}$ is the set of keys defined in the system and $\mathscr{L}$ is the set of corresponding labels
  - ϕ is a key assignment and encryption schema
  - $\mathscr{T}$ is a set of tokens defined on $\mathscr{K}$ and $\mathscr{L}$
- The encryption policy can be represented via a graph by extending the key and token graph to include:
  - a vertex for each user and each resource
  - an edge from each user vertex u to the vertex $\langle k, l \rangle$ such that $\phi(u)=l$
  - an edge from each vertex $\langle k, l \rangle$ to each resource vertex r such that $\phi(r) = l$

Encryption policy graph – Example



- user A can access $\{r_1, r_2\}$
- user B can access $\{r_2, r_3\}$
- user C can access $\{r_2\}$
- user D can access $\{r_1, r_2, r_3\}$
- user E can access $\{r_1, r_2, r_3\}$
- user F can access $\{r_3\}$

Policy transformation

Goal: translate an authorization policy $\mathscr{A}$ into an equivalent encryption policy $\mathscr{E}$.

$\mathscr{A}$ and $\mathscr{E}$ are equivalent if they allow exactly the same accesses:

- $\forall u \in \mathscr{U}, r \in \mathscr{R} : u \xrightarrow{\mathscr{E}} r \implies u \xrightarrow{\mathscr{A}} r$
- $\forall u \in \mathscr{U}, r \in \mathscr{R} : u \xrightarrow{\mathscr{A}} r \implies u \xrightarrow{\mathscr{E}} r$

Translating $\mathscr{A}$ into $\mathscr{E}$ – 1

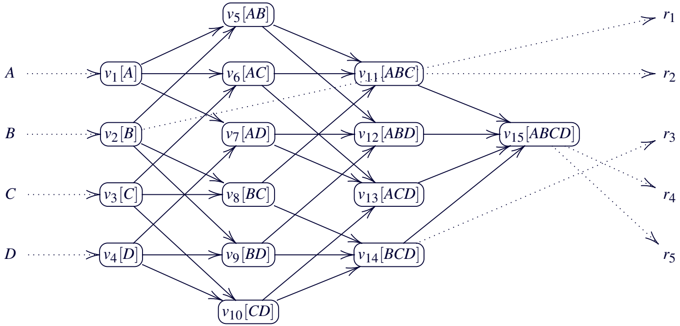
- Naive solution
  - each user is associated with a different key
  - each resource is encrypted with a different key
  - a token $t_{u,r}$ is generated and published for each permission $\langle u,r \rangle$

$\Longrightarrow$ producing and managing a token for each single permission can be unfeasible in practice

- Exploiting acls and user groups
  - group users with the same access privileges
  - encrypt each resource with the key associated with the set of users that can access it

Translating $\mathscr{A}$ into $\mathscr{E}$ – 2

- It is possible to create an encryption policy graph by exploiting the hierarchy among sets of users induced by the partial order relationship based on set containment ($\subseteq$)
- If the system has a large number of users, the encryption policy has a large number of tokens and keys ($2^{|\mathscr{U}|} - 1$)
  $\Longrightarrow$ inefficient key derivation



Minimum encryption policy

- Observation: user groups that do not correspond to any acl do not need to have a key
- Goal: compute a minimum encryption policy, equivalent to a given authorization policy, that minimize the number of tokens to be maintained by the server
- Solution: heuristic algorithm based on the observation that:
    - only vertices associated with user groups corresponding to actual acls need to be associated with a key
    - the encryption policy graph may include only the vertices that are needed to enforce a given authorization policy, connecting them to ensure a correct key derivability
    - other vertices can be included if they are useful for reducing the size of the catalog

Construction of the key and token graph

Start from an authorization policy $\mathscr{A}$

1. Create a vertex/key for each user and for each non-singleton *acl* (initialization)

2. For each vertex v corresponding to a non-singleton *acl*, find a cover without redundancies (covering)
   - for each user u in v.*acl*, find an ancestor v' of v with $u \in v'$.*acl*

3. Factorize common ancestors (factorization)

Key and token graph – Example

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \\ \hline 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

Initialization

$\boxed{v_1[A]}$ $\boxed{v_5[ABC]}$

$\boxed{v_2[B]}$

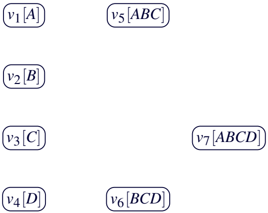
$\boxed{v_3[C]}$ $\boxed{v_7[ABCD]}$

$\boxed{v_4[D]}$ $\boxed{v_6[BCD]}$

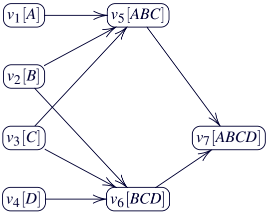
Key and token graph – Example

	r_1	r_2	r_3	r_4	r_5
A	0	1	0	1	1
B	1	1	1	1	1
C	0	1	1	1	1
D	0	0	1	1	1

Initialization

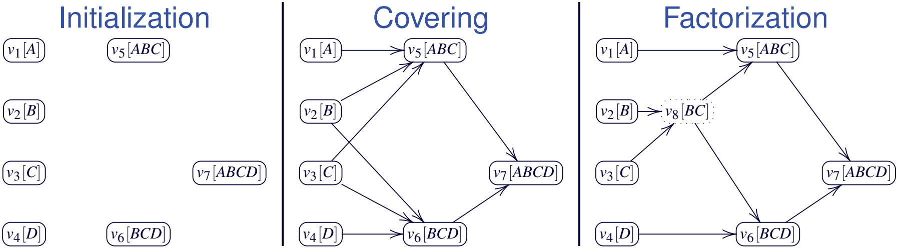


Covering

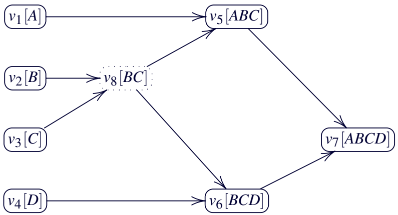


Key and token graph – Example

	r_1	r_2	r_3	r_4	r_5
A	0	1	0	1	1
B	1	1	1	1	1
C	0	1	1	1	1
D	0	0	1	1	1



Key assignment and encryption schema ϕ and catalog



u	$\phi(u)$
A	$v_1.l$
B	$v_2.l$
C	$v_3.l$
D	$v_4.l$

r	$\phi(r)$
r_1	$v_2.l$
r_2	$v_5.l$
r_3	$v_6.l$
r_4,r_5	$v_7.l$

source	destination	token_value
$v_1.l$	$v_5.l$	$t_{1,5}$
$v_2.l$	$v_8.l$	$t_{2,8}$
$v_3.l$	$v_8.l$	$t_{3,8}$
$v_4.l$	$v_6.l$	$t_{4,6}$
$v_5.l$	$v_7.l$	$t_{5,7}$
$v_6.l$	$v_7.l$	$t_{6,7}$
$v_8.l$	$v_5.l$	$t_{8,5}$
$v_8.l$	$v_6.l$	$t_{8,6}$

Multiple owners and policy changes

- When multiple owners need to share their data, the use of a key agreement method allows two data owners to share a secret key for subsequent cryptographic use [DFJPPS-10]
- When authorizations dynamically change, the data owner needs to:
  - download the resource from the server
  - create a new key for the resource
  - decrypt the resource with the old key
  - re-encrypt the resource with the new key
  - upload the resource to the server and communicate the public catalog updates

  $\Longrightarrow$ inefficient
- Possible solution: over-encryption

Over-encryption [DFJPS-07]

- Resources are encrypted twice
  - by the owner, with a key shared with the users and unknown to the server (Base Encryption Layer - BEL level)
  - by the server, with a key shared with authorized users (Surface Encryption Layer - SEL level)
- To access a resource a user must know both the corresponding BEL and SEL keys
- Grant and revoke operations may require
  - the addition of new tokens at the BEL level
  - the update of the SEL level according to the operations performed

BEL and SEL structures

- BEL. At the BEL level we distinguish two kinds of keys: access (k_a) and derivation (k) keys
  - each node in the BEL is associated with a pair of keys (k, k_a), where $k_a = h(k)$, with h a one-way hash function, and a pair of labels (l, l_a)
  - key k (with label l) is used for derivation purpose
  - key k_a (with label l_a) is used to encrypt the resources associated with the node
  - this distinction separates the two roles associated with keys: enabling key derivation and enabling resource access
- SEL. The SEL level is characterized by an encryption policy defined as previously illustrated

Full_SEL and Delta_SEL

- Full_SEL: starts from a SEL identical to the BEL and keeps the SEL always updated to represent the current policy

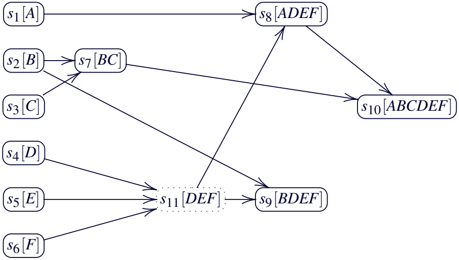
- Delta_SEL: starts from an empty SEL and adds elements to it as the policy evolves, such that the pair BEL-SEL represents the policy

Running example for over-encryption

Access matrix

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
A	0	0	0	0	0	1	1	0	1
B	0	0	1	1	1	0	0	1	1
C	0	0	1	1	1	0	0	0	1
D	1	1	0	0	0	1	1	1	1
E	0	0	0	0	0	1	1	1	1
F	0	0	0	0	0	1	1	1	1

Key and token graph

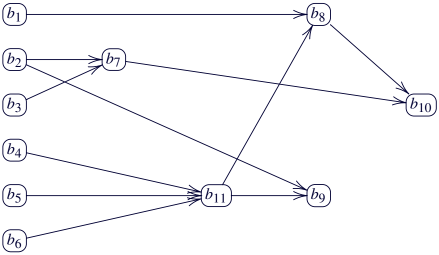


Initial configuration for Full_SEL – Example

BEL

u	$\phi_b(u)$
A	$b_1.l$
B	$b_2.l$
C	$b_3.l$
D	$b_4.l$
E	$b_5.l$
F	$b_6.l$

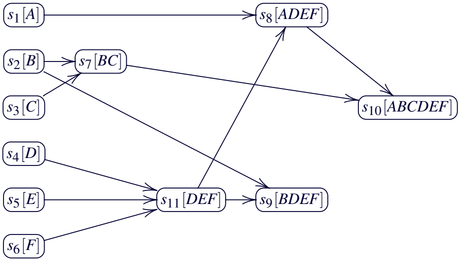
r	$\phi_b(r)$
r_1,r_2	$b_4.l_a$
r_3,r_4,r_5	$b_7.l_a$
r_6,r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



Full_SEL

u	$\phi_s(u)$
A	$s_1.l$
B	$s_2.l$
C	$s_3.l$
D	$s_4.l$
E	$s_5.l$
F	$s_6.l$

r	$\phi_s(r)$
r_1,r_2	$s_4.l$
r_3,r_4,r_5	$s_7.l$
r_6,r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

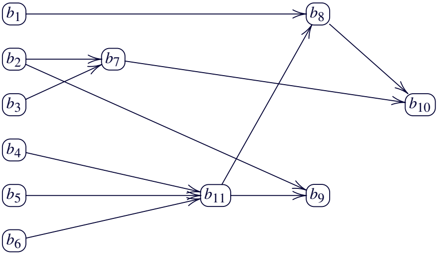


Initial configuration for Delta_SEL – Example

BEL

u	$\phi_b(u)$
A	$b_1.l$
B	$b_2.l$
C	$b_3.l$
D	$b_4.l$
E	$b_5.l$
F	$b_6.l$

r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



Delta_SEL

u	$\phi_s(u)$
A	$s_1.l$
B	$s_2.l$
C	$s_3.l$
D	$s_4.l$
E	$s_5.l$
F	$s_6.l$

r	$\phi_s(r)$
$r_1, \ldots, r_9$	NULL

Algorithms for the evolution of SEL and BEL

- The evolution of the BEL and SEL are managed by:
  - procedure over-encrypt that regulates the update process by over-encrypting the resources at the SEL level
  - grant and revoke procedures that are needed for granting and revoking a privilege, respectively

Procedure over-encrypt (at SEL)

Receive from BEL requests of the form over-encrypt(U,R) to make the set R of resources accessible only to users in U

1. for each resource in R, if currently over-encrypted $\Longrightarrow$ decrypt it;
2. if U = ALL end (no need to do anything);
3. check if $\exists s$ s.t. s.*key* is derivable only by users in U;
   if it does not exist, create it and add it to SEL graph
4. encrypt each resource $r \in R$ with s.*key* and update $\phi_s(r)$ and the corresponding table accordingly

Procedure Grant (at BEL)

Upon request to grant user u access to resource r, currently encrypted with b_j.*key*$_a$

1. add u to *acl*(r)
2. if u cannot derive b_j.*key*$_a$ $\Longrightarrow$ add a token from u's key to b_j.*key*$_a$ in the BEL graph
3. if there is a set R' of resources encrypted with b_j.*key*$_a$ that should not be accessible to u (need to be protected from u at SEL)
   3.1. partition R' in sets according to their acl (each set $S \subseteq R'$ includes all resources with *acl*$_S$)
   3.2. for each set S, request over-encrypt(*acl*$_S$,S) to SEL
4. make r accessible by u at SEL
   - Delta_SEL: if the set of users that can derive b_j.*key*$_a$ is *acl*(r), call over-encrypt(ALL,$\{r\}$); otherwise call over-encrypt(*acl*(r),$\{r\}$)
   - Full_SEL: call over-encrypt(*acl*(r),$\{r\}$)

Procedure Revoke (at BEL)

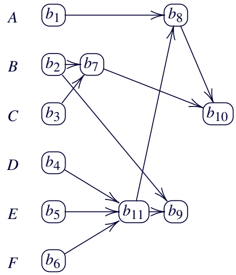
Receive a request to revoke from user u access to resource r

1. remove u from *acl*(r)

2. request over-encrypt(*acl*(r),$\{r\}$) to SEL to make r accessible only to users in *acl*(r)

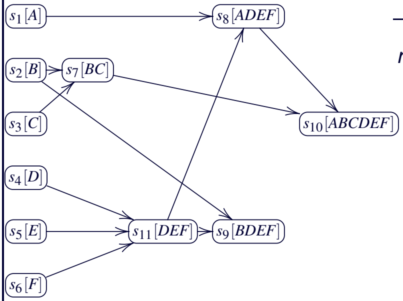
An example of grant operation – Full_SEL

BEL



r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$

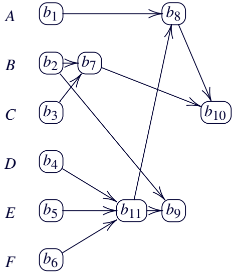
Full_SEL



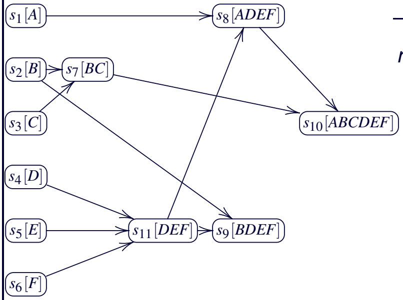
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3, r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Full_SEL

BEL	Full_SEL
grant(D, r_3)	



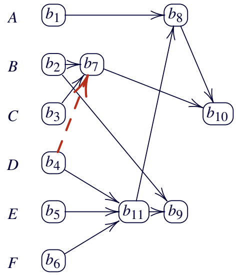
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



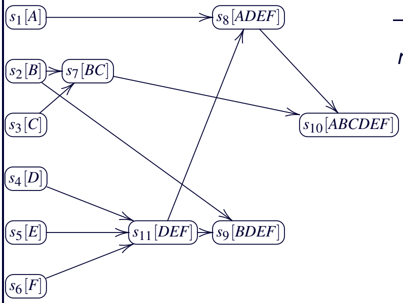
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3, r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Full_SEL

BEL	Full_SEL
grant(D, r_3)	



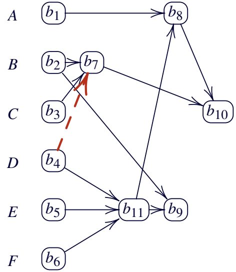
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



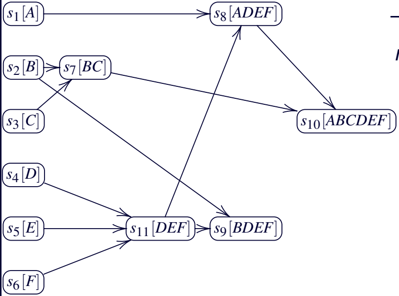
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3, r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Full_SEL

BEL	Full_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$



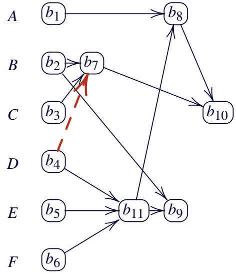
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



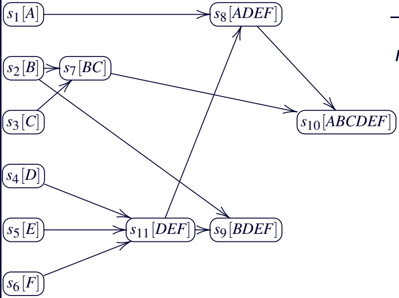
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3, r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Full_SEL

BEL	Full_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$ **over_encrypt**$(BCD, \{r_3\})$



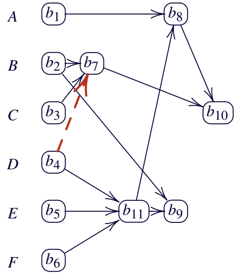
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



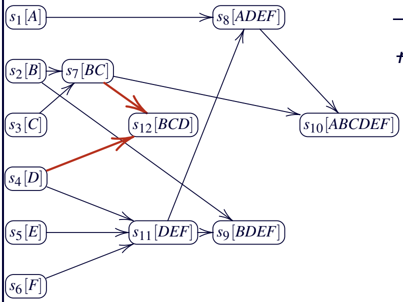
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3, r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Full_SEL

BEL	Full_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$ **over_encrypt**$(BCD, \{r_3\})$



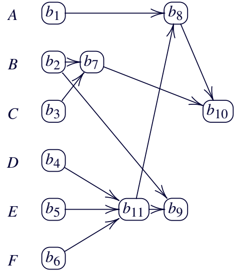
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



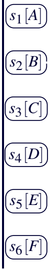
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
$\cancel{r_3}, r_4, r_5$	$s_7.l$
r_3	$s_{12}.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of grant operation – Delta_SEL

BEL	Delta_SEL



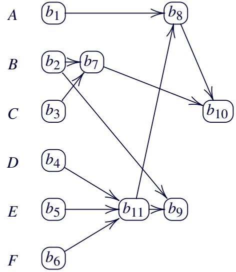
r	$\phi_b(r)$
r_1,r_2	$b_4.l_a$
r_3,r_4,r_5	$b_7.l_a$
r_6,r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



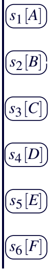
r	$\phi_S(r)$
$r_1,r_2,r_3,r_4,r_5,r_6,r_7,r_8,r_9$	NULL

An example of grant operation – Delta_SEL

BEL	Delta_SEL
grant(D, r_3)	



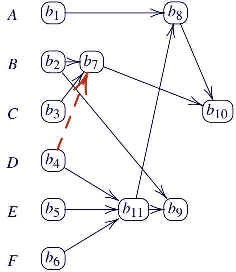
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



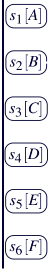
r	$\phi_S(r)$
$r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9$	NULL

An example of grant operation – Delta_SEL

BEL	Delta_SEL
grant(D, r_3)	



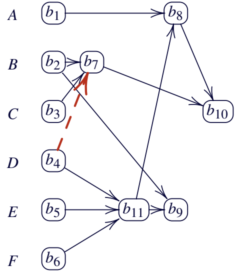
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



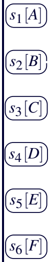
r	$\phi_S(r)$
$r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9$	NULL

An example of grant operation – Delta_SEL

BEL	Delta_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$



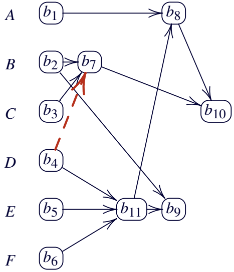
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



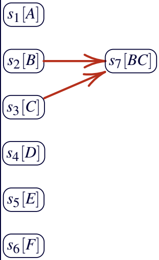
r	$\phi_S(r)$
$r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9$	NULL

An example of grant operation – Delta_SEL

BEL	Delta_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$



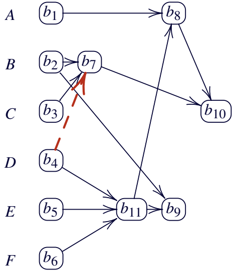
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



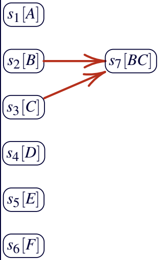
r	$\phi_S(r)$
$r_1, r_2, r_3, \cancel{r_4, r_5,} r_6, r_7, r_8, r_9$	NULL
r_4, r_5	$s_7.l$

An example of grant operation – Delta_SEL

BEL	Delta_SEL
grant(D, r_3)	**over_encrypt**$(BC, \{r_4, r_5\})$ **over_encrypt**(ALL,r_3)



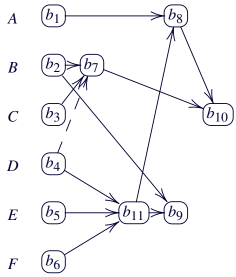
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



r	$\phi_S(r)$
$r_1, r_2, r_3, \cancel{r_4, r_5}, r_6, r_7, r_8, r_9$	NULL
r_4, r_5	$s_7.l$

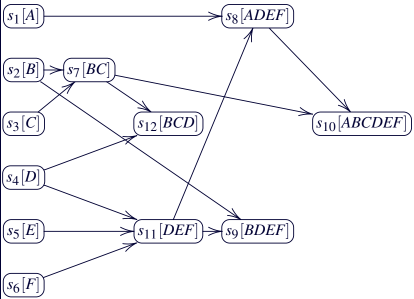
An example of revoke operation – Full_SEL

BEL



r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$

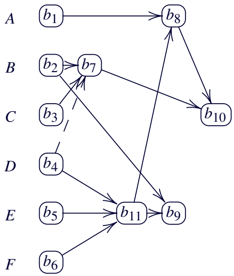
Full_SEL



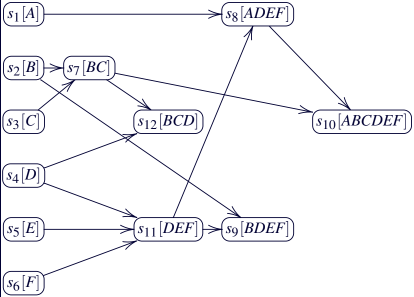
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3	$s_{12}.l$
r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of revoke operation – Full_SEL

BEL	Full_SEL
revoke(F, r_8)	



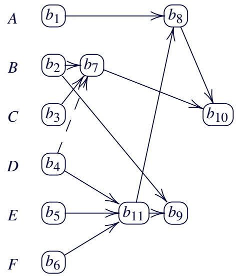
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



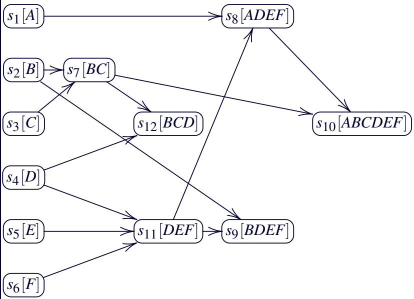
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3	$s_{12}.l$
r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of revoke operation – Full_SEL

BEL	Full_SEL
revoke(F, r_8)	**over_encrypt**$(BDE, \{r_8\})$



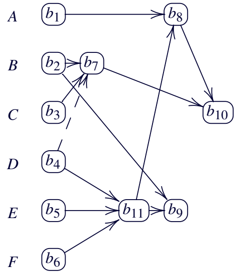
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



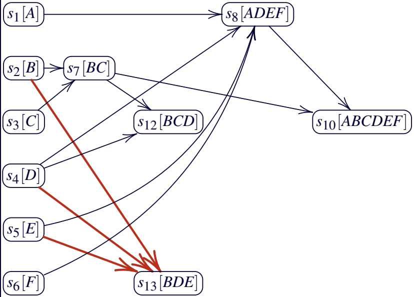
r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3	$s_{12}.l$
r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
r_8	$s_9.l$
r_9	$s_{10}.l$

An example of revoke operation – Full_SEL

BEL	Full_SEL
revoke(F, r_8)	**over_encrypt**$(BDE, \{r_8\})$



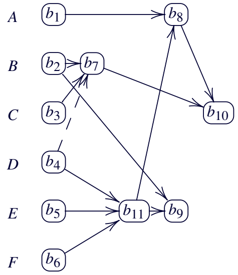
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



r	$\phi_s(r)$
r_1, r_2	$s_4.l$
r_3	$s_{12}.l$
r_4, r_5	$s_7.l$
r_6, r_7	$s_8.l$
$\cancel{r_8}$	$\cancel{s_9.l}$
r_8	$s_{13}.l$
r_9	$s_{10}.l$

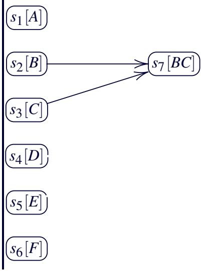
An example of revoke operation – Delta_SEL

BEL



r	$\phi_b(r)$
r_1,r_2	$b_4.l_a$
r_3,r_4,r_5	$b_7.l_a$
r_6,r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$

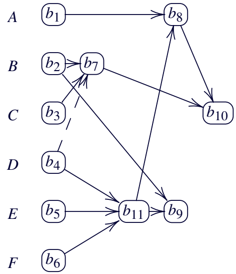
Delta_SEL



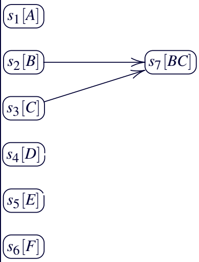
r	$\phi_S(r)$
r_1,r_2,r_3	NULL
r_4,r_5	$s_7.l$
r_6,r_7,r_8,r_9	NULL

An example of revoke operation – Delta_SEL

BEL	Delta_SEL
revoke(F, r_8)	



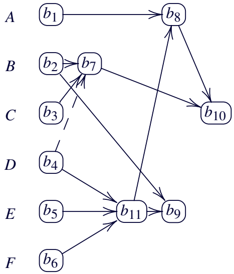
r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$



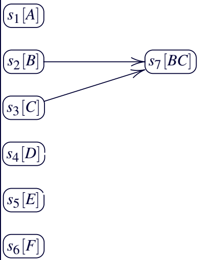
r	$\phi_S(r)$
r_1, r_2, r_3	NULL
r_4, r_5	$s_7.l$
r_6, r_7, r_8, r_9	NULL

An example of revoke operation – Delta_SEL

BEL	Delta_SEL
revoke(F, r_8)	**over_encrypt**$(BDE, \{r_8\})$



r	$\phi_b(r)$
r_1,r_2	$b_4.l_a$
r_3,r_4,r_5	$b_7.l_a$
r_6,r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$

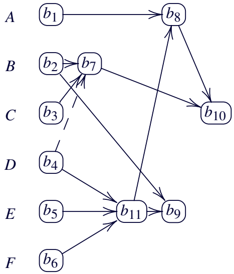


r	$\phi_S(r)$
r_1,r_2,r_3	NULL
r_4,r_5	$s_7.l$
r_6,r_7,r_8,r_9	NULL

An example of revoke operation – Delta_SEL

BEL

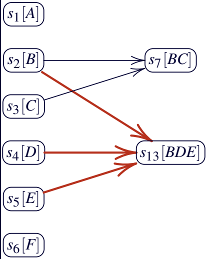
revoke(F, r_8)



r	$\phi_b(r)$
r_1, r_2	$b_4.l_a$
r_3, r_4, r_5	$b_7.l_a$
r_6, r_7	$b_8.l_a$
r_8	$b_9.l_a$
r_9	$b_{10}.l_a$

Delta_SEL

over_encrypt$(BDE, \{r_8\})$



r	$\phi_s(r)$
r_1, r_2, r_3	NULL
r_4, r_5	$s_7.l$
$r_6, r_7, \cancel{r_8}, r_9$	NULL
r_8	$s_{13}.l$

Protection evaluation

- The BEL and SEL encryption policy are equivalent to the authorization policy at initialization time
- Procedure grant, revoke, and over-encryption preserve the equivalence
- The key derivation function adopted is secure
- All the encryption functions and the tokens are robust and cannot be broken
- Each user correctly manages her keys, without the possibility for a user to steal keys from another user
- Vulnerable to collusion?

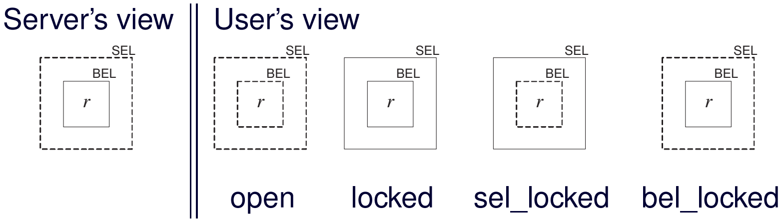
Collusion attacks

- Collusion exists every time two entities combining their knowledge can acquire knowledge that neither of them has access to
  - collusion among users
  - collusion with the server
- Collusion attacks depend on the different views that one can have on a resource r
- We assume users to be not oblivious

Views on resource r – 1

- Four views:
  - open: the user knows the key at the BEL level as well as the key at the SEL level
  - locked: the user knows neither the key at the BEL level nor the key at the SEL level
  - sel_locked: the user knows only the key at the BEL level but does not know the key at the SEL level
  - bel_locked: the user knows only the key at the SEL level but does not know the one at the BEL level
- The server always has the bel_locked view

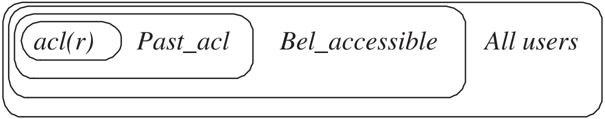
Views on resource r – 2



- Each layer is depicted as a fence
  - discontinuous, if the key is known
  - continuous, if the key is not known (protection cannot be passed)

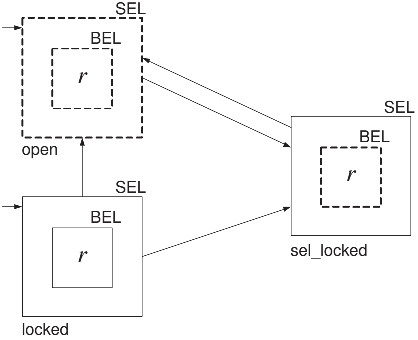
Classification of users

- Consider a resource r and the history of its *acl(r)*
- Users in *acl(r)* can be classified into 4 categories



- Collusion risk for r iff there are users in Bel_accessible that do not belong to Past_acl

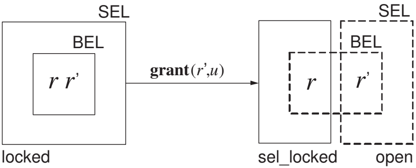
View transitions in the Full_SEL – 1



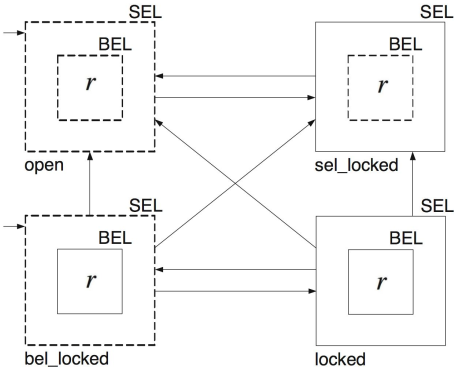
View transitions in the Full_SEL – 2

A user can have the sel_locked view on r due to:

- past acl or
- policy split: u is authorized to access r' (not r), encrypted at the BEL level with the same key as r



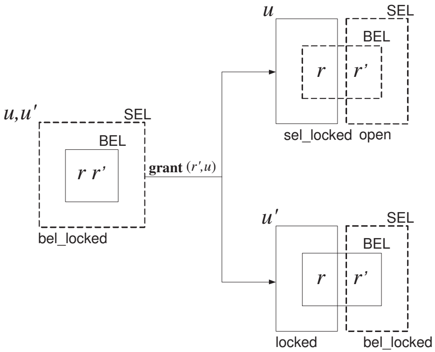
View transitions in the Delta_SEL – 1



View transitions in the Delta_SEL – 2

The view of a user u' on r can evolve from bel_locked to locked due to:

- policy split: u is authorized to access r' (not r), encrypted at the BEL level with the same key as r



Collusion in the Full_SEL

- Collusion among users:
  - not a problem: users never gain in the exchange
- Collusion with the server:
  - users in Bel_accessible who have a sel_locked view and who never had the authorization to access the resource
  - exposure is limited to resources involved in a policy split to make other resources, encrypted with the same BEL key, available to the user
    $\Longrightarrow$ easily identifiable; can be avoided by re-encrypting

Collusion in the Delta_SEL

- A single user by herself can hold the two different views: sel_locked and bel_locked
  - a user could retrieve the resources at initial time, when she is not authorized, getting and storing at her side resources' bel_locked views
  - if the user acquires the sel_locked view on a resource r (the user is released $\phi(r)$ to make accessible to her another resource r') she can enjoy the open view on r
- Again, exposure is limited to resources involved in a policy split $\Longrightarrow$ easily identifiable; can be avoided by re-encrypting

Mix&Slice for Policy Revocation

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, "Mix&slice for Efficient Access Revocation on Outsourced Data," in *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2023.

E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, P. Samarati, "Mix&Slice: Efficient Access Revocation in the Cloud," in *Proc. of the 23rd ACM Conference on Computer and Communications Security (CCS 2016)*, Vienna, Austria, October 2016.

Mix&Slice

- Over-encryption requires support by the server (i.e., the server implements more than simple get/put methods)

- Alternative solution to enforce revoke operations: Mix&Slice

- Use different rounds of encryption to provide complete mixing of the resource

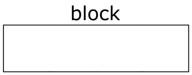
  $\Longrightarrow$ unavailability of a small portion of the encrypted resource prevents its (even partial) reconstruction

- Slice the resource into fragments and, every time a user is revoked access to the resource, re-encrypt a randomly chosen fragment

  $\Longrightarrow$ lack of a fragment prevents resource decryption

Resource organization

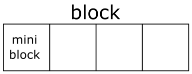
- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits



Resource organization

- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits

- Mini-block: sequence of bits in a block
  it is our atomic unit of protection
  mini-blocks of 32 bits imply a cost of
  2^{32} for brute-force attacks

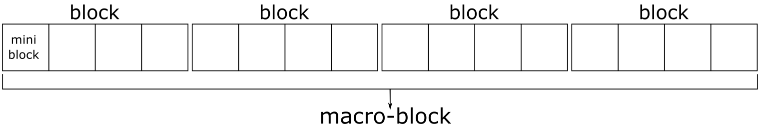


Resource organization

- Block: sequence of bits input to a block cipher
  AES uses block of 128 bits

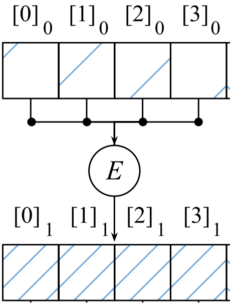
- Mini-block: sequence of bits in a block
  it is our atomic unit of protection
  mini-blocks of 32 bits imply a cost of
  2^{32} for brute-force attacks

- Macro-block: sequence of blocks
  mixing operates at the level of macro-block
  a macro-block of 1KB includes 8 blocks



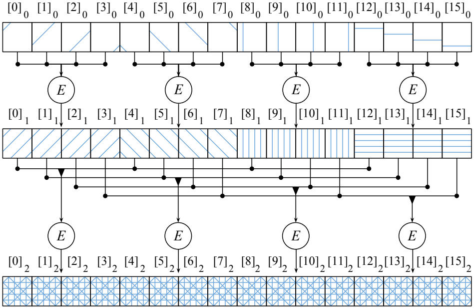
Mixing – 1

- When encryption is applied to a block, all the mini-blocks are mixed
  - \+ absence of a mini-block in a block from the result prevents reconstruction of the block
  - – does not prevent the reconstruction of other blocks in the resource



Mixing – 2

- Extend mixing to a macro-block
  - iteratively apply block encryption
  - at iteration i, each block has a mini-block for each encrypted block obtained at iteration $i-1$ (at distance 4^{i-1})
  - x rounds mix 4^x mini-blocks

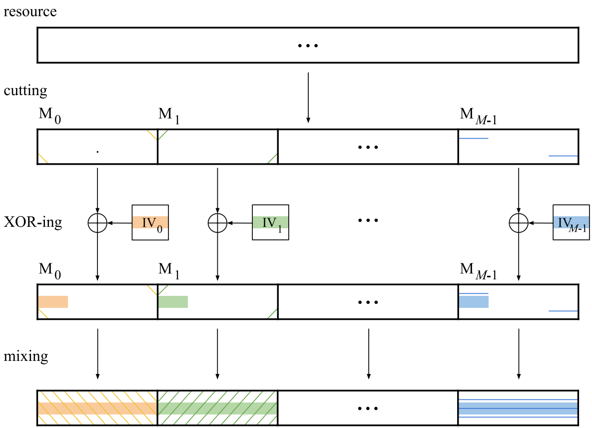


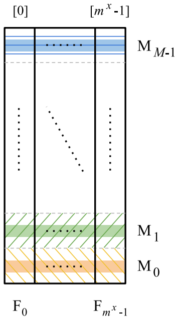
Slicing – 1

- To be mixed, large resources require large macro-blocks
  - many rounds of encryption
  - considerable computation and data transfer overhead
- Large resources are split in different macro-blocks for encryption
- Absence of a mini-block for each macro-block prevents the (even partial) reconstruction of the resource

Slicing – 2

- Slice resources in fragments having a mini-block for each macro-block (the ones in the same position)
  - absence of a fragment prevents reconstruction of the resource

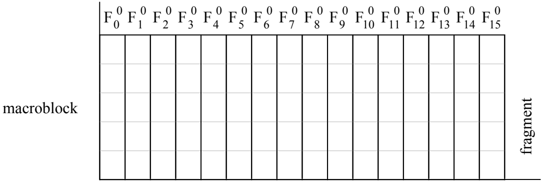




Revoke

To revoke user u access to a resource r

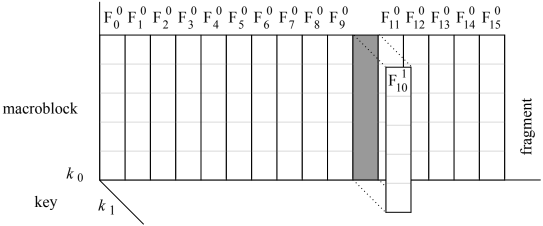
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

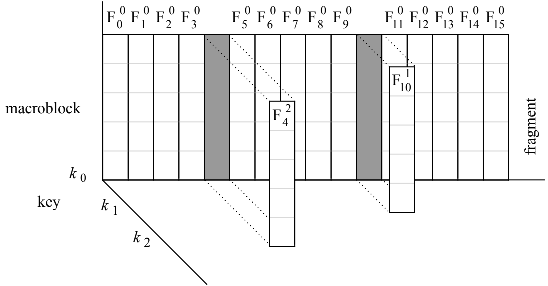
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

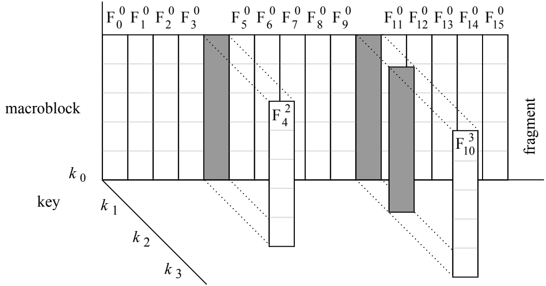
1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Revoke

To revoke user u access to a resource r

1. randomly select a fragment F_i of r and download it
2. decrypt F_i
3. generate a new key k_l that u does not know and cannot derive (generated with key regression and seed encrypted with new ACL)
4. re-encrypt F_i with the new key k_l
5. upload the encrypted fragment



Effectiveness of the approach

- A revoked user does not know the encryption key of at least one fragment
  - a brute force attack is needed to reconstruct the fragment (and the resource)
  - 2^{msize} attempts, with msize the number of bits in a mini-block
- A user can locally store f_{loc} of the f fragments of a resource
  - probability to be able to reconstruct the resource after f_{miss} fragments have been re-encrypted: $P = (f_{\text{loc}}/f)^{f_{\text{miss}}}$
    - proportional to the number of locally stored fragments
    - decreases exponentially with the number of policy updates

Write Authorizations

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Support for Write Privileges on Outsourced Data,” in *Proc. of SEC*, Heraklion, Crete, Greece, June 2012.

Write authorizations

Problem:

- The support of only read accesses may be limiting

  $\Longrightarrow$ users may be authorized to modify resources

- Keys regulating read accesses cannot regulate write accesses

  $\Longrightarrow$ the set $w[o]$ of users authorized to write o may be a subset of the set $r[o]$ of users authorized to read o

Solution: associate a write tag $tag[o]$ with each resource o encrypted with a key

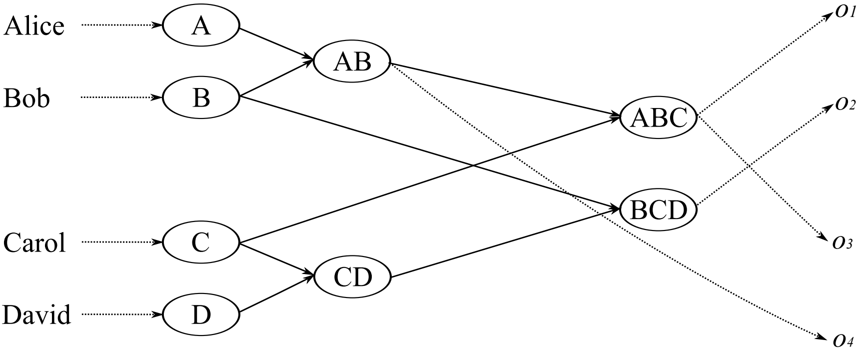
- known to the users in $w[o]$ (derivable from the key of $w[o]$ via secure hashing)
- known to the storage server (derivable from its key via tokens)

$\Longrightarrow$ write authorized iff u proves knowledge of $tag[o]$ to the server

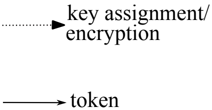
Key derivation graph

- Key derivation graph extended with the storage server S
- The key derivation graph has
  - a key k_u for each user u
  - a key k_S for the storage server S
  - a key $k_{r[o]}$ for each read access control list $r[o]$
  - a key $k_{w[o]}$ for each write access control list $w[o]$
  - a key $k_{w[o]\cup\{S\}}$ for each write access control list, extended with the server $w[o]\cup\{S\}$
  - a secure hash function h to compute $k_{w[o]\cup\{S\}}$ from $k_{w[o]}$
  - a set of tokens that permit each user u to derive $k_{r[o]}$ ($k_{w[o]}$) s.t. $u\in r[o]$ ($u\in w[o]$)
  - a set of tokens that permit the storage server S to derive $k_{w[o]\cup\{S\}}$

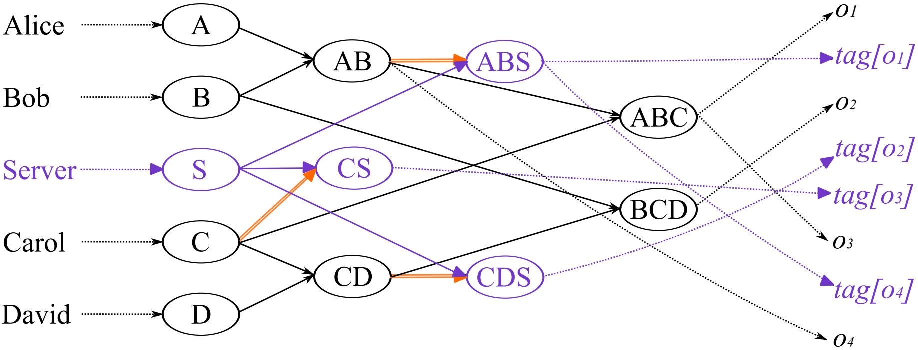
Key derivation graph – Example



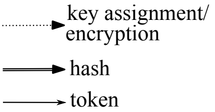
o	$r[o]$
o_1	ABC
o_2	BCD
o_3	ABC
o_4	AB



Key derivation graph – Example



o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB



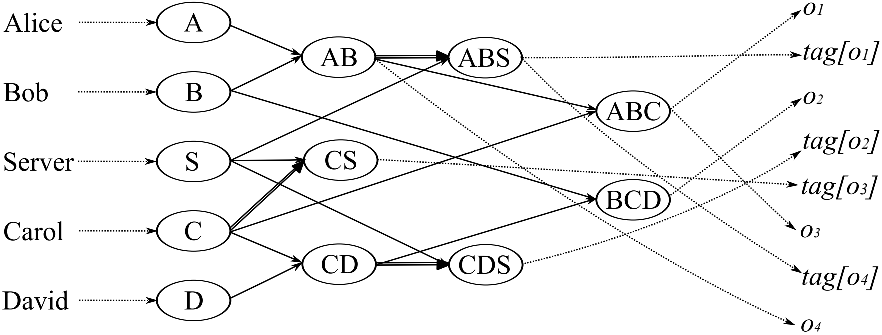
Authorization enforcement

- The data owner defines the key derivation graph and
  - communicates to each user u key k_u
  - communicates to the storage server S key k_S
  - encrypts each resource o with key $k_{r[o]}$
  - encrypts the write tag $tag[o]$ of each resource o with key $k_{w[o] \cup \{S\}}$
- Read accesses
  - u can read o iff she can decrypt its content (i.e., if $u \in r[o]$)
- Write accesses
  - u sends a request to write o to the storage server
  - the server accepts the request only if u provides (plaintext) $tag[o]$
  - u can provide $tag[o]$ only if u can decrypt it (i.e., if $u \in w[o]$)

Structure of outsourced resources

METADATA	r_label	$l_{r[o]}$	label of the key used for o
	w_label	$l_{w[o] \cup \{S\}}$	label of the key used for $tag[o]$
	o_id	*o_id*	object identifier
	encw_tag	$E(tag[o], k_{w[o] \cup \{S\}})$	encrypted write tag
RESOURCE	encr_resource	$E(o, k_{r[o]})$	encrypted resource

Authorization enforcement – Example

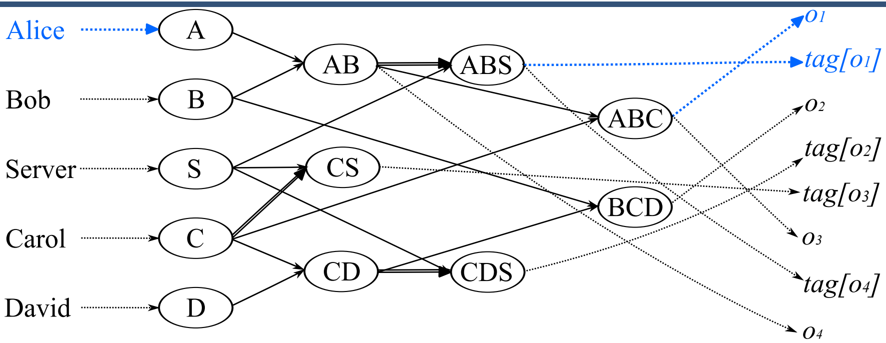


o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	w_label	o_id	encw_tag	encr_resource
l_{ABC}	l_{ABS}	1	α	zKZlJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AImXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

l_i	l_j	$t_{i,j}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
l_S	l_{ABS}	$k_{ABS} \oplus h(k_S, l_{ABS})$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Authorization enforcement – Example

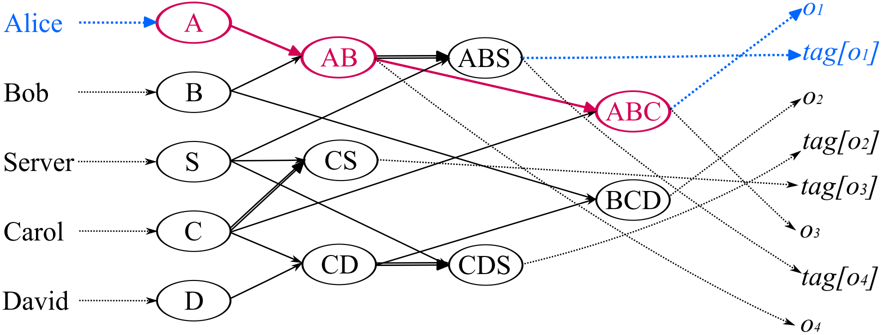


o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	**w_label**	**o_id**	**encw_tag**	**encr_resource**
l_{ABC}	l_{ABS}	1	α	zKZlJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AImXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

l_i	l_j	$t_{i,j}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
l_S	l_{ABS}	$k_{ABS} \oplus h(k_S, l_{ABS})$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Authorization enforcement – Example

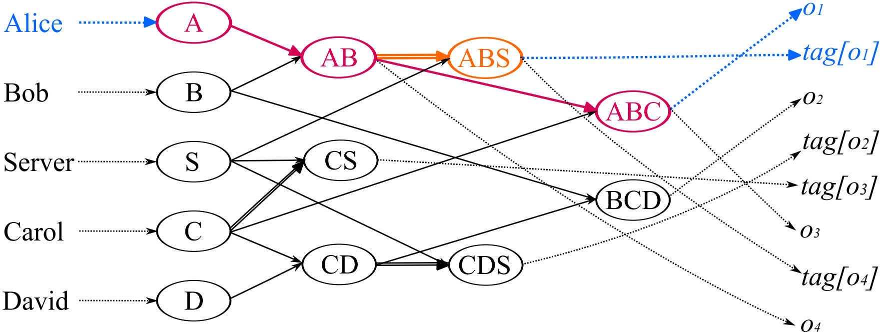


o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	w_label	o_id	encw_tag	encr_resource
l_{ABC}	l_{ABS}	1	α	zKZlJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AlmXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

l_i	l_j	$t_{i,j}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
l_S	l_{ABS}	$k_{ABS} \oplus h(k_S, l_{ABS})$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Authorization enforcement – Example

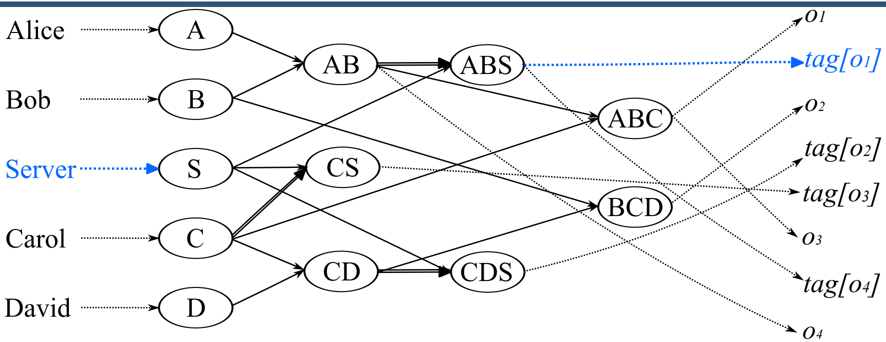


o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	w_label	o_id	encw_tag	encr_resource
l_{ABC}	l_{ABS}	1	α	zKZIJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AImXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

l_i	l_j	$t_{i,j}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
l_S	l_{ABS}	$k_{ABS} \oplus h(k_S, l_{ABS})$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Authorization enforcement – Example

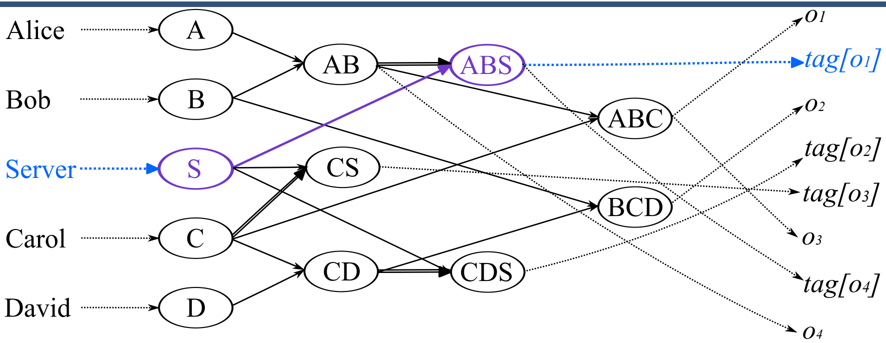


o	$r[o]$	$w[o]$
o_1	ABC	AB
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	w_label	o_id	encw_tag	encr_resource
l_{ABC}	l_{ABS}	1	α	zKZlJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AImXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

l_i	l_j	$t_{i,j}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
l_S	l_{ABS}	$k_{ABS} \oplus h(k_S, l_{ABS})$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Authorization enforcement – Example



o	$r[o]$	$w[o]$
$\boldsymbol{o_1}$	ABC	$\boldsymbol{AB}$
o_2	BCD	CD
o_3	ABC	C
o_4	AB	AB

r_label	w_label	o_id	encw_tag	encr_resource
l_{ABC}	$\boldsymbol{l_{ABS}}$	**1**	α	zKZlJxVcCrC0g
l_{BCD}	l_{CDS}	2	β	t9qdJqC7AlmXU
l_{ABC}	l_{CS}	3	γ	AxalPH8Kv37Ts
l_{AB}	l_{ABS}	4	δ	xwfPJSLn.MVqY

$\mathbf{l_i}$	$\mathbf{l_j}$	$\mathbf{t_{i,j}}$
l_A	l_{AB}	$k_{AB} \oplus h(k_A, l_{AB})$
l_B	l_{AB}	$k_{AB} \oplus h(k_B, l_{AB})$
l_B	l_{BCD}	$k_{BCD} \oplus h(k_B, l_{BCD})$
l_C	l_{ABC}	$k_{ABC} \oplus h(k_C, l_{ABC})$
l_C	l_{CD}	$k_{CD} \oplus h(k_C, l_{CD})$
l_D	l_{CD}	$k_{CD} \oplus h(k_D, l_{CD})$
l_{AB}	l_{ABC}	$k_{ABC} \oplus h(k_{AB}, l_{ABC})$
l_{CD}	l_{BCD}	$k_{BCD} \oplus h(k_{CD}, l_{BCD})$
l_S	l_{CS}	$k_{CS} \oplus h(k_S, l_{CS})$
$\boldsymbol{l_S}$	$\boldsymbol{l_{ABS}}$	$\boldsymbol{k_{ABS} \oplus h(k_S, l_{ABS})}$
l_S	l_{CDS}	$k_{CDS} \oplus h(k_S, l_{CDS})$

Write integrity

- The data owner needs to verify the proper behavior of users and storage server
- Write integrity control
  - allows detecting resource tampering
  - discourages improper behaviors
  - provides non repudiation
- Straightforward solution: signature-based approach
  - users sign the resource with their private key
  - the data owner checks if the signature has been produced by an authorized user for the resource content

$\Longrightarrow$ it is computationally expensive

HMAC-based approach

- Each resource o has
  - a timestamp, encw_ts, of the last write operation
  - a user_tag computed as the HMAC, with the key k_u of the writer, over o, the old value of the user_tag, and the timestamp of the write operation
  - a group_tag computed as the HMAC, with key $k_{w[o]}$, over o and the timestamp of the write operation
- At each write operation, the writer updates the user_tag and group_tag
- Aggregated signature guarantees metadata integrity and that no resource is dropped

Integrity tags

- User_tag of resource o
  - write integrity and accountability of user actions
  - checked only by the data owner
- Group_tag of resource o
  - write integrity of the resource content
  - checked by all the users in $w[o]$
- Permit to detect
  - tampering by S with $o \Longrightarrow S$ cannot produce a valid user_tag for o
  - tampering by S with $tag[o]$ to include u in $w[o] \Longrightarrow u$ cannot produce valid integrity tags
  - unauthorized write operations by $u \Longrightarrow u$ cannot produce valid integrity tags

Structure of outsourced resources

METADATA	r_label	$l_{r[o]}$	label of the key used for o
	w_label	$l_{w[o] \cup \{S\}}$	label of the key used for $tag[o]$
	o_id	*o_id*	object identifier
	encw_tag	$E(tag[o], k_{w[o] \cup \{S\}})$	encrypted write tag
RESOURCE	encr_resource	$E(o, k_{r[o]})$	encrypted resource
WRITE INTEGRITY	encw_ts	$E(ts, k_{w[o] \cup \{S\}})$	timestamp
	user_tag	HMAC$(o \| \| u_t' \| \| ts, k_u)$	tag for the owner
	group_tag	HMAC$(o \| \| ts, k_{w[o]})$	tag for writers

Other issues

- Write integrity controlled by any reader
- Support for write privileges for data collections with multiple owners
- Selective encryption for supporting subscription-based authorization policies [DFJL-12]
  - users are authorized to access all and only the resources published during their subscribed periods
  - user authorizations remain valid also after the expiration of their subscriptions
    $\Longrightarrow$ need to take into account both the subscriptions of the users and the time when resources have been published

Fragmentation and Encryption

Fragmentation and encryption

- Encryption makes query evaluation and application execution 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

- Recent solutions for enforcing privacy requirements couple:
  - encryption
  - data fragmentation

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 of some attributes are considered sensitive and should not be visible
  $\Longrightarrow$ singleton constraints
- Sensitive associations: the associations among values of given attributes are sensitive and should not be visible
  $\Longrightarrow$ non-singleton constraints

Confidentiality constraints – Example

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

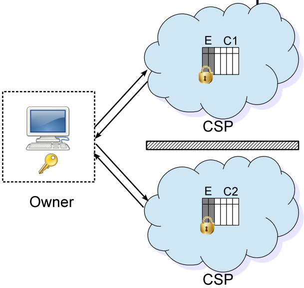
- {Telephone}, {Email}
  - attributes Telephone and Email are sensitive (cannot be stored in the clear)
- {Name,Salary}, {Name,Position}, {Name,DoB}
  - attributes Salary, Position, and DoB are private of an individual and cannot be stored in the clear in association with the name
- {DoB,Gender,Zip,Salary}, {DoB,Gender,Zip,Position}
  - attributes DoB, Gender, Zip can work as quasi-identifier
- {Position,Salary}, {Salary,DoB}
  - association rules between Position and Salary and between Salary and DoB need to be protected from an adversary

Outline

- Data fragmentation
  - Non-communicating pair of servers [ABGGKMSTX-05]
  - Multiple non-linkable fragments [CDFJPS-07,CDFJPS-10]
  - Departing from encryption: Keep a few [CDFJPS-09b]
  - Fragmentation and inferences [DFJLPS-14]
- Publishing obfuscated associations
  - Anonymizing bipartite graph [CSYZ-08]
  - Fragments and loose associations [DFJPS-10]

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) [ABGGKMSTX-05]
  - Sensitive associations are protected by distributing the attributes among the two servers
  - Encryption is applied only when explicitly demanded by the confidentiality constraints or when storing an attribute in any of the two 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 $c \in \mathscr{C}$, $c \nsubseteq (R_1 - E)$ and $c \nsubseteq (R_2 - E)$

Non-communicating pair of servers – Example

PATIENTS

	<u>SSN</u>	Name	YoB	Job	Disease
t_1	123456789	Alice	1980	Clerk	Asthma
t_2	234567891	Bob	1980	Doctor	Asthma
t_3	345678912	Carol	1970	Nurse	Asthma
t_4	456789123	David	1970	Lawyer	Bronchitis
t_5	567891234	Eva	1970	Doctor	Bronchitis
t_6	678912345	Frank	1960	Doctor	Gastritis
t_7	789123456	Gary	1960	Teacher	Gastritis
t_8	891234567	Hilary	1960	Nurse	Diabetes

c_0 = {SSN}
c_1 = {Name, Disease}
c_2 = {Name, Job}
c_3 = {Job, Disease}

F_1

<u>tid</u>	Name	YoB	SSN^k	Disease^k
1	Alice	1980	jdkis	hyaf4k
2	Bob	1980	u9hs9	j97;qx
3	Carol	1970	j9und	9jp‘md
4	David	1970	p0vp8	p;nd92
5	Eva	1970	8nn[	0-mw-n
6	Frank	1960	j9jMK	wqp9[i
7	Gary	1960	87l’D	L0MB2G
8	Hilary	1960	8pm}n	@h8hwu

F_2

<u>tid</u>	Job	SSN^k	Disease^k
1	Clerk	uwq8hd	jsd7ql
2	Doctor	j-0.dl;	0],nid
3	Nurse	8ojqdkf	j-0/?n
4	Lawyer	j0i12nd	5lkdpq
5	Doctor	mj[9;’s	j0982e
6	Doctor	aQ14l[	jnd%d
7	Teacher	8qsdQW	OP[’
8	Nurse	0890UD	UP0D@

Query execution

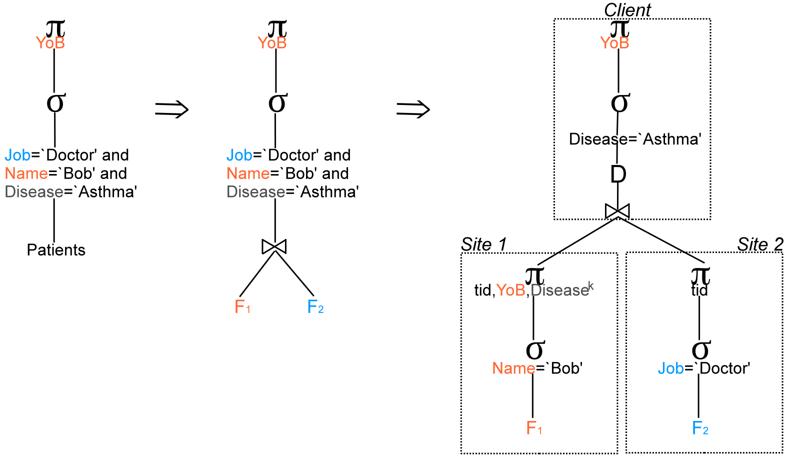
At the logical level: replace R with $R_1 \bowtie R_2$

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 cannot be pushed down on encrypted attributes
  - 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

- F_1: (tid,Name,YoB,SSN^k,Disease^k)
- F_2: (tid,Job,SSN^k,Disease^k)



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 vertexes
- 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 vertexes

Identifying the optimal decomposition – 4

Find a 2-coloring of the vertexes 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:

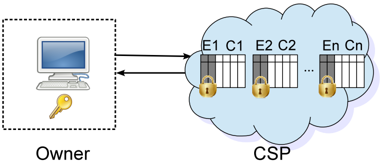
- approximate min-cuts
- approximate weighted set cover

Multiple non-linkable fragments – 1

Coupling fragmentation and encryption is interesting and provides advantages, but assumption of two non-communicating 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 [CDFJPS-10]



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

Multiple non-linkable 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 non-linkable fragments – 3

- Each fragment F is mapped into 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 into a tuple $t^e \in f_i^e$ where f_i^e is a relation over F_i^e and:
    - $t^e[enc] = E_k(t[R - F_i] \otimes t^e[salt])$
    - $t^e[A_{i_j}] = t[A_{i_j}]$, for $j = 1, \ldots, n$

Multiple non-linkable fragments – Example

PATIENTS

	SSN	Name	YoB	Job	Disease
t_1	123456789	Alice	1980	Clerk	Asthma
t_2	234567891	Bob	1980	Doctor	Asthma
t_3	345678912	Carol	1970	Nurse	Asthma
t_4	456789123	David	1970	Lawyer	Bronchitis
t_5	567891234	Eva	1970	Doctor	Bronchitis
t_6	678912345	Frank	1960	Doctor	Gastritis
t_7	789123456	Gary	1960	Teacher	Gastritis
t_8	891234567	Hilary	1960	Nurse	Diabetes

c_0 = {SSN}
c_1 = {Name, Disease}
c_2 = {Name, Job}
c_3 = {Job, Disease}

F_1

salt	enc	Name	YoB
S_{11}	Bd6!l3	Alice	1980
S_{12}	Oij3X.	Bob	1980
S_{13}	9kEf6?	Carol	1970
S_{14}	ker5/2	David	1970
S_{15}	C:mE91	Eva	1970
S_{16}	4lDwqz	Frank	1960
S_{17}	me3,op	Gary	1960
S_{18}	zWf4g>	Hilary	1960

F_2

salt	enc	Job
S_{21}	8de6TO	Clerk
S_{22}	X'mlE3	Doctor
S_{23}	wq.vy0	Nurse
S_{24}	nh=l3a	Lawyer
S_{25}	hh%kj)	Doctor
S_{26}	;vf5eS	Doctor
S_{27}	e4+YUp	Teacher
S_{28}	pgt6eC	Nurse

F_3

salt	enc	Disease
S_{31}	ew3)V!	Asthma
S_{32}	LkEd69	Asthma
S_{33}	w8vd66	Asthma
S_{34}	1"qPdd	Bronchitis
S_{35}	(mn2eW	Bronchitis
S_{36}	wD}x1X	Gastritis
S_{37}	0opAuEl	Gastritis
S_{38}	Sw@Fez	Diabetes

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**
Q :=SELECT SSN, Name FROM PATIENTS WHERE (Disease=‘Gastritis’ OR Disease=‘Asthma’) AND Job=‘Doctor’	Q^3 :=SELECT salt, enc FROM F_3 WHERE (Disease=‘Gastritis’ OR Disease=‘Asthma’) Q' := SELECT SSN, Name FROM *Decrypt*(Q^3, *Key*) WHERE Job=‘Doctor’

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 [CDFJPS-07]
  - affinity among attributes [CDFJPS-10]
  - query workload [CDFJPS-09a]

- 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 $\mathscr{F}$

- 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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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	A. Hellman	81/01/03	94142	hypertension	M. White
987-65-4321	B. Dooley	53/10/07	94141	obesity	D. Warren
246-89-1357	C. McKinley	52/02/12	94139	hypertension	M. White
135-79-2468	D. Ripley	81/01/03	94139	obesity	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

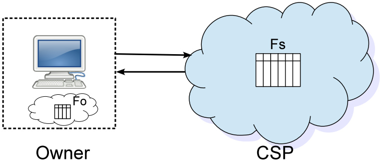
Fragmentation

Keep a few

Basic idea (hybrid scenarios):

– 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 [CDFJPS-09b, CDFJPS-11]



- $F_o \cup F_s = R$

Keep a few – 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

Keep a few – Example

PATIENTS

	SSN	**Name**	**YoB**	**Job**	**Disease**
t_1	123456789	Alice	1980	Clerk	Asthma
t_2	234567891	Bob	1980	Doctor	Asthma
t_3	345678912	Carol	1970	Nurse	Asthma
t_4	456789123	David	1970	Lawyer	Bronchitis
t_5	567891234	Eva	1970	Doctor	Bronchitis
t_6	678912345	Frank	1960	Doctor	Gastritis
t_7	789123456	Gary	1960	Teacher	Gastritis
t_8	891234567	Hilary	1960	Nurse	Diabetes

c_0 = {SSN}
c_1 = {Name, Disease}
c_2 = {Name, Job}
c_3 = {Job, Disease}

F_o

tid	**SSN**	**Job**	**Disease**
1	123456789	Clerk	Asthma
2	234567891	Doctor	Asthma
3	345678912	Nurse	Asthma
4	456789123	Lawyer	Bronchitis
5	567891234	Doctor	Bronchitis
6	678912345	Doctor	Gastritis
7	789123456	Teacher	Gastritis
8	891234567	Nurse	Diabetes

F_s

tid	**Name**	**YoB**
1	Alice	1980
2	Bob	1980
3	Carol	1970
4	David	1970
5	Eva	1970
6	Frank	1960
7	Gary	1960
8	Hilary	1960

Query evaluation

- Queries are formulated on R, therefore 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,Job,Disease}, F_s={Name,YoB}

- $q =$ SELECT SSN, YoB
  FROM Patients
  WHERE (Disease="Bronchitis")
  AND (YoB="1970")
  AND (Name=Job)

- The conditions in the WHERE clause are split as follows
  - $C_o = \{$Disease = "Bronchitis"$\}$
  - $C_s = \{$YoB = "1970"$\}$
  - $C_{so} = \{$Name = Job$\}$

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, YoB
FROM Patients
WHERE (Disease = “Bronchitis”)
AND (YoB = “1970”)
AND (Name = Job)

C_o={Disease = “Bronchitis”}
C_s={YoB = “1970”}
C_{so}={Name = Job}

q_s = SELECT tid,Name,YoB
FROM F_s
WHERE YoB = “1970”

q_{so} = SELECT SSN, YoB
FROM F_o JOIN r_s
ON F_o.tid=r_s.tid
WHERE (Disease = “Bronchitis”) AND (Name = Job)

Client-server strategy – Example

q = SELECT SSN, YoB
FROM Patients
WHERE (Disease = "Bronchitis")
AND (YoB = "1970")
AND (Name = Job)

C_o={Disease = "Bronchitis"}
C_s={YoB = "1970"}
C_{so}={Name $=$ Job}

q_o = SELECT tid
FROM F_o
WHERE Disease = "Bronchitis"

q_s = SELECT tid,Name,YoB
FROM F_s JOIN r_o ON F_s.tid=r_o.tid
WHERE YoB = "1970"

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

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 the 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
  - $w_c(\mathcal{F})=\sum_{cnd \in Cond(\mathcal{Q})} freq(cnd)$ *s.t.* $cnd \cap F_o \neq \emptyset$, where $Cond(\mathcal{Q})$ denotes the set of all conditions of queries in $\mathcal{Q}$, and $freq(cnd)$ is the overall frequency of *cnd*

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

Weighted Minimum Target Hitting Set Problem (WMTHSP). Given a finite set A, a set C of subsets of A, a set $\mathscr{T}$ (target) of subsets of A, and a weight function $w: \mathscr{T} \rightarrow \mathbb{R}^+$, 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 $\sum_{t \in \mathscr{T}, t \cap S' \neq \emptyset} w(t) < \sum_{t \in \mathscr{T}, t \cap S \neq \emptyset} w(t)$

Modeling of the minimization problems – 3

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

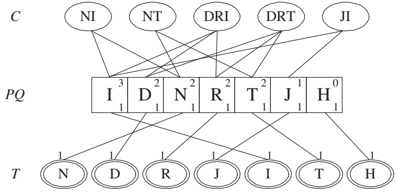
Confidentiality constraints

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}

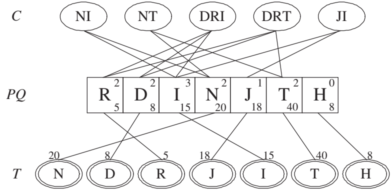
A	*size*(*A*)
SSN	9
Name	20
DoB	8
Race	5
Job	18
Illness	15
Treatment	40
HDate	8

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⟩

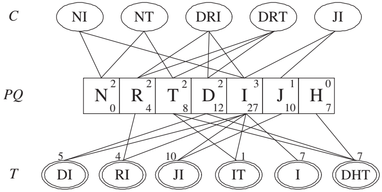
Heuristic algorithm – Example of initialization (2)



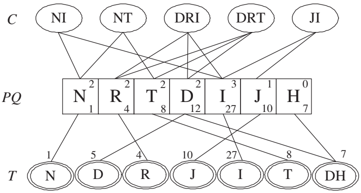
Min-Attr



Min-Size



Min-Query



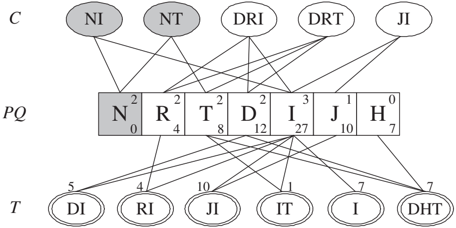
Min-Cond

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 $\mathscr{H}$

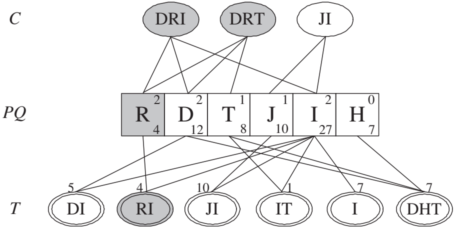
Heuristic algorithm – Example of Min-Query

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



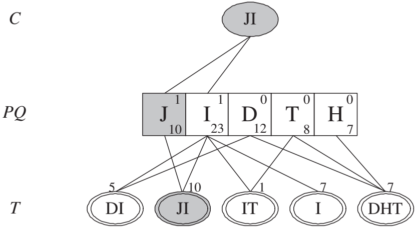
Heuristic algorithm – Example of Min-Query

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



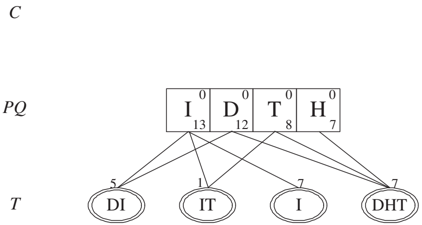
Heuristic algorithm – Example of Min-Query

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



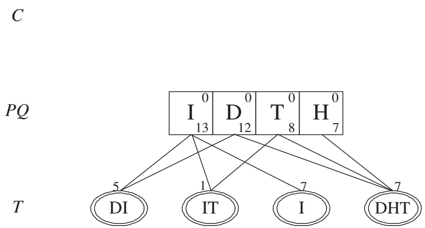
Heuristic algorithm – Example of Min-Query

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



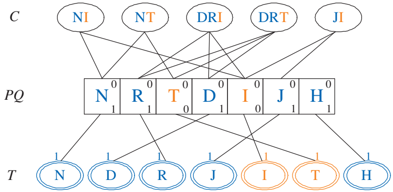
Heuristic algorithm – Example of Min-Query

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

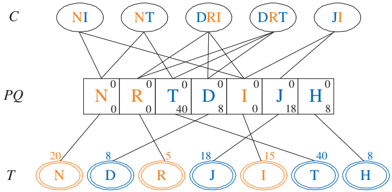


F_o={SSN,Name,Race,Job} F_s={Illness,DoB,Treatment,HDate}

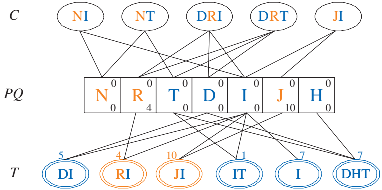
Example of solutions computed by the heuristic algorithm



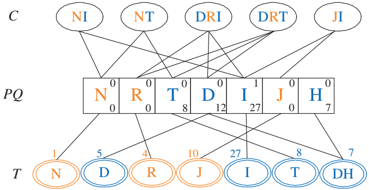
Min-Attr (2)



Min-Size (40)



Min-Query (14)



Min-Cond (15)

Fragmentation and inference

- Fragmentation assumes attributes to be independent
- In presence of data dependencies:
  - sensitive attributes/associations may be indirectly exposed
  - fragments may be indirectly linkable

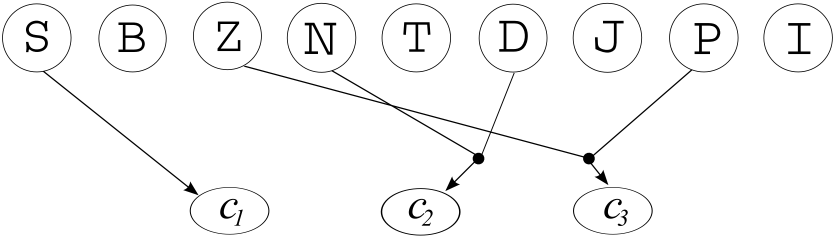
Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)

S B Z N T D J P I

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)

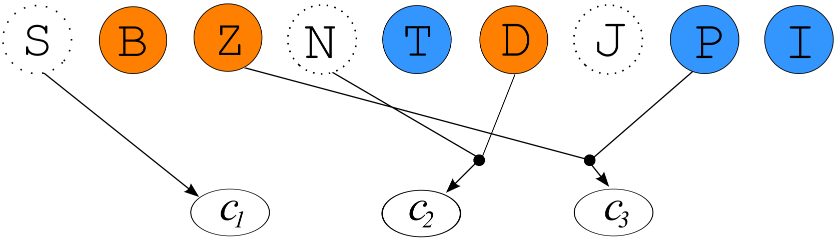


Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)

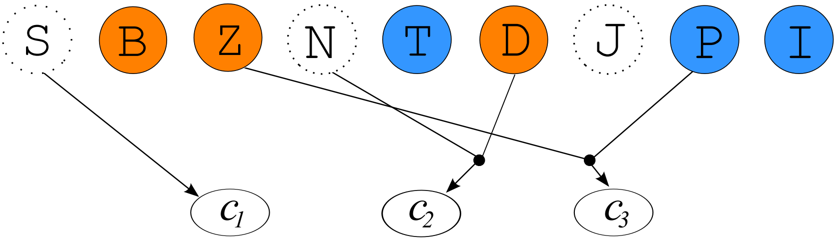


Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

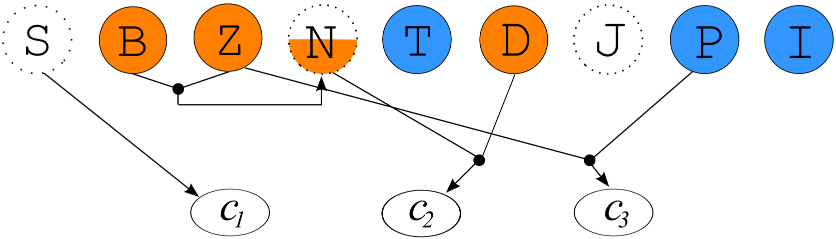
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

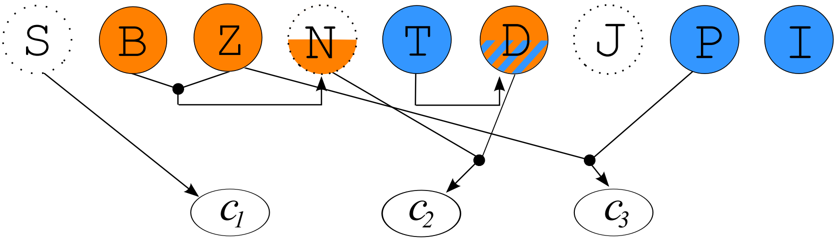
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

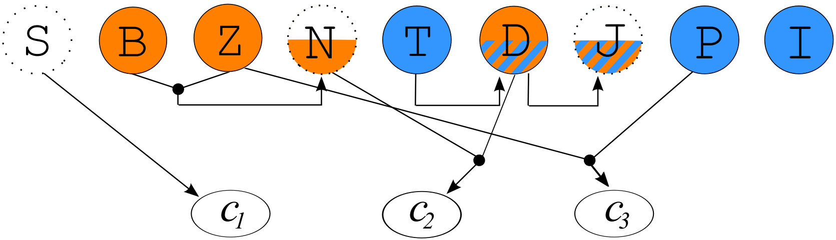
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

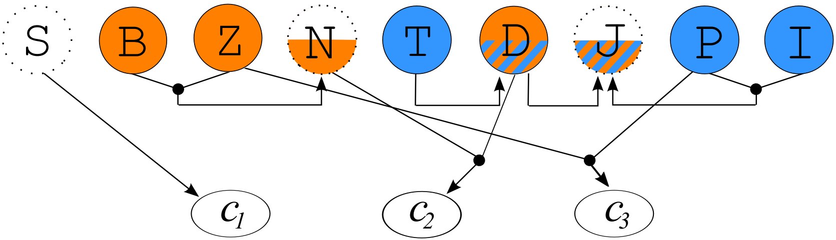
c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP} $\rightsquigarrow$ Name
d_2 = {Treatment} $\rightsquigarrow$ Disease
d_3 = {Disease} $\rightsquigarrow$ Job
d_4 = {Insurance, Premium} $\rightsquigarrow$ Job

Fragmentation and inference – Example

R(SSN, Birth, ZIP, Name, Treatment, Disease, Job, Premium, Insurance)



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

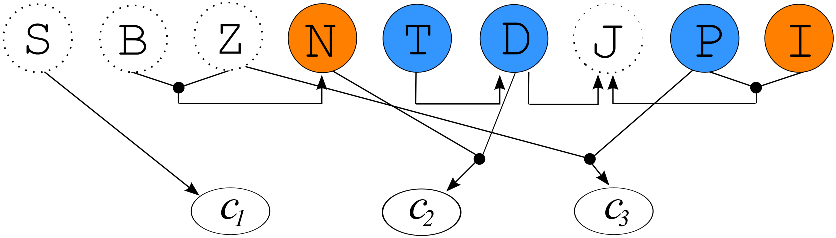
Dependencies

d_1 = {Birth, ZIP}$\rightsquigarrow$ Name
d_2 = {Treatment}$\rightsquigarrow$ Disease
d_3 = {Disease}$\rightsquigarrow$ Job
d_4 = {Insurance, Premium}$\rightsquigarrow$ Job

Fragmenting with data dependencies

Take into account data dependencies in fragmentation

- Fragments should not contain sensitive attributes/associations neither directly nor indirectly



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

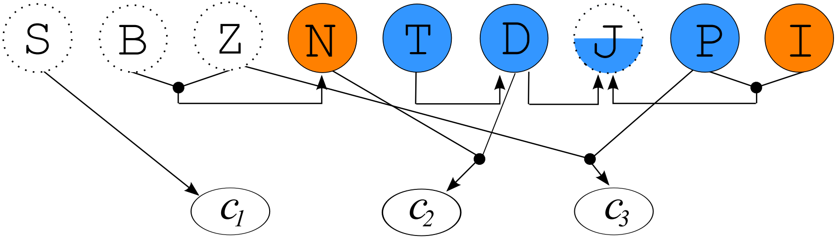
Dependencies

d_1 = {Birth, ZIP}$\rightsquigarrow$ Name
d_2 = {Treatment}$\rightsquigarrow$ Disease
d_3 = {Disease}$\rightsquigarrow$ Job
d_4 = {Insurance, Premium}$\rightsquigarrow$ Job

Fragmenting with data dependencies

Take into account data dependencies in fragmentation

- Fragments should not contain sensitive attributes/associations neither directly nor indirectly



Constraints

c_1 = {SSN}
c_2 = {Name, Disease}
c_3 = {ZIP, Premium}

Dependencies

d_1 = {Birth, ZIP}$\rightsquigarrow$ Name
d_2 = {Treatment}$\rightsquigarrow$ Disease
d_3 = {Disease}$\rightsquigarrow$ Job
d_4 = {Insurance, Premium}$\rightsquigarrow$ Job

Publishing Obfuscated Associations

Motivation

- Sensitive associations among data may need to be protected, while allowing execution of certain queries
  - e.g., the set of products available in a pharmacy and the set of customers may be of public knowledge; allow retrieving the average number of products purchased by customers while protecting the association between a particular customer and a particular product

- Possible solutions:
  - [CSYZ-08] exploits a graphical representation of sensitive associations and masks the mapping from entities to nodes of the graph while preserving the graph structure
  - [DFJPS-10a] exploits fragmentation for enforcing confidentiality constraints and visibility requirements and publishes a sanitized form of associations

Anonymizing Bipartite Graph

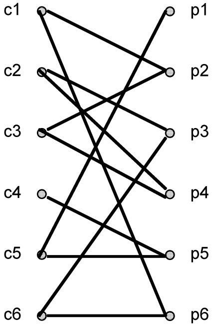
G. Cormode, D. Srivastava, T. YU, Q. Zhang, “Anonymizing Bipartite Graph Data Using Safe Groupings,” in *Proc. of VLDB*, Auckland, New Zealand, August 2008.

Private associations – Example [CSYZ-08]

Customer	State
c1	NJ
c2	NC
c3	CA
c4	NJ
c5	NC
c6	CA

Product	Avail
p1	Rx
p2	OTC
p3	OTC
p4	OTC
p5	Rx
p6	OTC

Customer	Product
c1	p2
c1	p6
c2	p3
c2	p4
c3	p2
c3	p4
c4	p5
c5	p1
c5	p5
c6	p3
c6	p6



Problem statement

Publish anonymized and useful version of bipartite graph in such a way that:

- a broad class of queries can be answered accurately
  - Type 0 - Graph structure only. E.g., what is the average number of products purchased by customers?
  - Type 1 - Attribute predicate on one side only. E.g., what is the average number of products purchased by NJ customers?
  - Type 2 - Attribute predicate on both side. E.g., what is the average number of OTC products purchased by NJ customers?
- privacy of the specific associations is preserved

(k,l) grouping

Basic idea: preserve the graph structure but permute mapping from entities to nodes

(k,l) grouping of bipartite graph $G = (V, W, E)$

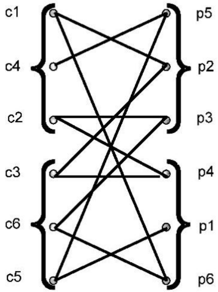
- Partition V (W, resp.) into non-intersecting subsets of size $\geq$ k (l, resp.)
- Publish edges E' that are isomorphic to E, where mapping from E to E' is anonymized based on partitions of V and W

(3,3) grouping – Example (1)

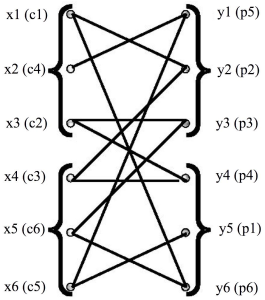
Customer	State
c1	NJ
c2	NC
c3	CA
c4	NJ
c5	NC
c6	CA

Product	Avail
p1	Rx
p2	OTC
p3	OTC
p4	OTC
p5	Rx
p6	OTC

Customer	Product
c1	p2
c1	p6
c2	p3
c2	p4
c3	p2
c3	p4
c4	p5
c5	p1
c5	p5
c6	p3
c6	p6



(3,3) grouping – Example (2)



x1	y2
x1	y6
x2	y1
x3	y3
x3	y4
x4	y2
x4	y4
x5	y3
x5	y6
x6	y1
x6	y5

E'

Customer	Group
c1	CG1
c2	CG1
c3	CG2
c4	CG1
c5	CG2
c6	CG2

H_V

Product	Group
p1	PG2
p2	PG1
p3	PG1
p4	PG2
p5	PG1
p6	PG2

H_W

X-node	Group
x1	CG1
x2	CG1
x3	CG1
x4	CG2
x5	CG2
x6	CG2

R_V

Y-node	Group
y1	PG1
y2	PG1
y3	PG1
y4	PG2
y5	PG2
y6	PG2

R_W

Safe groupings

- There are different ways for creating a (k,l) grouping but not all the resulting groupings offer the same level of privacy (e.g., local clique)
  $\Longrightarrow$ safe (k,l) groupings: nodes in the same group of V are not connected to a same node in W
- The computation of a safe grouping can be hard even for small values of k and l
  - the computation of a safe, strict (3,3)-grouping is NP-hard (reduction from partitioning a graph into triangles)
- The authors propose a greedy algorithm that iteratively adds a node to a group with fewer than k nodes, if it is safe (it creates a new group if such insertion is not possible)
- The algorithm works when bipartite graph is sparse enough

Fragments and Loose Associations

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Fragments and Loose Associations: Respecting Privacy in Data Publishing,” in *Proc. of the VLDB Endowment*, vol. 3, no. 1, September 2010.

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

- To increase utility of published information fragments could be coupled with some associations in sanitized form
  $\Longrightarrow$ loose associations: associations among groups of values (in contrast to specific values)

Confidentiality constraints

As already discussed....

- Sets of attributes such that the (joint) visibility of values of the attributes in the sets should be protected

- They permit to express different requirements
  - sensitive attributes: the values of some attributes are considered sensitive and should not be visible
  - sensitive associations: the associations among values of given attributes are sensitive and should not be visible

Confidentiality constraints – Example

SSN	Patient	Birth	City	Illness	Doctor
123-45-6789	Page	56/12/9	Rome	diabetes	David
987-65-4321	Patrick	53/3/19	Paris	gastritis	Daisy
963-85-2741	Patty	58/5/18	Oslo	flu	Damian
147-85-2369	Paul	53/12/9	Oslo	asthma	Daniel
782-90-5280	Pearl	56/12/9	Rome	gastritis	Dorothy
816-52-7272	Philip	57/6/25	Paris	obesity	Drew
872-62-5178	Phoebe	53/12/1	NY	measles	Dennis
712-81-7618	Piers	60/7/25	Rome	diabetes	Daisy

- SSN is sensitive
  - {SSN}
- Illness and Doctor are private of an individual and cannot be stored in association with the name of the patient
  - {Patient, Illness}, {Patient, Doctor}
- {Birth,City} can work as quasi-identifier
  - {Birth, City, Illness}, {Birth, City, Doctor}

Visibility requirements

- Monotonic Boolean formulas over attributes, representing views over data (negations are captured by confidentiality constraints)
- They permit to express different requirements
  - visible attributes: some attributes should be visible
  - visible associations: the association among values of given attributes should be visible
  - alternative views: at least one of the specified views should be visible

Visibility requirements – Example

SSN	Patient	Birth	City	Illness	Doctor
123-45-6789	Page	56/12/9	Rome	diabetes	David
987-65-4321	Patrick	53/3/19	Paris	gastritis	Daisy
963-85-2741	Patty	58/5/18	Oslo	flu	Damian
147-85-2369	Paul	53/12/9	Oslo	asthma	Daniel
782-90-5280	Pearl	56/12/9	Rome	gastritis	Dorothy
816-52-7272	Philip	57/6/25	Paris	obesity	Drew
872-62-5178	Phoebe	53/12/1	NY	measles	Dennis
712-81-7618	Piers	60/7/25	Rome	diabetes	Daisy

- Either names of Patients or their Cities should be released
  - Patient $\vee$ City
- Either Birth dates and Cities of patients in association should be released or the SSN of patients should be released
  - (Birth $\wedge$ City)$\vee$ SSN
- Illnesses and Doctors, as well as their association, should be released
  - Illness $\wedge$ Doctor

Fragmentation

Fragmentation can be applied to satisfy both confidentiality constraints and visibility requirements

- Publish/release to external parties only fragments that
  - do not include sensitive attributes and sensitive associations
  - include the requested attributes and/or associations (all the requirements should be satisfied, not necessarily by a single fragment)

Fragmentation – Example

SSN	Patient	Birth	City	Illness	Doctor
123-45-6789	Page	56/12/9	Rome	diabetes	David
987-65-4321	Patrick	53/3/19	Paris	gastritis	Daisy
963-85-2741	Patty	58/5/18	Oslo	flu	Damian
147-85-2369	Paul	53/12/9	Oslo	asthma	Daniel
782-90-5280	Pearl	56/12/9	Rome	gastritis	Dorothy
816-52-7272	Philip	57/6/25	Paris	obesity	Drew
872-62-5178	Phoebe	53/12/1	NY	measles	Dennis
712-81-7618	Piers	60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

v_1=Patient $\vee$ City
v_2=(Birth $\wedge$ City)$\vee$ SSN
v_3=Illness $\wedge$ Doctor

Fragmentation – Example

SSN	Patient	Birth	City	Illness	Doctor
123-45-6789	Page	56/12/9	Rome	diabetes	David
987-65-4321	Patrick	53/3/19	Paris	gastritis	Daisy
963-85-2741	Patty	58/5/18	Oslo	flu	Damian
147-85-2369	Paul	53/12/9	Oslo	asthma	Daniel
782-90-5280	Pearl	56/12/9	Rome	gastritis	Dorothy
816-52-7272	Philip	57/6/25	Paris	obesity	Drew
872-62-5178	Phoebe	53/12/1	NY	measles	Dennis
712-81-7618	Piers	60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

v_1=Patient $\vee$ City
v_2=(Birth $\wedge$ City)$\vee$ SSN
v_3=Illness $\wedge$ Doctor

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

Correct and minimal fragmentation

- A fragmentation is correct if
  - each confidentiality constraint is satisfied by all fragments
  - each visibility requirement is satisfied by at least a fragment
  - fragments do not have attributes in common (to prevent joins on fragments to retrieve associations)
- A correct fragmentation is minimal if
  - the number of fragments is minimum (i.e., any other correct fragmentation has an equal or greater number of fragments)
- The Min-CF problem of computing a correct and minimal fragmentation is NP-hard

Computing a correct and minimal fragmentation

A SAT solver can efficiently solve the Min-CF problem

- An instance of the Min-CF problem is translated into an instance of the SAT problem
- The inputs to the Min-CF problem are interpreted as boolean formulas
  - visibility requirements are already represented as boolean formulas
  - each confidentiality constraint is represented via a boolean formula as a conjunction of the attributes appearing in the constraint
- Iterate the evaluation of a SAT solver, starting with one fragment and increasing fragments by one at each iteration, until a solution is found (solution is guaranteed to be minimal)

Publishing loose associations – 1

- Fragmentation breaks associations among attributes
- To increase utility of published information, fragments can be coupled with some associations in sanitized form
- A given privacy degree of the association must be guaranteed

$\Longrightarrow$ loose associations: associations among groups of values (in contrast to specific values)

Publishing loose associations – 2

Given two fragments F_l and F_r, a loose association between F_l and F_r

- partitions tuples in the fragments in groups
- provides information on the associations at the group level
- does not permit to exactly reconstruct the original associations among the tuples in the fragments
- provides enriched utility of the published data

Grouping

- Given fragment F_i and its instance f_i, a k-grouping over f_i partitions the tuples in f_i in groups of size greater than or equal to k
$\Longrightarrow$ each tuple t in f_i is associated with a group identifier $G_i(t)$

- A k-grouping is minimal if it maximizes the number of groups (intuitively, it minimizes the size of the groups)

- (k_l,k_r)-grouping denotes the groupings over two instances f_l and f_r of F_l and F_r

- A (k_l,k_r)-grouping is minimal if both the k_l-grouping and the k_r-grouping are minimal

Minimal (2,2)-grouping – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

Minimal (2,2)-grouping – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association

- A (k_l,k_r)-grouping induces a group association A among the groups in f_l and f_r

- A group association A over f_l and f_r is a set of pairs of group identifiers such that:
  - A has the same cardinality as the original relation
  - there is a bijective mapping between the original relation and A that associates each tuple in the original relation with a pair $(G_l(l),G_r(r))$ in A, with $l \in f_l$ and $r \in f_r$

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Group association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City	G
53/3/19	Paris	bc1
53/12/9	Oslo	bc1
56/12/9	Rome	bc2
57/6/25	Paris	bc2
58/5/18	Oslo	bc3
56/12/9	Rome	bc3
53/12/1	NY	bc4
60/7/25	Rome	bc4

G_l	G_r
bc1	id1
bc1	id2
bc2	id1
bc2	id3
bc3	id2
bc3	id4
bc4	id3
bc4	id4

F_r

G	Illness	Doctor
id1	gastritis	Daisy
id1	diabetes	David
id2	asthma	Daniel
id2	flu	Damian
id3	obesity	Drew
id3	measles	Dennis
id4	gastritis	Dorothy
id4	diabetes	Daisy

Group association protection

- Duplicates in fragments are maintained (all fragments have the same cardinality as the original relation)
  - fragments may contain tuples that are equal
- Even tuples that are different may have the same values for attributes involved in a confidentiality constraint
- The looseness protection offered by grouping can be compromised

  $\Longrightarrow$ need to control occurrences of the same values

Alikeness

- Two tuples l_i, l_j in f_l (r_i, r_j in f_r) are alike w.r.t. a constraint c, denoted $l_i \simeq_c l_j$ ($r_i \simeq_c r_j$), if
  - $c \subseteq (F_l \cup F_r)$ (c is covered by F_l and F_r)
  - $l_i[c \cap F_l]$ = $l_j[c \cap F_l]$ ($r_i[c \cap F_r]$ = $r_j[c \cap F_r]$)
- Two tuples l_i, l_j in f_l (r_i, r_j in f_r) are alike $l_i \simeq l_j$ ($r_i \simeq r_j$) if they are alike w.r.t. at least a constraint $c \subseteq (F_l \cup F_r)$
- $\simeq_c$ is transitive for any constraint c
- $\simeq$ is not transitive if there are at least two constraints covered by F_l and F_r

Alikeness – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

Alikeness – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

$\simeq_{c_4}$

Alikeness – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

$\simeq_{c_4}$ $\simeq_{c_3}$

Alikeness – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
56/12/9	Rome
53/3/19	Paris
58/5/18	Oslo
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
diabetes	David
gastritis	Daisy
flu	Damian
asthma	Daniel
gastritis	Dorothy
obesity	Drew
measles	Dennis
diabetes	Daisy

$\not\simeq$

k-loose association

- A group association is k-loose if every tuple in the group association A indistinguishably corresponds to at least k distinct associations among tuples in the fragments

- A k-loose association is also k'-loose for any $k' \leq k$

- A (k_l, k_r)-grouping induces a minimal group association A if
  - A is k-loose
  - $\nexists$ a (k'_l, k'_r)-grouping inducing a k-loose association s.t. $k'_l \cdot k'_r < k_l \cdot k_r$

4-loose association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

4-loose association – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City	G
53/3/19	Paris	bc1
53/12/9	Oslo	bc1
56/12/9	Rome	bc2
57/6/25	Paris	bc2
58/5/18	Oslo	bc3
56/12/9	Rome	bc3
53/12/1	NY	bc4
60/7/25	Rome	bc4

G_l	G_r
bc1	id1
bc1	id2
bc2	id1
bc2	id3
bc3	id2
bc3	id4
bc4	id3
bc4	id4

F_r

G	Illness	Doctor
id1	gastritis	Daisy
id1	diabetes	David
id2	asthma	Daniel
id2	flu	Damian
id3	obesity	Drew
id3	measles	Dennis
id4	gastritis	Dorothy
id4	diabetes	Daisy

Heterogeneity properties

- There is a correspondence between k_l, k_r of the groupings and the degree of k-looseness of the induced group association
  - a (k_l,k_r)-grouping cannot induce a k-loose association for a $k > k_l \cdot k_r$
  - the value $k \leq k_l \cdot k_r$ depends on how groups are defined
- If a (k_l,k_r)-grouping satisfies given heterogeneity properties, the induced group association is k-loose with k=$k_l \cdot k_r$
  - group heterogeneity
  - association heterogeneity
  - deep heterogeneity

Group heterogeneity

No group can contain tuples that are alike with respect to the constraints covered by F_l and F_r

- it ensures diversity of tuples within groups

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor	
gastritis	Daisy	NO
gastritis	Dorothy	
asthma	Daniel	
flu	Damian	
obesity	Drew	
measles	Dennis	
diabetes	David	NO
diabetes	Daisy	

Group heterogeneity

No group can contain tuples that are alike with respect to the constraints covered by F_l and F_r

- it ensures diversity of tuples within groups

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Association heterogeneity

No group can be associated twice with another group (the group association cannot contain any duplicate)

- it ensures that for each real tuple in the original relation there are at least $k_l \cdot k_r$ pairs in the group association that may correspond to it

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

NO

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Association heterogeneity

No group can be associated twice with another group (the group association cannot contain any duplicate)

- it ensures that for each real tuple in the original relation there are at least $k_l \cdot k_r$ pairs in the group association that may correspond to it

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Deep heterogeneity

No group can be associated with two groups that contain alike tuples

- it ensures that all $k_l \cdot k_r$ pairs in the group association to which each tuple could correspond to contain diverse values for attributes involved in constraints

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
56/12/9	Rome
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
53/12/9	Oslo
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
gastritis	Dorothy
flu	Damian
obesity	Drew
measles	Dennis
asthma	Daniel
diabetes	Daisy

NO

Deep heterogeneity

No group can be associated with two groups that contain alike tuples

- it ensures that all $k_l \cdot k_r$ pairs in the group association to which each tuple could correspond to contain diverse values for attributes involved in constraints

c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Flat grouping vs sparse grouping

- A (k_l,k_r)-grouping is
  - flat if either k_l or k_r is equal to 1
  - sparse if both k_l and k_r are different from 1
- Flat grouping resembles k-anonymity and captures at the same time the ℓ-diversity property, but it works on associations and attributes' values are not generalized
- Sparse grouping guarantees larger applicability than flat grouping, with the same level of protection
  (there may exist a sparse grouping providing k-looseness but not a flat grouping)

Flat grouping – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
58/5/18	Oslo
53/12/1	NY
56/12/9	Rome
57/6/25	Paris
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
asthma	Daniel
diabetes	David
flu	Damian
measles	Dennis
gastritis	Dorothy
obesity	Drew
diabetes	Daisy

Sparse grouping – Example

Birth	City	Illness	Doctor
56/12/9	Rome	diabetes	David
53/3/19	Paris	gastritis	Daisy
58/5/18	Oslo	flu	Damian
53/12/9	Oslo	asthma	Daniel
56/12/9	Rome	gastritis	Dorothy
57/6/25	Paris	obesity	Drew
53/12/1	NY	measles	Dennis
60/7/25	Rome	diabetes	Daisy

c_0={SSN}
c_1={Patient,Illness}
c_2={Patient,Doctor}
c_3={Birth,City,Illness}
c_4={Birth,City,Doctor}

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Privacy vs utility

- The publication of loose associations increases data utility
  - it makes it possible to evaluate queries more precisely than if only the fragments were published

- Increased utility corresponds to a greater exposure of information (lower privacy degree)

Association exposure

- The exposure of a sensitive association $\langle l[c \cap F_l], r[c \cap F_r] \rangle$, with c a constraint covered by F_l, F_r, can be expressed as the probability of the association to hold in the original relation (given the published information)

- The increased exposure due to the publication of loose associations can be measured as the difference between
  - the probability $P^A(l[c \cap F_l], r[c \cap F_r])$ that the sensitive association $\langle l[c \cap F_l], r[c \cap F_r] \rangle$ appears in the original relation, given f_l, f_r, and A
  - the probability $P(l[c \cap F_l], r[c \cap F_r])$ that the sensitive association $\langle l[c \cap F_l], r[c \cap F_r] \rangle$ appears in the original relation, given f_l and f_r

Exposure without loose association – 1

- Given $l \in f_l$ and $r \in f_r$ the probability $P(l,r)$ that tuple $\langle l,r \rangle$ belongs to the original relation is $1/|f_l| = 1/|f_r|$

Exposure without loose association – 1

- Given $l \in f_l$ and $r \in f_r$ the probability $P(l,r)$ that tuple $\langle l,r \rangle$ belongs to the original relation is $1/|f_l| = 1/|f_r|$

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
		Daisy	David	Daniel	Damian	Drew	Dennis	Dorothy	Daisy
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

Exposure without loose association – 2

- Exposure ($P(l[c \cap F_l], r[c \cap F_r])$) depends on the presence of alike tuples
- Let l_i,l_j be two tuples in f_l s.t. $l_i \simeq_c l_j$, $P(l_i[c \cap F_l], r[c \cap F_r])$ is the composition of the probability that
  - l_i is associated with r
  - l_j is associated with r

$$P(l_i,r) + P(l_j,r) - (P(l_i,r) \cdot P(l_j,r))$$

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
		Daisy	David	Daniel	Damian	Drew	Dennis	Dorothy	Daisy
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
		Daisy	David	Daniel	Damian	Drew	Dennis	Dorothy	Daisy
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

$\simeq_{c_3}$

c_3={Birth,City,Illness}

$$P(\text{56/12/9,Rome,gastritis}) = P(\text{56/12/9,Rome,diabetes}) = \ldots = P(\text{56/12/9,Rome,diabetes}) = \frac{1}{8} + \frac{1}{8} - \left(\frac{1}{8} \cdot \frac{1}{8}\right)$$

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	15/64	15/64	15/64	15/64	15/64	15/64	15/64	15/64
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

P(56/12/9,Rome,gastritis) = P(56/12/9,Rome,diabetes) = … = P(56/12/9,Rome,diabetes) =

$$\frac{1}{8} + \frac{1}{8} - \left(\frac{1}{8} \cdot \frac{1}{8}\right) = \frac{15}{64}$$

Exposure without loose association – Example

$\simeq_{c_3}$ (over columns gastritis … measles)

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	15/64	15/64	15/64	15/64	15/64	15/64	15/64	15/64
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

$$P(\text{53/3/19,Paris,gastritis}) = P(\text{53/12/9,Oslo,gastritis}) = \ldots = P(\text{60/7/25,Rome,gastritis}) = \frac{1}{8} + \frac{1}{8} - \left(\frac{1}{8} \cdot \frac{1}{8}\right)$$

$$P(\text{56/12/9,Rome,gastritis}) = \frac{15}{64} + \frac{15}{64} - \left(\frac{15}{64} \cdot \frac{15}{64}\right)$$

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	diabetes
53/3/19	Paris	15/64	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	15/64	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1695/4096	15/64	15/64	15/64	15/64	15/64	15/64
57/6/25	Paris	15/64	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	15/64	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	15/64	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	15/64	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

P(53/3/19,Paris,gastritis) = P(53/12/9,Oslo,gastritis) = … = P(60/7/25,Rome,gastritis) =

$$\frac{1}{8}+\frac{1}{8}-\left(\frac{1}{8}\cdot\frac{1}{8}\right)=\frac{15}{64}$$

P(56/12/9,Rome,gastritis) = $\frac{15}{64}+\frac{15}{64}-\left(\frac{15}{64}\cdot\frac{15}{64}\right)=\frac{1695}{4096}$

Exposure without loose association – Example

$\simeq_{c_3}$ (over the columns diabetes through measles)

		gastritis	diabetes	asthma	flu	obesity	measles	diabetes
53/3/19	Paris	15/64	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	15/64	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1695/4096	15/64	15/64	15/64	15/64	15/64	15/64
57/6/25	Paris	15/64	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	15/64	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	15/64	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	15/64	1/8	1/8	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

$$P(\text{53/3/19,Paris,diabetes}) = P(\text{53/12/9,Oslo,diabetes}) = \ldots = P(\text{60/7/25,Rome,diabetes}) = \frac{1}{8}+\frac{1}{8}-\left(\frac{1}{8}\cdot\frac{1}{8}\right)$$

$$P(\text{56/12/9,Rome,diabetes}) = \frac{15}{64}+\frac{15}{64}-\left(\frac{15}{64}\cdot\frac{15}{64}\right)$$

Exposure without loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	15/64	15/64	1/8	1/8	1/8	1/8
53/12/9	Oslo	15/64	15/64	1/8	1/8	1/8	1/8
56/12/9	Rome	1695/4096	1695/4096	15/64	15/64	15/64	15/64
57/6/25	Paris	15/64	15/64	1/8	1/8	1/8	1/8
58/5/18	Oslo	15/64	15/64	1/8	1/8	1/8	1/8
53/12/1	NY	15/64	15/64	1/8	1/8	1/8	1/8
60/7/25	Rome	15/64	15/64	1/8	1/8	1/8	1/8

c_3={Birth,City,Illness}

$$P(\text{53/3/19,Paris,diabetes}) = P(\text{53/12/9,Oslo,diabetes}) = \ldots = P(\text{60/7/25,Rome,diabetes}) = \frac{1}{8} + \frac{1}{8} - \left(\frac{1}{8} \cdot \frac{1}{8}\right) = \frac{15}{64}$$

$$P(\text{56/12/9,Rome,diabetes}) = \frac{15}{64} + \frac{15}{64} - \left(\frac{15}{64} \cdot \frac{15}{64}\right) = \frac{1695}{4096}$$

Exposure with loose association

- Given $l \in f_l$ and $r \in f_r$ the probability $P^A(l,r)$ that tuple $\langle l,r \rangle$ belongs to the original relation is at most $1/k$
- $P^A(l[c \cap F_l], r[c \cap F_r])$ is evaluated considering the alike $\simeq_c$ relationship
  - let l_i, l_j in f_l s.t. $l_i \simeq_c l_j$, $P^A(l_i[c \cap F_l], r[c \cap F_r])$ is the composition of the probability that
    - l_i is associated with r
    - l_j is associated with r

$$P^A(l_i,r) + P^A(l_j,r) - (P^A(l_i,r) \cdot P^A(l_j,r))$$

Exposure with loose association – Example

		gastritis Daisy	diabetes David	asthma Daniel	flu Damian	obesity Drew	measles Dennis	gastritis Dorothy	diabetes Daisy
53/3/19	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/9	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
57/6/25	Paris	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
58/5/18	Oslo	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
56/12/9	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
53/12/1	NY	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8
60/7/25	Rome	1/8	1/8	1/8	1/8	1/8	1/8	1/8	1/8

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
		Daisy	David	Daniel	Damian	Drew	Dennis	Dorothy	Daisy
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–	–
56/12/9	Rome	1/4	1/4	–	–	1/4	1/4	–	–
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–	–
58/5/18	Oslo	–	–	1/4	1/4	–	–	1/4	1/4
56/12/9	Rome	–	–	1/4	1/4	–	–	1/4	1/4
53/12/1	NY	–	–	–	–	1/4	1/4	1/4	1/4
60/7/25	Rome	–	–	–	–	1/4	1/4	1/4	1/4

F_l

Birth	City
53/3/19	Paris
53/12/9	Oslo
56/12/9	Rome
57/6/25	Paris
58/5/18	Oslo
56/12/9	Rome
53/12/1	NY
60/7/25	Rome

F_r

Illness	Doctor
gastritis	Daisy
diabetes	David
asthma	Daniel
flu	Damian
obesity	Drew
measles	Dennis
gastritis	Dorothy
diabetes	Daisy

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
		Daisy	David	Daniel	Damian	Drew	Dennis	Dorothy	Daisy
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–	–
56/12/9	Rome	1/4	1/4	–	–	1/4	1/4	–	–
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–	–
58/5/18	Oslo	–	–	1/4	1/4	–	–	1/4	1/4
56/12/9	Rome	–	–	1/4	1/4	–	–	1/4	1/4
53/12/1	NY	–	–	–	–	1/4	1/4	1/4	1/4
60/7/25	Rome	–	–	–	–	1/4	1/4	1/4	1/4

c_3={Birth,City,Illness}

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–	–
56/12/9	Rome	1/4	1/4	–	–	1/4	1/4	–	–
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–	–
58/5/18	Oslo	–	–	1/4	1/4	–	–	1/4	1/4
56/12/9	Rome	–	–	1/4	1/4	–	–	1/4	1/4
53/12/1	NY	–	–	–	–	1/4	1/4	1/4	1/4
60/7/25	Rome	–	–	–	–	1/4	1/4	1/4	1/4

$\simeq_{c_3}$

c_3={Birth,City,Illness}

$$P(\text{56/12/9,Rome,gastritis}) = P(\text{56/12/9,Rome,diabetes}) = \ldots = P(\text{56/12/9,Rome,diabetes}) = \frac{1}{4} + 0 - \left(\frac{1}{4} \cdot 0\right)$$

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–	–
56/12/9	Rome	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–	–
58/5/18	Oslo	–	–	1/4	1/4	–	–	1/4	1/4
53/12/1	NY	–	–	–	–	1/4	1/4	1/4	1/4
60/7/25	Rome	–	–	–	–	1/4	1/4	1/4	1/4

c_3={Birth,City,Illness}

P(56/12/9,Rome,gastritis) = P(56/12/9,Rome,diabetes) = … = P(56/12/9,Rome,diabetes) =

$$\frac{1}{4} + 0 - \left(\frac{1}{4} \cdot 0\right) = \frac{1}{4}$$

Exposure with loose association – Example

$\overbrace{\hspace{20em}}^{\simeq_{c_3}}$

		gastritis	diabetes	asthma	flu	obesity	measles	gastritis	diabetes
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–	–
56/12/9	Rome	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–	–
58/5/18	Oslo	–	–	1/4	1/4	–	–	1/4	1/4
53/12/1	NY	–	–	–	–	1/4	1/4	1/4	1/4
60/7/25	Rome	–	–	–	–	1/4	1/4	1/4	1/4

c_3={Birth,City,Illness}

P(53/3/19,Paris,gastritis) = P(53/12/9,Oslo,gastritis) = … = P(60/7/25,Rome,gastritis) =

$$\frac{1}{4} + 0 - \left(\frac{1}{4} \cdot 0\right)$$

P(56/12/9,Rome,gastritis) = $\frac{1}{4} + \frac{1}{4} - \left(\frac{1}{4} \cdot \frac{1}{4}\right)$

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles	diabetes
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–
56/12/9	Rome	7/16	1/4	1/4	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–
58/5/18	Oslo	1/4	–	1/4	1/4	–	–	1/4
53/12/1	NY	1/4	–	–	–	1/4	1/4	1/4
60/7/25	Rome	1/4	–	–	–	1/4	1/4	1/4

c_3={Birth,City,Illness}

P(53/3/19,Paris,gastritis) = P(53/12/9,Oslo,gastritis) = … = P(60/7/25,Rome,gastritis) =

$$\frac{1}{4} + 0 - \left(\frac{1}{4} \cdot 0\right) = \frac{1}{4}$$

$$P(\text{56/12/9,Rome,gastritis}) = \frac{1}{4} + \frac{1}{4} - \left(\frac{1}{4} \cdot \frac{1}{4}\right) = \frac{7}{16}$$

Exposure with loose association – Example

$\simeq_{c_3}$

		gastritis	diabetes	asthma	flu	obesity	measles	diabetes
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–	–
56/12/9	Rome	7/16	1/4	1/4	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4	–
58/5/18	Oslo	1/4	–	1/4	1/4	–	–	1/4
53/12/1	NY	1/4	–	–	–	1/4	1/4	1/4
60/7/25	Rome	1/4	–	–	–	1/4	1/4	1/4

c_3={Birth,City,Illness}

$$P(\text{53/3/19,Paris,diabetes}) = P(\text{53/12/9,Oslo,diabetes}) = \ldots = P(\text{60/7/25,Rome,diabetes}) = \frac{1}{4} + 0 - \left(\frac{1}{4} \cdot 0\right)$$

$$P(\text{56/12/9,Rome,diabetes}) = \frac{1}{4} + \frac{1}{4} - \left(\frac{1}{4} \cdot \frac{1}{4}\right)$$

Exposure with loose association – Example

		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–
56/12/9	Rome	7/16	7/16	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4
58/5/18	Oslo	1/4	1/4	1/4	1/4	–	–
53/12/1	NY	1/4	1/4	–	–	1/4	1/4
60/7/25	Rome	1/4	1/4	–	–	1/4	1/4

c_3={Birth,City,Illness}

P(53/3/19,Paris,diabetes) = P(53/12/9,Oslo,diabetes) = $\ldots$ = P(60/7/25,Rome,diabetes) =

$$\frac{1}{4}+0-\left(\frac{1}{4}\cdot 0\right)=\frac{1}{4}$$

P(56/12/9,Rome,diabetes) = $\frac{1}{4}+\frac{1}{4}-\left(\frac{1}{4}\cdot\frac{1}{4}\right)=\frac{7}{16}$

Measuring privacy and utility

- Utility: average over the variation of probability $|P^A(l[c \cap F_l], r[c \cap F_r]) - P(l[c \cap F_l], r[c \cap F_r])|$ for each sensitive association $\langle l[c \cap F_l], r[c \cap F_r] \rangle$
  - measured also in terms of the precision in responding to queries
- Privacy: in addition to the k-loose degree, an exposure threshold $\delta_{\max}$ could be specified
  - given a threshold $\delta_{\max}$, A can be published if $\delta_{\max} \geq (P^A(l[c \cap F_l], r[c \cap F_r]) - P(l[c \cap F_l], r[c \cap F_r]))$ for all sensitive associations $\langle l[c \cap F_l], r[c \cap F_r] \rangle$

Measuring utility – Example

P^A

		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	1/4	1/4	1/4	1/4	–	–
53/12/9	Oslo	1/4	1/4	1/4	1/4	–	–
56/12/9	Rome	7/16	7/16	1/4	1/4	1/4	1/4
57/6/25	Paris	1/4	1/4	–	–	1/4	1/4
58/5/18	Oslo	1/4	1/4	1/4	1/4	–	–
53/12/1	NY	1/4	1/4	–	–	1/4	1/4
60/7/25	Rome	1/4	1/4	–	–	1/4	1/4

P

		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	15/64	15/64	1/8	1/8	1/8	1/8
53/12/9	Oslo	15/64	15/64	1/8	1/8	1/8	1/8
56/12/9	Rome	1695/4096	1695/4096	15/64	15/64	15/64	15/64
57/6/25	Paris	15/64	15/64	1/8	1/8	1/8	1/8
58/5/18	Oslo	15/64	15/64	1/8	1/8	1/8	1/8
53/12/1	NY	15/64	15/64	1/8	1/8	1/8	1/8
60/7/25	Rome	15/64	15/64	1/8	1/8	1/8	1/8

$$P^A(l[\text{Birth,City}],\ r[\text{Illness}]) - P(l[\text{Birth,City}],\ r[\text{Illness}])$$

Measuring utility – Example

$P^A(l[\text{Birth,City}],\ r[\text{Illness}]) - P(l[\text{Birth,City}],\ r[\text{Illness}])$

		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	1/64	1/64	1/8	1/8	-1/8	-1/8
53/12/9	Oslo	1/64	1/64	1/8	1/8	-1/8	-1/8
56/12/9	Rome	97/4096	97/4096	1/64	1/64	1/64	1/64
57/6/25	Paris	1/64	1/64	-1/8	-1/8	1/8	1/8
58/5/18	Oslo	1/64	1/64	1/8	1/8	-1/8	-1/8
53/12/1	NY	1/64	1/64	-1/8	-1/8	1/8	1/8
60/7/25	Rome	1/64	1/64	-1/8	-1/8	1/8	1/8

Measuring utility – Example

$P^A(l[\text{Birth,City}],\ r[\text{Illness}]) - P(l[\text{Birth,City}],\ r[\text{Illness}])$

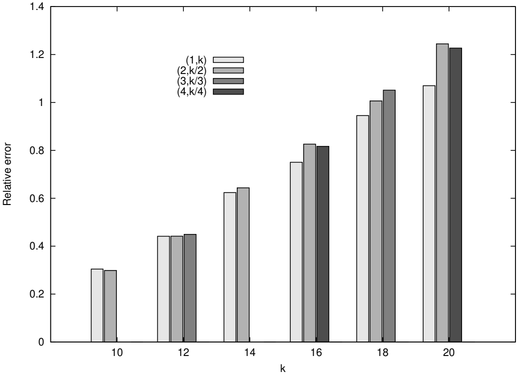
		gastritis	diabetes	asthma	flu	obesity	measles
53/3/19	Paris	1/64	1/64	1/8	1/8	-1/8	-1/8
53/12/9	Oslo	1/64	1/64	1/8	1/8	-1/8	-1/8
56/12/9	Rome	97/4096	97/4096	1/64	1/64	1/64	1/64
57/6/25	Paris	1/64	1/64	-1/8	-1/8	1/8	1/8
58/5/18	Oslo	1/64	1/64	1/8	1/8	-1/8	-1/8
53/12/1	NY	1/64	1/64	-1/8	-1/8	1/8	1/8
60/7/25	Rome	1/64	1/64	-1/8	-1/8	1/8	1/8

$$\text{Utility} = \frac{\sum_{l,r} |P^A(l[\text{Birth,City}], r[\text{Illness}]) - P(l[\text{Birth,City}], r[\text{Illness}])|}{42} = \frac{13506}{172032}$$

Experimental evaluation

- Considered Census data (IPUMS-USA, http://www.ipums.org)
- Evaluated queries of the form
  - SELECT FROM WHERE returning a COUNT aggregation function
  - WHERE condition $\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m} a_i = v_{i_j})$
- Evaluated precision of queries
- Evaluated impact of k, k_l, and k_r on query precision

Experimental evaluation – Results



- Precision in query evaluation progressively decreases as k increases
- The critical parameter in the configuration is the overall privacy degree k, rather than individual values of k_l and k_r

Summary of contributions

- Novel approach to the problem of protecting privacy when publishing data
- Generic setting of the privacy problem that explicitly takes into consideration both privacy needs and visibility requirements
- Definition of loose associations for increasing data utility while preserving a given degree of privacy

Some open issues...

- Schema vs. instance constraints and visibility requirements
- Data dependencies not captured by confidentiality constraints
- External knowledge
- Support for different kinds of queries
- Different metrics to measure privacy and utility

Combining Indexes, Selective Encryption, and Fragmentation

Exposure of confidential information

- Indexes, fragmentation, and selective encryption are all solutions providing the required security and privacy guarantees but...

- ...What happens when such solutions are combined?

Exposure of confidential information

- Indexes, fragmentation, and selective encryption are all solutions providing the required security and privacy guarantees but...
- ...What happens when such solutions are combined?

$\Longrightarrow$ They may open the door to inferences by users

Exposure of confidential information

- Indexes, fragmentation, and selective encryption are all solutions providing the required security and privacy guarantees but...
- ...What happens when such solutions are combined?

$\Longrightarrow$ They may open the door to inferences by users

  - Indexes and selective encryption
  - Indexes and fragmentation

Indexes and Selective Access to Outsourced Data

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Private Data Indexes for Selective Access to Outsourced Data,” in *Proc. of WPES*, Chicago, IL, USA, October 2011.

The inference problem

- The storage server can be honest-but-curious
- The server cannot decrypt the data for executing queries
  $\Longrightarrow$ indexes can be associated with encrypted data to allow the server to execute queries on them
- The data owner may want to provide different data views to different users
  $\Longrightarrow$ selective encryption uses different keys for different portions of the data
- The combination of the two solutions may open the door to inferences by users

Blocking inferences [DFJPS-11]

- Characterize the exposure of confidential information due to indexes and access control enforcement
- Define a index function, depending on plaintext values and access control restrictions, that
  - supports efficient query evaluation
  - protects against inference exposure

Encrypted relation

- Symmetric encryption is applied at the tuple-level
- The encrypted version of relation r over schema $R(\mathtt{A}_1, \ldots, \mathtt{A}_n)$ is a relation r^e over schema $R^e(\underline{\text{tid}}, \text{etuple}, \mathtt{I}_1, \ldots, \mathtt{I}_l)$:
  - tid: numerical attribute acting as primary key
  - etuple: ciphertext resulting from the encryption of a tuple
  - $\mathtt{I}_i$, i=1,...,l: index over attribute $\mathtt{A}_{j_i} \in R$

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Indexing techniques

Remember . . .:

- Direct index (e.g., [CDDJPS-05])
  each plaintext value is mapped to a different index value and viceversa

- Flattened index (e.g., [WL-06])
  each plaintext value is mapped to a set of index values and each index value corresponds to a unique plaintext value

- Bucket/hash-based index (e.g., [CDDJPS-05, HIML-02])
  different plaintext values are mapped to the same index value

User knowledge

Each user knows the:

- index functions used to define indexes in R^e
- plaintext tuples that she is authorized to access
- encrypted relation r^e in its entirety

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

User knowledge

Each user knows the:

- index functions used to define indexes in R^e
- plaintext tuples that she is authorized to access
- encrypted relation r^e in its entirety

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa
  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa
  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
$\boldsymbol{t_2}$	$\boldsymbol{A,B}$
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	002	Rome	**2010**	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(**2010**)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			**2010**	
t_2	002	Rome	**2010**	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι**(2010)**	ι(600)
2	β	ι(Rome)	ι**(2010)**	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
$\boldsymbol{t_3}$	$\boldsymbol{B}$
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	700
t_3	003	Rome	**2011**	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι**(2011)**	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	700
t_3	003	Rome	**2011**	600
t_4			**2011**	
t_5			**2011**	

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι**(2011)**	ι(600)
4	δ	ι(NY)	ι**(2011)**	ι(700)
5	ε	ι(Oslo)	ι**(2011)**	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
$\boldsymbol{t_2}$	$\boldsymbol{A,B}$
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	**700**
t_3	003	Rome	2011	600
t_4			2011	
t_5			2011	

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	**ι(700)**
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
$\boldsymbol{t_2}$	$\boldsymbol{A,B}$
t_3	B
$\boldsymbol{t_4}$	$\boldsymbol{A,C}$
$\boldsymbol{t_5}$	$\boldsymbol{C}$

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	**700**
t_3	003	Rome	2011	600
t_4			2011	**700**
t_5			2011	**700**

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι**(700)**
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι**(700)**
5	ε	ι(Oslo)	ι(2011)	ι**(700)**

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
$\boldsymbol{t_3}$	$\boldsymbol{B}$
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	700
t_3	003	Rome	2011	**600**
t_4			2011	700
t_5			2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι**(600)**
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
$\boldsymbol{t_1}$	$\boldsymbol{A}$
t_2	A,B
$\boldsymbol{t_3}$	$\boldsymbol{B}$
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	**600**
t_2	002	Rome	2010	700
t_3	003	Rome	2011	**600**
t_4			2011	700
t_5			2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι**(600)**
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι**(600)**
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
$\boldsymbol{t_2}$	$\boldsymbol{A,B}$
$\boldsymbol{t_3}$	$\boldsymbol{B}$
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	600
t_2	002	**Rome**	2010	700
t_3	003	**Rome**	2011	600
t_4			2011	700
t_5			2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι**(Rome)**	ι(2010)	ι(700)
3	γ	ι**(Rome)**	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Direct index (1)

- Plaintext values are always represented by the same index value and viceversa

  $\Longrightarrow$ cells having the same plaintext values are exposed

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1		~~Rome~~	2010	600
t_2	002	**Rome**	2010	700
t_3	003	**Rome**	2011	600
t_4		~~Rome~~	2011	700
t_5		~~Rome~~	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι**(NY)**	ι(2010)	ι(600)
2	β	ι**(Rome)**	ι(2010)	ι(700)
3	γ	ι**(Rome)**	ι(2011)	ι(600)
4	δ	ι**(NY)**	ι(2011)	ι(700)
5	ε	ι**(Oslo)**	ι(2011)	ι(700)

Exposure risk – Direct index (2)

- Each user knows index function ι

  $\Longrightarrow$ all index-plaintext value correspondences are exposed to brute-force attacks

  $\Longrightarrow$ the whole outsourced relation is exposed to brute-force attacks

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1		NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4		NY	2011	700
t_5		Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι(NY)	ι(2010)	ι(600)
2	β	ι(Rome)	ι(2010)	ι(700)
3	γ	ι(Rome)	ι(2011)	ι(600)
4	δ	ι(NY)	ι(2011)	ι(700)
5	ε	ι(Oslo)	ι(2011)	ι(700)

Exposure risk – Flattened and bucket/hash-based index

- Flattened index: an index value always represents the same plaintext value and users know the index function
  $\Longrightarrow$ cells having the same plaintext values are exposed

  $\Longrightarrow$ all index-plaintext value correspondences are exposed to brute-force attacks

  $\Longrightarrow$ the whole outsourced relation is exposed to brute-force attacks

- Bucket/hash-based index: the same index value may represent different plaintext values
  $\Longrightarrow$ users can only infer with certainty that certain values do not correspond to given cells

Intuitive approach – ACL-based index

Index values directly depend on ACLs

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_{AB}(Rome)	ι_{AB}(2010)	ι_{AB}(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_{AC}(NY)	ι_{AC}(2011)	ι_{AC}(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

Intuitive approach – ACL-based index

Index values directly depend on ACLs

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_{AB}(Rome)	ι_{AB}(2010)	ι_{AB}(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_{AC}(NY)	ι_{AC}(2011)	ι_{AC}(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

+ block inference exposure

– considerable burden at the client side for query translation

Intuitive approach – ACL-based index

Index values directly depend on ACLs

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	**002**	**Rome**	**2010**	**700**
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	I_c	I_y	I_s
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_{AB}(Rome)	ι_{AB}(2010)	ι_{AB}(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_{AC}(NY)	ι_{AC}(2011)	ι_{AC}(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

+ block inference exposure

– considerable burden at the client side for query translation

Ex: query submitted by user B with condition

`Year`=2010

Intuitive approach – ACL-based index

Index values directly depend on ACLs

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	**002**	**Rome**	**2010**	**700**
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_{AB}**(Rome)**	ι_{AB}**(2010)**	ι_{AB}**(700)**
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_{AC}(NY)	ι_{AC}(2011)	ι_{AC}(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

+ block inference exposure

– considerable burden at the client side for query translation

Ex: query submitted by user B with condition

$\texttt{Year}$=2010 $\Longrightarrow$ $\mathtt{I}_y$ IN $\{\iota_B(2010), \iota_{AB}(2010), \iota_{BC}(2010), \iota_{ABC}(2010)\}$

Intuitive approach – User-based index

- Each user u has an index function ι_u that depends on a private piece of information shared with the data owner

- For each cell $t[\mathtt{A}]$ in r and user u in $acl(t)$ there is index value $\iota_u(t[\mathtt{A}])$ in $t^e[\mathtt{I}_\mathtt{A}]$

Intuitive approach – User-based index

- Each user u has an index function ι_u that depends on a private piece of information shared with the data owner

- For each cell $t[\mathtt{A}]$ in *r* and user u in *acl*(t) there is index value $\iota_u(t[\mathtt{A}])$ in $t^e[\mathtt{I}_\mathtt{A}]$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A(700)ι_B(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A(700)ι_C(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

Intuitive approach – User-based index

- Each user u has an index function ι_u that depends on a private piece of information shared with the data owner

- For each cell $t[\mathtt{A}]$ in r and user u in *acl*(t) there is index value $\iota_u(t[\mathtt{A}])$ in $t^e[\mathtt{I}_\mathtt{A}]$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	**City**	**Year**	**Sales**
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	**etuple**	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A(700)ι_B(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A(700)ι_C(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

$\Longrightarrow$ remains vulnerable to inference

Intuitive approach – User-based index

- Each user u has an index function ι_u that depends on a private piece of information shared with the data owner

- For each cell $t[\mathtt{A}]$ in r and user u in $acl(t)$ there is index value $\iota_u(t[\mathtt{A}])$ in $t^e[\mathtt{I_A}]$

SHOPS

	acl		Id	City	Year	Sales
t_1	A	t_1				
t_2	A,B	t_2	002	Rome	2010	700
t_3	B	t_3	003	Rome	2011	600
t_4	A,C	t_4				
t_5	C	t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A(700)ι_B(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A(700)ι_C(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

$\Longrightarrow$ remains vulnerable to inference

Intuitive approach – User-based index

- Each user u has an index function ι_u that depends on a private piece of information shared with the data owner

- For each cell $t[\mathtt{A}]$ in r and user u in $acl(t)$ there is index value $\iota_u(t[\mathtt{A}])$ in $t^e[\mathtt{I}_\mathtt{A}]$

	acl		Id	City	Year	Sales
t_1	A	t_1			**2010**	
t_2	A,B	t_2	002	Rome	**2010**	**700**
t_3	B	t_3	003	Rome	2011	600
t_4	A,C	t_4				**700**
t_5	C	t_5				**700**

SHOPS

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A**(2010)**	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A**(2010)**ι_B**(2010)**	ι_A**(700)**ι_B**(700)**
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A**(700)**ι_C**(700)**
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C**(700)**

SHOPS^e

$\Longrightarrow$ remains vulnerable to inference

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{City}}$

$t_1 \sim_{\mathtt{City}} t_4$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{City}}$

$t_1 \sim_{\mathtt{City}} t_4$

$t_2 \sim_{\mathtt{City}} t_3$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{Year}}$

$t_1 \sim_{\mathtt{City}} t_4$
$t_2 \sim_{\mathtt{City}} t_3$
$t_1 \sim_{\mathtt{Year}} t_2$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{Year}}$

$t_1 \sim_{\mathtt{City}} t_4$
$t_2 \sim_{\mathtt{City}} t_3$
$t_1 \sim_{\mathtt{Year}} t_2$
$t_4 \sim_{\mathtt{Year}} t_5$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{Sales}}$

$t_1 \sim_{\mathtt{City}} t_4$
$t_2 \sim_{\mathtt{City}} t_3$
$t_1 \sim_{\mathtt{Year}} t_2$
$t_4 \sim_{\mathtt{Year}} t_5$
$t_2 \sim_{\mathtt{Sales}} t_4$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\sim_{\mathtt{Sales}}$

$t_1 \sim_{\mathtt{City}} t_4$
$t_2 \sim_{\mathtt{City}} t_3$
$t_1 \sim_{\mathtt{Year}} t_2$
$t_4 \sim_{\mathtt{Year}} t_5$
$t_2 \sim_{\mathtt{Sales}} t_4$
$t_4 \sim_{\mathtt{Sales}} t_5$

Conflicting tuples

- Tuples t_i and t_j are in conflict over attribute $\mathtt{A}$, $t_i \sim_{\mathtt{A}} t_j$, iff
  - have the same value for the attribute
  - can be accessed by different but overlapping sets of users

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$\not\sim_{\mathtt{Sales}}$

$t_1 \sim_{\mathtt{City}} t_4$
$t_2 \sim_{\mathtt{City}} t_3$
$t_1 \sim_{\mathtt{Year}} t_2$
$t_4 \sim_{\mathtt{Year}} t_5$
$t_2 \sim_{\mathtt{Sales}} t_4$
$t_4 \sim_{\mathtt{Sales}} t_5$

Tuple exposure – Example

$t_i \sim_{\mathtt{A}} \ldots \sim_{\mathtt{A}} t_j \Longrightarrow t_i[\mathtt{A}]$ is exposed to all users in $acl(t_j) \setminus acl(t_i)$

$\Longrightarrow t_j[\mathtt{A}]$ is exposed to all users in $acl(t_i) \setminus acl(t_j)$

Tuple exposure – Example

$$t_i \sim_{\mathtt{A}} \ldots \sim_{\mathtt{A}} t_j \implies t_i[\mathtt{A}] \text{ is exposed to all users in } acl(t_j) \setminus acl(t_i)$$
$$\implies t_j[\mathtt{A}] \text{ is exposed to all users in } acl(t_i) \setminus acl(t_j)$$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A(700)ι_B(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A(700)ι_C(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

Tuple exposure – Example

$t_i \sim_{\mathtt{A}} \ldots \sim_{\mathtt{A}} t_j \implies t_i[\mathtt{A}]$ is exposed to all users in $acl(t_j) \setminus acl(t_i)$

$\implies t_j[\mathtt{A}]$ is exposed to all users in $acl(t_i) \setminus acl(t_j)$

SHOPS

	acl		Id	City	Year	Sales
t_1	A	t_1			**2010**	
t_2	A,B	t_2	002	Rome	**2010**	700
t_3	B	t_3	003	Rome	2011	600
t_4	A,C	t_4				
t_5	C	t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A**(2010)**	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A**(2010)**ι_B**(2010)**	ι_A(700)ι_B(700)
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A(700)ι_C(700)
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

Exposures to B

- $t_1 \sim_{\mathtt{Year}} t_2 \implies t_1[\mathtt{Year}]$

Tuple exposure – Example

$t_i \sim_{\mathtt{A}} \ldots \sim_{\mathtt{A}} t_j \implies t_i[\mathtt{A}]$ is exposed to all users in $acl(t_j) \setminus acl(t_i)$
$\implies t_j[\mathtt{A}]$ is exposed to all users in $acl(t_i) \setminus acl(t_j)$

	acl
t_1	A
$\mathbf{t_2}$	$\mathbf{A,B}$
t_3	B
$\mathbf{t_4}$	$\mathbf{A,C}$
t_5	C

SHOPS

	Id	City	Year	Sales
t_1			2010	
t_2	002	Rome	2010	**700**
t_3	003	Rome	2011	600
t_4				**700**
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A**(700)**ι_B**(700)**
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A**(700)**ι_C**(700)**
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C(700)

Exposures to B

- $t_1 \sim_{\mathtt{Year}} t_2 \implies t_1[\mathtt{Year}]$
- $t_2 \sim_{\mathtt{Sales}} t_4 \implies t_4[\mathtt{Sales}]$

Tuple exposure – Example

$t_i \sim_{\mathtt{A}} \ldots \sim_{\mathtt{A}} t_j \implies t_i[\mathtt{A}]$ is exposed to all users in $acl(t_j) \setminus acl(t_i)$

$\implies t_j[\mathtt{A}]$ is exposed to all users in $acl(t_i) \setminus acl(t_j)$

SHOPS

	acl		Id	City	Year	Sales
t_1	A	t_1			2010	
$\boldsymbol{t_2}$	$\boldsymbol{A,B}$	t_2	002	Rome	2010	**700**
t_3	B	t_3	003	Rome	2011	600
$\boldsymbol{t_4}$	$\boldsymbol{A,C}$	t_4				**700**
$\boldsymbol{t_5}$	$\boldsymbol{C}$	t_5				**700**

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY)	ι_A(2010)	ι_A(600)
2	β	ι_A(Rome)ι_B(Rome)	ι_A(2010)ι_B(2010)	ι_A**(700)**ι_B**(700)**
3	γ	ι_B(Rome)	ι_B(2011)	ι_B(600)
4	δ	ι_A(NY)ι_C(NY)	ι_A(2011)ι_C(2011)	ι_A**(700)**ι_C**(700)**
5	ε	ι_C(Oslo)	ι_C(2011)	ι_C**(700)**

Exposures to B

- $t_1 \sim_{\mathtt{Year}} t_2 \implies t_1[\mathtt{Year}]$
- $t_2 \sim_{\mathtt{Sales}} t_4 \implies t_4[\mathtt{Sales}]$
- $t_2 \sim_{\mathtt{Sales}} t_4 \sim_{\mathtt{Sales}} t_5 \implies t_5[\mathtt{Sales}]$

Safe index

- An index function is safe if conflicting tuples have different index values for all the users who can access them
- The index values computed by a safe index function cannot be exploited for inference purposes
- We define a safe index for attribute A by
  - safely partitioning tuples in clusters such that tuples in conflict over A do not belong to the same cluster
  - adopting a different salt for each cluster in the definition of the index function for A
- To minimize the burden at the client side for query translation, the number of salts (i.e., the number of clusters) must be minimized

Conflict graph

- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathbb{A}}(V_{\mathbb{A}}, E_{\mathbb{A}})$ is a non-directed graph with

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

Conflict graph

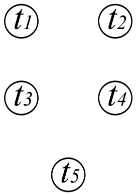
- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ is a non-directed graph with
  - a vertex in $V_{\mathtt{A}}$ for each tuple in *r*

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



Conflict graph

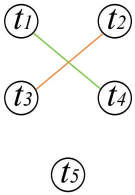
- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ is a non-directed graph with
  - a vertex in $V_{\mathtt{A}}$ for each tuple in r
  - an edge (t_i, t_j) in $E_{\mathtt{A}}$ iff $t_i \sim_{\mathtt{A}} t_j$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



Conflict graph

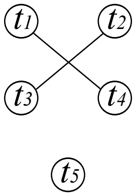
- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ is a non-directed graph with
  - a vertex in $V_{\mathtt{A}}$ for each tuple in *r*
  - an edge (t_i, t_j) in $E_{\mathtt{A}}$ iff $t_i \sim_{\mathtt{A}} t_j$
- A minimum coloring of $G_{\mathtt{A}}$ is a minimum safe partitioning of *r* that solves conflicts w.r.t. $\mathtt{A}$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	**City**	**Year**	**Sales**
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



Conflict graph

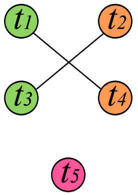
- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ is a non-directed graph with
  - a vertex in $V_{\mathtt{A}}$ for each tuple in r
  - an edge (t_i, t_j) in $E_{\mathtt{A}}$ iff $t_i \sim_{\mathtt{A}} t_j$
- A minimum coloring of $G_{\mathtt{A}}$ is a minimum safe partitioning of r that solves conflicts w.r.t. $\mathtt{A}$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



Safe but not minimum coloring

Conflict graph

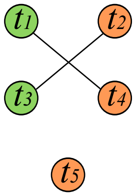
- Our minimization problem is equivalent to the minimum vertex coloring problem
- A conflict graph $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ is a non-directed graph with
  - a vertex in $V_{\mathtt{A}}$ for each tuple in *r*
  - an edge (t_i, t_j) in $E_{\mathtt{A}}$ iff $t_i \sim_{\mathtt{A}} t_j$
- A minimum coloring of $G_{\mathtt{A}}$ is a minimum safe partitioning of *r* that solves conflicts w.r.t. $\mathtt{A}$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



Safe and minimum coloring

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α			
2	β			
3	γ			
4	δ			
5	ε			

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

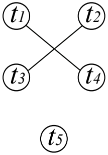
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α			
2	β			
3	γ			
4	δ			
5	ε			

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

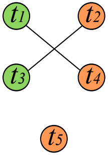
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α			
2	β			
3	γ			
4	δ			
5	ε			

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

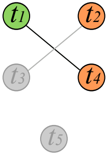
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)		
2	β	ι_A(Rome,s'_A)		
3	γ			
4	δ	ι_A(NY,s'_A)		
5	ε			

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

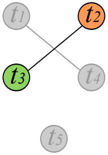
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{City}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)		
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)		
3	γ	ι_B(Rome,s'_B)		
4	δ	ι_A(NY,s'_A)		
5	ε			

Computing a safe index

Index function ι_u for user u over attribute $\mathbb{A}$ is defined applying randomly generated salts to tuples

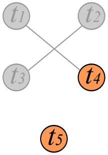
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\texttt{City}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)		
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)		
3	γ	ι_B(Rome,s'_B)		
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)		
5	ε	ι_C(Oslo,s_C)		

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

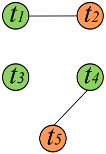
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{Year}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)	ι_A(2010,s_A)	
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)	ι_A(2010,s'_A)ι_B(2010,s_B)	
3	γ	ι_B(Rome,s'_B)	ι_B(2011,s'_B)	
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)	ι_A(2011,s_A)ι_C(2011,s_C)	
5	ε	ι_C(Oslo,s_C)	ι_C(2011,s'_C)	

Computing a safe index

Index function ι_u for user u over attribute $\mathtt{A}$ is defined applying randomly generated salts to tuples

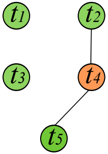
- tuples in different clusters are assigned different salts
- tuples in the same cluster are assigned the same salt

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

$G_{\mathtt{Sales}}$



SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)	ι_A(2010,s_A)	ι_A(600,s_A)
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)	ι_A(2010,s'_A)ι_B(2010,s_B)	ι_A(700,s_A)ι_B(700,s_B)
3	γ	ι_B(Rome,s'_B)	ι_B(2011,s'_B)	ι_B(600,s_B)
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)	ι_A(2011,s_A)ι_C(2011,s_C)	ι_A(700,s'_A)ι_C(700,s_C)
5	ε	ι_C(Oslo,s_C)	ι_C(2011,s'_C)	ι_C(700,s'_C)

Relation level approach – 1

- The conflict graph can also be defined over the whole schema of the outsourced relation, defining a unique partitioning of r

- Each tuple t is associated with a unique salt, used to compute all the index values associated with t

- Conflict graph $G_R(V_R, E_R)$ is a non-directed graph with

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

Relation level approach – 1

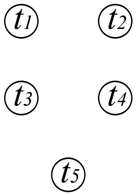
- The conflict graph can also be defined over the whole schema of the outsourced relation, defining a unique partitioning of *r*
- Each tuple *t* is associated with a unique salt, used to compute all the index values associated with *t*
- Conflict graph $G_R(V_R, E_R)$ is a non-directed graph with
  - a vertex in V_R for each tuple in *r*

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

G_{SHOPS}



Relation level approach – 1

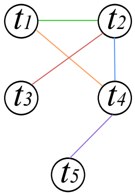
- The conflict graph can also be defined over the whole schema of the outsourced relation, defining a unique partitioning of *r*
- Each tuple *t* is associated with a unique salt, used to compute all the index values associated with *t*
- Conflict graph $G_R(V_R, E_R)$ is a non-directed graph with
  - a vertex in V_R for each tuple in *r*
  - an edge (t_i, t_j) in E_R if $\exists \mathbb{A} \in R$ s.t. $t_i \sim_{\mathbb{A}} t_j$

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

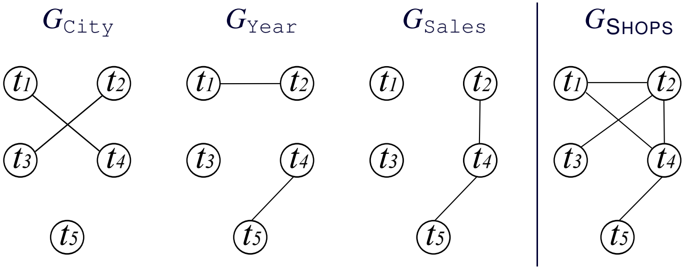
	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

G_{SHOPS}



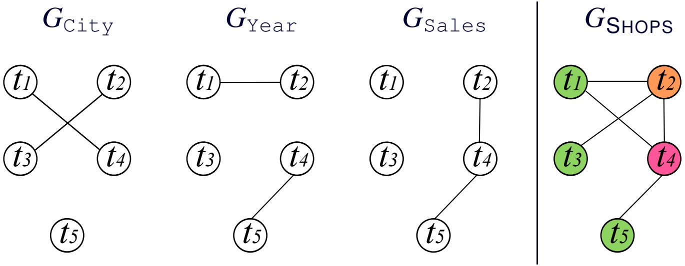
Relation level approach – 2

- Conflict graph $G_R(V_R,E_R)$ can be obtained by composing the conflict graphs $G_{\mathtt{A}}(V_{\mathtt{A}},E_{\mathtt{A}})$ of attributes in R
  - a coloring for G_R is a coloring for $G_{\mathtt{A}}$, with $\mathtt{A}\in R$, but not viceversa
  - a minimum coloring for G_R may not be minimum for $G_{\mathtt{A}}$, with $\mathtt{A}\in R$



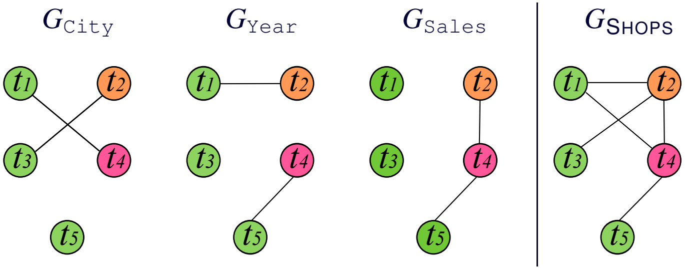
Relation level approach – 2

- Conflict graph $G_R(V_R, E_R)$ can be obtained by composing the conflict graphs $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ of attributes in R
  - a coloring for G_R is a coloring for $G_{\mathtt{A}}$, with $\mathtt{A} \in R$, but not viceversa
  - a minimum coloring for G_R may not be minimum for $G_{\mathtt{A}}$, with $\mathtt{A} \in R$



Relation level approach – 2

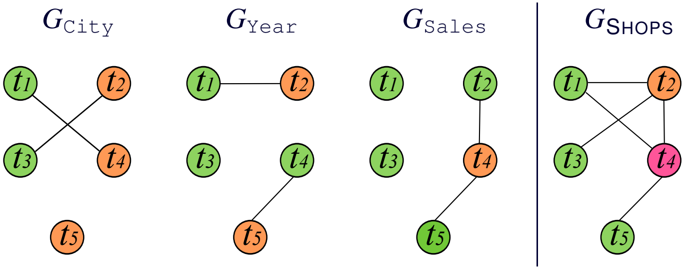
- Conflict graph $G_R(V_R,E_R)$ can be obtained by composing the conflict graphs $G_{\mathtt{A}}(V_{\mathtt{A}},E_{\mathtt{A}})$ of attributes in R
  - a coloring for G_R is a coloring for $G_{\mathtt{A}}$, with $\mathtt{A}\in R$, but not viceversa
  - a minimum coloring for G_R may not be minimum for $G_{\mathtt{A}}$, with $\mathtt{A}\in R$



Safe but not minimum coloring

Relation level approach – 2

- Conflict graph $G_R(V_R, E_R)$ can be obtained by composing the conflict graphs $G_{\mathtt{A}}(V_{\mathtt{A}}, E_{\mathtt{A}})$ of attributes in R
  - a coloring for G_R is a coloring for $G_{\mathtt{A}}$, with $\mathtt{A} \in R$, but not viceversa
  - a minimum coloring for G_R may not be minimum for $G_{\mathtt{A}}$, with $\mathtt{A} \in R$



Safe and minimum coloring

Query evaluation

- Each user u knows
  - index function ι_u
  - the maximum number of salts $n_{\mathtt{A},u}$ used to define the index for attribute $\mathtt{A}$
  - the pseudo-random function used to generate salts
- Condition $\mathtt{A}=v$ in a query submitted by user u is translated as $\mathtt{I}_{\mathtt{A}}$ IN V, with
  - $\mathtt{I}_{\mathtt{A}}$: index over $\mathtt{A}$
  - $V=\{\iota_u(v,s_1),\ldots,\iota_u(v,s_{n_{\mathtt{A},u}})\}$: values obtained applying ι_u to v combined with each of the $n_{\mathtt{A},u}$ salts

Query evaluation – Example

SHOPS

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

	Id	City	Year	Sales
t_1	001	NY	2010	600
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4	004	NY	2011	700
t_5	005	Oslo	2011	700

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)	ι_A(2010,s_A)	ι_A(600,s_A)
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)	ι_A(2010,s'_A)ι_B(2010,s_B)	ι_A(700,s_A)ι_B(700,s_B)
3	γ	ι_B(Rome,s'_B)	ι_B(2011,s'_B)	ι_B(600,s_B)
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)	ι_A(2011,s_A)ι_C(2011,s_C)	ι_A(700,s'_A)ι_C(700,s_C)
5	ε	ι_C(Oslo,s_C)	ι_C(2011,s'_C)	ι_C(700,s'_C)

Query evaluation – Example

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	002	Rome	2010	700
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	$\mathtt{I}_c$	$\mathtt{I}_y$	$\mathtt{I}_s$
1	α	ι_A(NY,s_A)	ι_A(2010,s_A)	ι_A(600,s_A)
2	β	ι_A(Rome,s'_A)ι_B(Rome,s_B)	ι_A(2010,s'_A)ι_B(2010,s_B)	ι_A(700,s_A)ι_B(700,s_B)
3	γ	ι_B(Rome,s'_B)	ι_B(2011,s'_B)	ι_B(600,s_B)
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)	ι_A(2011,s_A)ι_C(2011,s_C)	ι_A(700,s'_A)ι_C(700,s_C)
5	ε	ι_C(Oslo,s_C)	ι_C(2011,s'_C)	ι_C(700,s'_C)

Query by B, who has 2 salts for `Year`

```
SELECT City, Sales
FROM   SHOPS
WHERE  Year=2010
```

$\Longrightarrow$

Query evaluation – Example

	acl
t_1	A
t_2	A,B
t_3	B
t_4	A,C
t_5	C

SHOPS

	Id	City	Year	Sales
t_1				
t_2	**002**	**Rome**	**2010**	**700**
t_3	003	Rome	2011	600
t_4				
t_5				

SHOPS^e

tid	etuple	I_c	I_y	I_s
1	α	ι_A(NY,s_A)	ι_A(2010,s_A)	ι_A(600,s_A)
2	β	ι_A**(Rome,**s'_A**)**ι_B**(Rome,**s_B**)**	ι_A**(2010,**s'_A**)**ι_B**(2010,**s_B**)**	ι_A**(700,**s_A**)**ι_B**(700,**s_B**)**
3	γ	ι_B(Rome,s'_B)	ι_B(2011,s'_B)	ι_B(600,s_B)
4	δ	ι_A(NY,s'_A)ι_C(NY,s_C)	ι_A(2011,s_A)ι_C(2011,s_C)	ι_A(700,s'_A)ι_C(700,s_C)
5	ε	ι_C(Oslo,s_C)	ι_C(2011,s'_C)	ι_C(700,s'_C)

Query by B, who has 2 salts for `Year` translates to

SELECT `City, Sales`
FROM SHOPS
WHERE `Year`=2010

$\Longrightarrow$

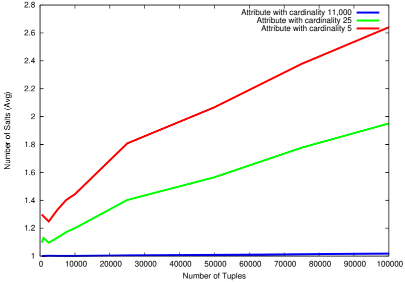
SELECT `etuple`
FROM SHOPS^e
WHERE I_y IN $\{\iota_B(2010,s_B),\iota_B(2010,s'_B)\}$

Experimental results – 1

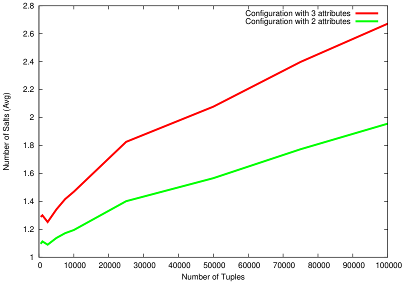
- Relational table built starting from the TPC-H benchmark
  - three attributes with 5, 25, and 11,000 distinct values
  - from 500 to 100,000 tuples
- Access control policy obtained extracting the authorship information from the DBLP repository
  - each paper is represented by a tuple in the table
  - each author can access all and only her papers
- Attribute level and relation level approaches compared w.r.t.
  - the number of clusters composing a safe partitioning (i.e., upper bound of the number of salts required)
  - the average number of salts per user (i.e., user overhead in query translation)

Experimental results – 2

Attribute level



Relation level



- Attribute level salt, three attributes:
  - cardinality 5
  - cardinality 25
  - cardinality 11000
- Relation level salt, two relations:
  - three attributes, with cardinality 5, 25, 11000
  - two attributes, with cardinality 25, 11000

Experimental results – 3

Specifying salts at the attribute level (in contrast to relation)

+ permits to reduce the overhead of queries with condition on the most selective attributes (the difference for non-selective attributes is minimal)

– requires storing a different value for the number of salts for every attribute (in contrast to a value for the whole relation), for every user

$\Longrightarrow$ If queries over selective attributes are more frequent: the attribute level approach is preferred; otherwise, the relation level approach is preferred for its simplicity and limited storage overhead

Some open issues

- Protect against the server observing multiple queries
- Protect against collusion between users and server
- Use of indexes associated with clusters of tuples in contrast to individual tuples

Indexes and Fragmentation

S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “On Information Leakage by Indexes over Data Fragments,” in *Proc. of PrivDB*, Brisbane, Australia, April 2013.

Information exposure

+ Provides effectiveness and efficiency in query execution
  - enables the partial server-side evaluation of selection conditions over encrypted attributes

– Indexes combined with fragmentation can cause information leakage of confidential (encrypted or fragmented) information
  - exposure to leakage varies depending on the kind of indexes

Kinds of knowledge

A curious observer can exploit

F_1^e

<u>**salt**</u>	**enc**	**Name**	**State**	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

F_2^e

<u>**salt**</u>	**enc**	**Disease**
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

Kinds of knowledge

A curious observer can exploit

- vertical knowledge due to values appearing in the clear in one fragment and indexed in other fragments

F_1^e

salt	**enc**	**Name**	**State**	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	**Disease**
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

Kinds of knowledge

A curious observer can exploit

- vertical knowledge due to values appearing in the clear in one fragment and indexed in other fragments
- horizontal knowledge due to external knowledge of the presence of specific tuples in the table

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Direct index

F_1^e

salt	enc	Name	State	$\mathrm{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Direct index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

Direct index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- $\iota(\text{Flu}) = \alpha$
- $\iota(\text{Gastritis}) = \gamma$

Direct index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- ι(Flu) = $\alpha \implies$ Adams, Brown, Cooper have Flu
- ι(Gastritis) = $\gamma \implies$ Falk has Gastritis
- the other patients have Diabetes or Arthritis with $p = 50\%$

Direct index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

Direct index

F_1^e

salt	**enc**	**Name**	**State**	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	**enc**	**Disease**
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	**Disease**
Adams	Flu

Horizontal knowledge

- ι(Flu) = α

Direct index

F_1^e

salt	enc	Name	State	$\mathrm{i_d}$
s_{11}	t_{11}^e	Adams	VA	α
s_{12}	t_{12}^e	Brown	MN	α
s_{13}	t_{13}^e	Cooper	CA	α
s_{14}	t_{14}^e	Davis	VA	β
s_{15}	t_{15}^e	Eden	NY	β
s_{16}	t_{16}^e	Falk	CA	γ
s_{17}	t_{17}^e	Green	NY	δ
s_{18}	t_{18}^e	Hack	NY	δ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

- ι(Flu) = $\alpha \Longrightarrow$ also Brown and Cooper have Flu

Bucket index

F_1^e

salt	**enc**	**Name**	**State**	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	**enc**	**Disease**
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	**Disease**
Adams	Flu

Bucket index

F_1^e

salt	enc	Name	State	$\mathrm{i_d}$
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- $\iota(\text{Flu}) = \zeta$

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- $\iota(\text{Flu}) = \zeta \implies \iota(\text{Gastritis}) = \zeta$

Bucket index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t^e_{11}	Adams	VA	ζ
s_{12}	t^e_{12}	Brown	MN	ζ
s_{13}	t^e_{13}	Cooper	CA	ζ
s_{14}	t^e_{14}	Davis	VA	η
s_{15}	t^e_{15}	Eden	NY	η
s_{16}	t^e_{16}	Falk	CA	ζ
s_{17}	t^e_{17}	Green	NY	θ
s_{18}	t^e_{18}	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t^e_{21}	Flu
s_{22}	t^e_{22}	Flu
s_{23}	t^e_{23}	Flu
s_{24}	t^e_{24}	Diabetes
s_{25}	t^e_{25}	Diabetes
s_{26}	t^e_{26}	Gastritis
s_{27}	t^e_{27}	Arthritis
s_{28}	t^e_{28}	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- ι(Flu) = ι(Gastritis) = $\zeta \implies$ Adams, Brown, Cooper, and Falk have Flu with $p = 75\%$, Gastritis with $p = 25\%$

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

- ι(Flu) = ζ

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

- ι(Flu) = $\zeta \implies$ no inference

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical and Horizontal knowledge

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical and Horizontal knowledge

- ι(Flu) = ι(Gastritis) = ζ

Bucket index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical and Horizontal knowledge

- ι(Flu) = ι(Gastritis) = $\zeta \Longrightarrow$ Brown, Cooper, and Falk have Flu with $p = 66\%$, Gastritis with $p = 33\%$

Bucket index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Davis	Diabetes

Vertical and Horizontal knowledge

Bucket index

F_1^e

salt	enc	Name	State	$\mathrm{i_d}$
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Davis	Diabetes

Vertical and Horizontal knowledge

- ι(Diabetes) = η

Bucket index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	ζ
s_{12}	t_{12}^e	Brown	MN	ζ
s_{13}	t_{13}^e	Cooper	CA	ζ
s_{14}	t_{14}^e	Davis	VA	η
s_{15}	t_{15}^e	Eden	NY	η
s_{16}	t_{16}^e	Falk	CA	ζ
s_{17}	t_{17}^e	Green	NY	θ
s_{18}	t_{18}^e	Hack	NY	θ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Davis	Diabetes

Vertical and Horizontal knowledge

- ι(Diabetes) = $\eta \implies$ Eden has Diabetes

Flattened index

F_1^e

salt	enc	Name	State	$\mathrm{i_d}$
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Flattened index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

Flattened index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Vertical knowledge

- each correspondence between plaintext and index values is equally like

Flattened index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

Flattened index

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

- ι(Flu) = κ

Flattened index

F_1^e

salt	enc	Name	State	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	κ
s_{12}	t_{12}^e	Brown	MN	λ
s_{13}	t_{13}^e	Cooper	CA	μ
s_{14}	t_{14}^e	Davis	VA	ν
s_{15}	t_{15}^e	Eden	NY	ξ
s_{16}	t_{16}^e	Falk	CA	π
s_{17}	t_{17}^e	Green	NY	ρ
s_{18}	t_{18}^e	Hack	NY	σ

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

Horizontal knowledge

- ι(Flu) = $\kappa \Longrightarrow$ no inference

Intuitive approach: flattening and collisions

F_1^e

salt	**enc**	**Name**	**State**	$\mathbf{i_d}$
s_{11}	t_{11}^e	Adams	VA	ϕ
s_{12}	t_{12}^e	Brown	MN	ϕ
s_{13}	t_{13}^e	Cooper	CA	ψ
s_{14}	t_{14}^e	Davis	VA	χ
s_{15}	t_{15}^e	Eden	NY	χ
s_{16}	t_{16}^e	Falk	CA	ψ
s_{17}	t_{17}^e	Green	NY	ω
s_{18}	t_{18}^e	Hack	NY	ω

vertical knowledge

salt	**enc**	**Disease**
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	**Disease**
Adams	Flu

+ blocks inference exposure

Intuitive approach: flattening and collisions

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ϕ
s_{12}	t_{12}^e	Brown	MN	ϕ
s_{13}	t_{13}^e	Cooper	CA	ψ
s_{14}	t_{14}^e	Davis	VA	χ
s_{15}	t_{15}^e	Eden	NY	χ
s_{16}	t_{16}^e	Falk	CA	ψ
s_{17}	t_{17}^e	Green	NY	ω
s_{18}	t_{18}^e	Hack	NY	ω

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

+ blocks inference exposure
– exposed to inferences exploiting dynamic observations

Intuitive approach: flattening and collisions

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ϕ
s_{12}	t_{12}^e	Brown	MN	ϕ
s_{13}	t_{13}^e	Cooper	CA	ψ
s_{14}	t_{14}^e	Davis	VA	χ
s_{15}	t_{15}^e	Eden	NY	χ
s_{16}	t_{16}^e	Falk	CA	ψ
s_{17}	t_{17}^e	Green	NY	ω
s_{18}	t_{18}^e	Hack	NY	ω

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

+ blocks inference exposure

– exposed to inferences exploiting dynamic observations

Disease='Flu' translates to i_d IN $\{\phi,\psi\} \Longrightarrow \iota$(Flu)=$\{\phi,\psi\}$

Intuitive approach: flattening and collisions

F_1^e

salt	enc	Name	State	i_d
s_{11}	t_{11}^e	Adams	VA	ϕ
s_{12}	t_{12}^e	Brown	MN	ϕ
s_{13}	t_{13}^e	Cooper	CA	ψ
s_{14}	t_{14}^e	Davis	VA	χ
s_{15}	t_{15}^e	Eden	NY	χ
s_{16}	t_{16}^e	Falk	CA	ψ
s_{17}	t_{17}^e	Green	NY	ω
s_{18}	t_{18}^e	Hack	NY	ω

vertical knowledge

salt	enc	Disease
s_{21}	t_{21}^e	Flu
s_{22}	t_{22}^e	Flu
s_{23}	t_{23}^e	Flu
s_{24}	t_{24}^e	Diabetes
s_{25}	t_{25}^e	Diabetes
s_{26}	t_{26}^e	Gastritis
s_{27}	t_{27}^e	Arthritis
s_{28}	t_{28}^e	Arthritis

horizontal

Name	Disease
Adams	Flu

+ blocks inference exposure

– exposed to inferences exploiting dynamic observations

Disease='Flu' translates to i_d IN $\{\phi,\psi\} \Longrightarrow \iota$(Flu)=$\{\phi,\psi\}$

ι(Flu)=$\{\phi,\psi\} \Longrightarrow$ Brown, Cooper, Frank have Flu with $p = 66\%$

Still several open issues

- Protection against observation of accesses to fragments
- Protection against the release of multiple indexes
  - multiple indexes in the same fragment
  - indexes on the same attribute in multiple fragments
  - two attributes appear one in plaintext and the other indexed in one fragment and reversed in another fragment
- Protection against different types of observer's knowledge
- Development of flattened index functions that generate collisions
- Definition of metrics for assessing exposures due to indexes
- ...

References – 1

- [ABGGKMSTX-05] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, Y. Xu, “Two Can Keep a Secret: A Distributed Architecture for Secure Database Services,” in *Proc. of CIDR*, Asilomar, CA, USA, January 2005.
- [AFB-05] M. Atallah, K. Frikken, M. Blanton, “Dynamic and Efficient Key Management for Access Hierarchies,” in *Proc. of CCS*, Alexandria, VA, USA, November 2005.
- [AKSX-04] R. Agrawal, J. Kierman, R. Srikant, Y. Xu, “Order Preserving Encryption for Numeric Data,” in *Proc. of ACM SIGMOD*, Paris, France, 2004.
- [AT-83] S. Akl, P. Taylor, “Cryptographic Solution to a Problem of Access Control in a Hierarchy,” *ACM TOCS*, vol. 1, no. 3, August 1983.
- [B-70] B.H. Bloom, “Trade-offs in Hash Coding with Allowable Error,” in *Communication of the ACM*, vol. 13, no. 7, July 1970.
- [BV11] Z. Brakerski, V. Vaikuntanathan, “Efficient Fully Homomorphic Encryption from (standard) LWE,” in Proc. of FOCS, Palm Springs, CA, USA, October 2011.

References – 2

- [CDDJPS-05] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, P. Samarati, "Modeling and Assessing Inference Exposure in Encrypted Databases," in *ACM TISSEC*, vol. 8, no. 1, February 2005.
- [CDFJPS-07] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Fragmentation and Encryption to Enforce Privacy in Data Storage," in *Proc. of ESORICS*, Dresden, Germany, September 2007.
- [CDFJPS-09a] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases," in *Proc. of ICDCS*, Montreal, Quebec, Canada, June 2009.
- [CDFJPS-09b] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Keep a Few: Outsourcing Data while Maintaining Confidentiality," in *Proc. of ESORICS*, Saint Malo, France, September 2009.
- [CDFJPS-10] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Combining Fragmentation and Encryption to Protect Privacy in Data Storage," in *ACM TISSEC*, vol. 13, no. 3, July 2010.
- [CMW-06] J. Crampton, K. Martin, P. Wild, "On Key Assignment for Hierarchical Access Control," in *Proc. of CSFW*, Venice, Italy, July 2006.

References – 3

- [CSYZ-08] G. Cormode, D. Srivastava, T. YU, Q. Zhang, “Anonymizing Bipartite Graph Data Using Safe Groupings,” in *Proc. of VLDB*, Auckland, New Zealand, August 2008.
- [DFJL-12] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, “Enforcing Subscription-based Authorization Policies in Cloud Scenarios,” in *Proc. of DBSec*, Paris, France, July 2012.
- [DFJLPS-14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, P. Samarati, “Fragmentation in Presence of Data Dependencies,” in *IEEE TDSC*, 2014 (to appear).
- [DFJPPS-10] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Pelosi, P. Samarati, “Encryption-based Policy Enforcement for Cloud Storage,” in *Proc. of SPCC 2010,* Genova, Italy, June 2010.
- [DFJPS-07] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Over-encryption: Management of Access Control Evolution on Outsourced Data,” in *Proc. of VLDB*, Vienna, Austria, 2007.
- [DFJPS-10a] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Fragments and Loose Associations: Respecting Privacy in Data Publishing,” in *Proc. of the VLDB Endowment*, vol. 3, no. 1, September 2010.

References – 4

- [DFJPS-10b] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Encryption Policies for Regulating Access to Outsourced Data,” in *ACM TODS*, vol. 35, no. 2, April 2010.
- [DFJPS-11] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Private Data Indexes for Selective Access to Outsourced Data,” in *Proc. of WPES*, Chicago, IL, USA, October 2011.
- [DFJPS-12] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “Support for Write Privileges on Outsourced Data,” in *Proc. of SEC*, Heraklion, Crete, Greece, June 2012.
- [DFJPS-13] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, “On Information Leakage by Indexes over Data Fragments,” in *Proc. of PrivDB*, Brisbane, Australia, April 2013.
- [DFM-04] A. De Santis, A.L. Ferrara, B. Masucci, “Cryptographic Key Assignment Schemes for any Access Control Policy,” in *Inf. Process. Lett.*, vol. 92, no. 4, 2004.
- [G-09] C. Gentry, “Fully Homomorphic Encryption using Ideal Lattices,” in *Proc. of STOC*, Bethesda, MA, USA, 2009.

References – 5

- [G-80] E. Gudes, “The Design of a Cryptography based Secure File System,” in *IEEE TSE*, vol. 6, no. 5, September 1980.
- [GKPVZ-13] S. Goldwasser, Y.T. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich, “Reusable Garbled Circuits and Succinct Functional Encryption,” in *Proc. of STOC*, Palo Alto, CA, USA, June 2013.
- [GSW13] C. Gentry, A. Sahai, B. Waters, “Homomorphic Encryption from Learning with Errors: Conceptually-simpler, Asymptotically-faster, Attribute-based,” in Proc. of CRYPTO, Santa Barbara, CA, USA, August 2013.
- [HIML-02] H. Hacigümüş, B. Iyer, S. Mehrotra, C. Li, “Executing SQL over Encrypted Data in the Database-Service-Provider Model,” in *Proc. of the ACM SIGMOD*, Madison, Wisconsin, USA, June 2002.
- [HL-90] L. Harn, H. Lin, “A Cryptographic Key Generation Scheme for Multilevel Data Security,” *Computers and Security*, vol. 9, no. 6, October 1990.
- [HY-03] M. Hwang, W. Yang, “Controlling Access in Large Partially Ordered Hierarchies using Cryptographic Keys,” *The Journal of Systems and Software*, vol. 67, no. 2, August 2003.

References – 6

- [LWL-89] H. Liaw, S. Wang, C. Lei, "On the Design of a Single-Key-Lock Mechanism Based on Newton's Interpolating Polynomial," in *IEEE TSE*, vol. 15, no. 9, September 1989.
- [M-85] S. MacKinnon et al., "An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy," in *IEEE TC,* vol. 34, no. 9, September 1985.
- [MNT-06] E. Mykletun, M. Narasimha, G. Tsudik, "Authentication and Integrity in Outsourced Databases," in *ACM TS*, vol. 2, no. 2, May 2006.
- [S-87] R. Sandhu, "On Some Cryptographic Solutions for Access Control in a Tree Hierarchy," in *Proc. of the 1987 Fall Joint Computer Conference on Exploring Technology: Today and Tomorrow*, Dallas, TX, USA, October 1987.
- [S-88] R. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," in *Information Processing Letters*, vol. 27, no. 2, 1988.
- [SC-02] V. Shen, T. Chen, "A Novel Key Management Scheme Based on Discrete Logarithms and Polynomial Interpolations," in *Computers and Security*, vol. 21, no. 2, March 2002.

References – 7

- [WL-06] H. Wang, Laks V. S. Lakshmanan, "Efficient Secure Query Evaluation over Encrypted XML Databases," in *Proc. of VLDB*, Seoul, Korea, September 2006.